

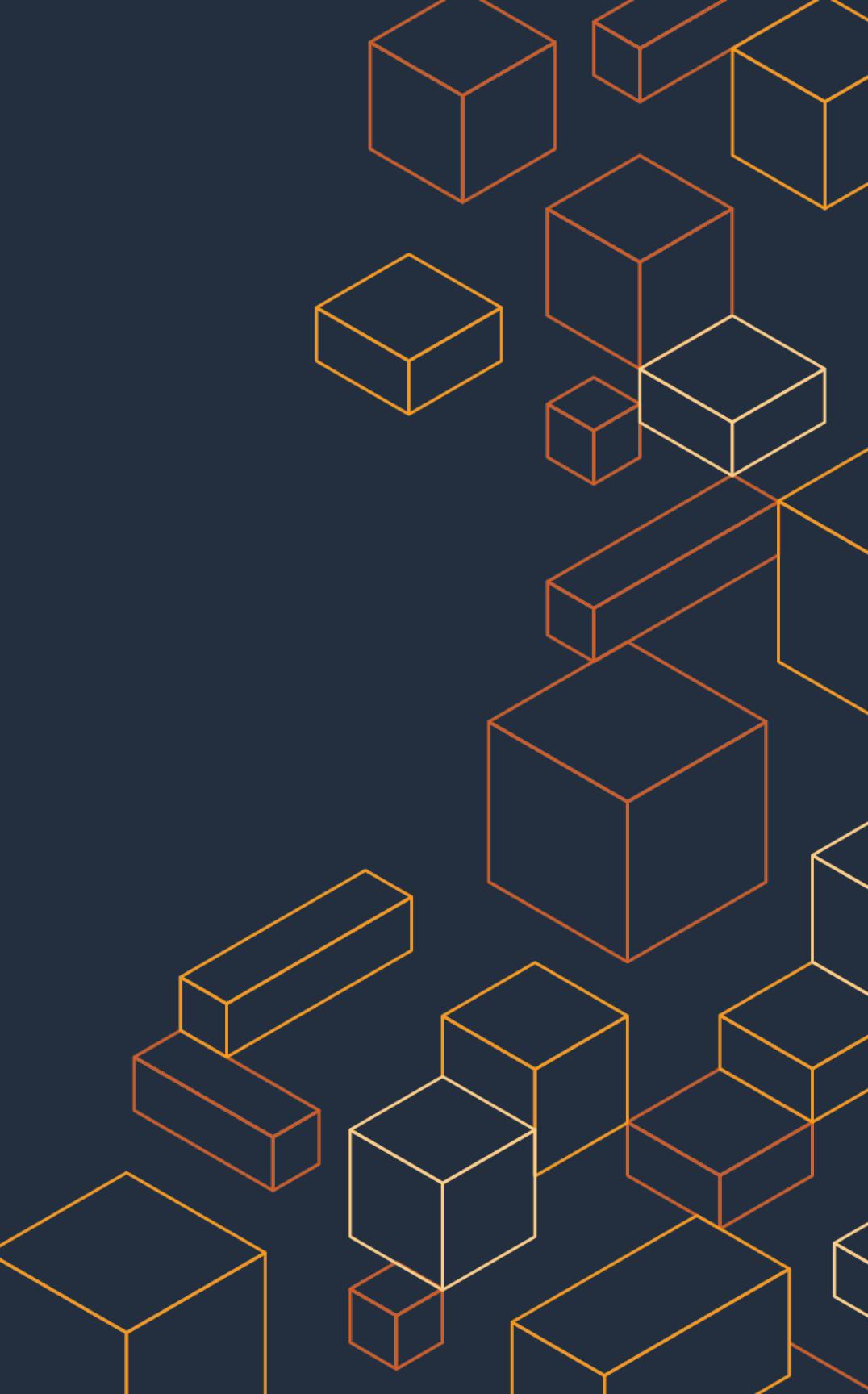


Serverless

Desbravando erros e falhas

Rafael Barbosa

Cloud Application Architect



Agenda

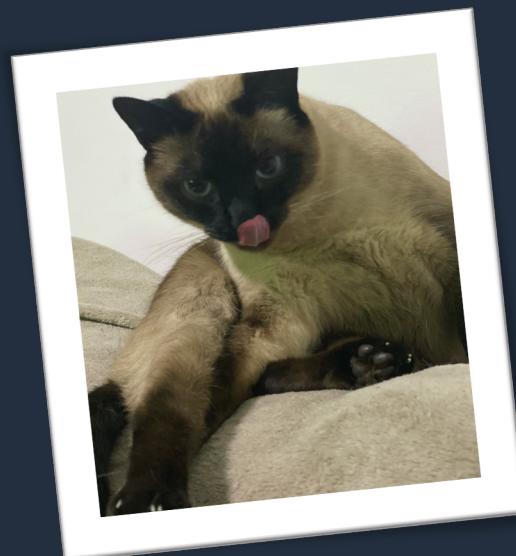
- whoami
- Fases de implementação e erros
 - Amazon Simple Notification Service (SNS) + AWS Lambda
 - Amazon Simple Queue Service (SQS)
 - Amazon DynamoDB
 - Amazon API Gateway
- Resumo

\$whoami



Rafael Barbosa

- Pai de gatos
- Car detailer Amador
- Cloud Application Architect
- Professor de MBA
- 6 anos de serverless



Fase 1

Implementando notificações

Fase 1 – Implementando notificações



Amazon Simple Notification Service
(SNS)

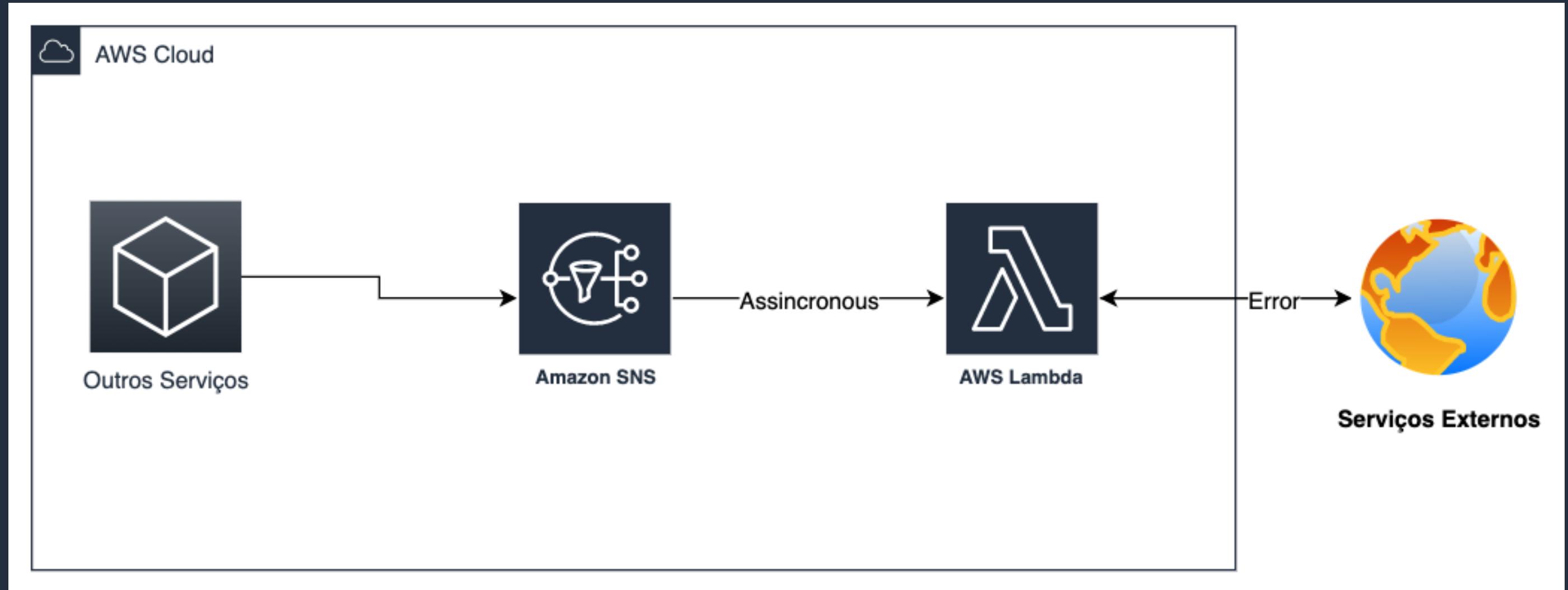
- publicação/assinatura
- Tópicos para mensagens de alto rendimento
- Enviando mensagens para:
 - Amazon SQS
 - AWS Lambda
 - HTTP/S
 - Email
 - SMS
 - Notificação push

Fase 1 – Implementando notificações

- Execute código sem provisionar ou gerenciar servidores
- Escalabilidade contínua
- Medidor de fração de segundo
- Performance uniforme



Fase 1 – Arquitetura com erros



Casos de falha de AWS Lambda

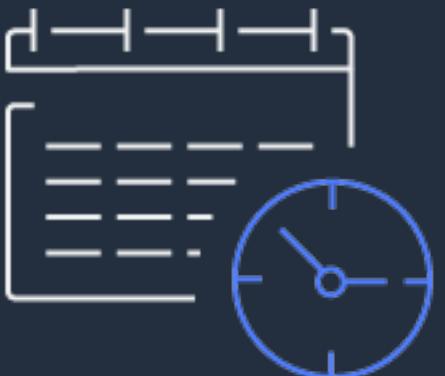


Exceção no Código executado:

- Serviço externo falhou
- Entrada errada
- Bug no código

Timeout:

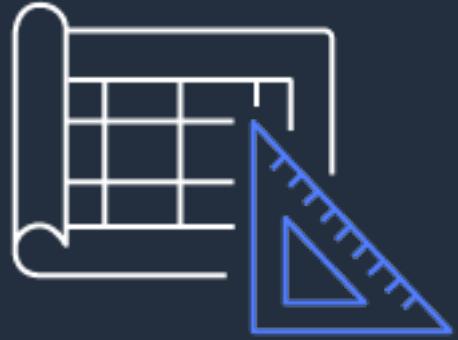
- Código demorou mais que o configurado



Sem memória:

- O processo precisou de mais memória RAM que o configurado

Tipos de evento AWS Lambda

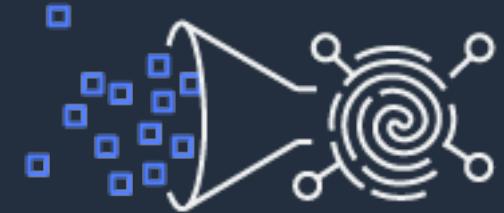


Síncrono:

- Amazon SQS
- Amazon API Gateway
- Chamada pelo SDK

Assíncrono :

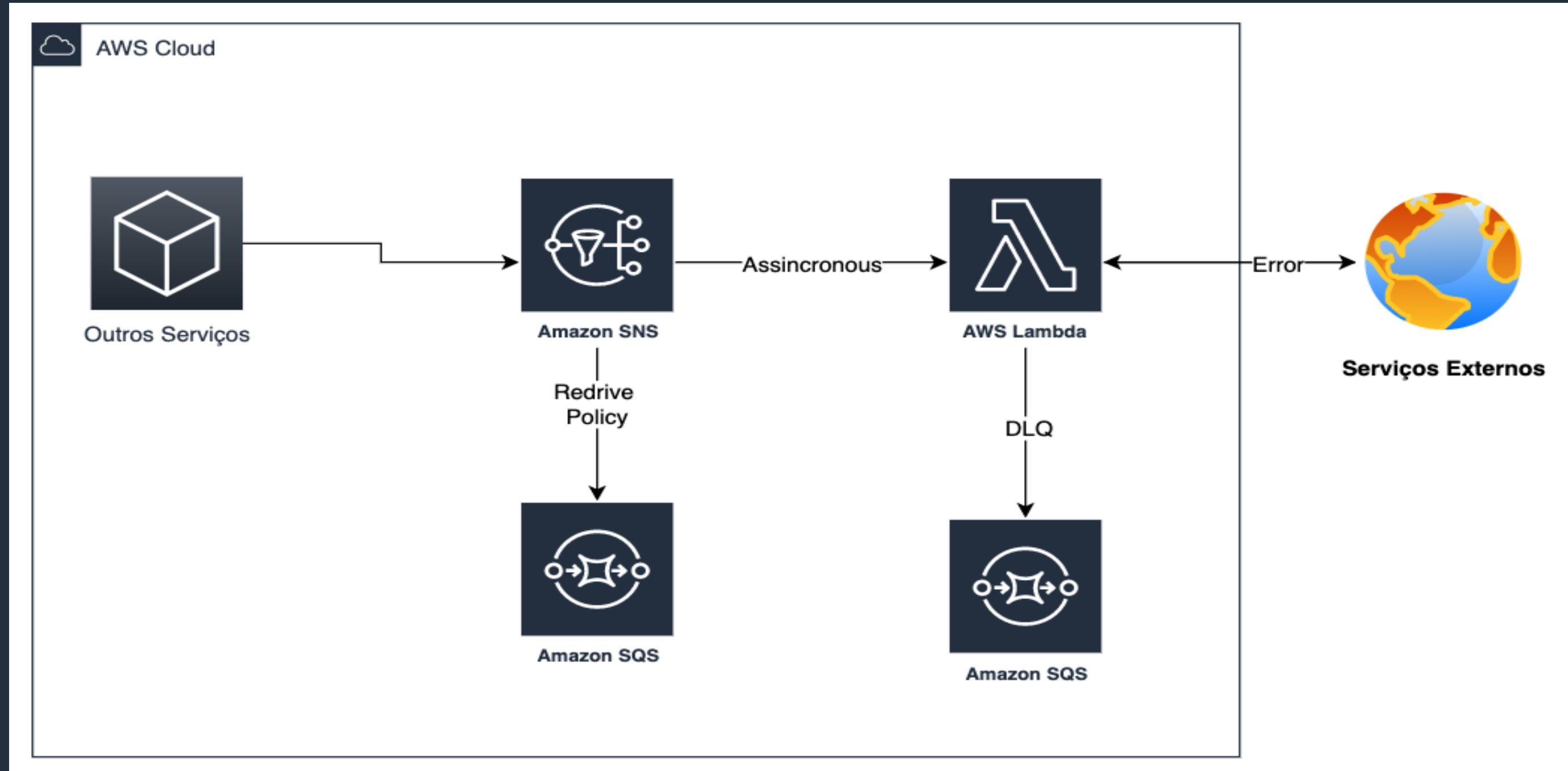
- Amazon SNS
- Amazon EventBridge
- Amazon API Gateway
- Amazon Simple Storage Service (S3)



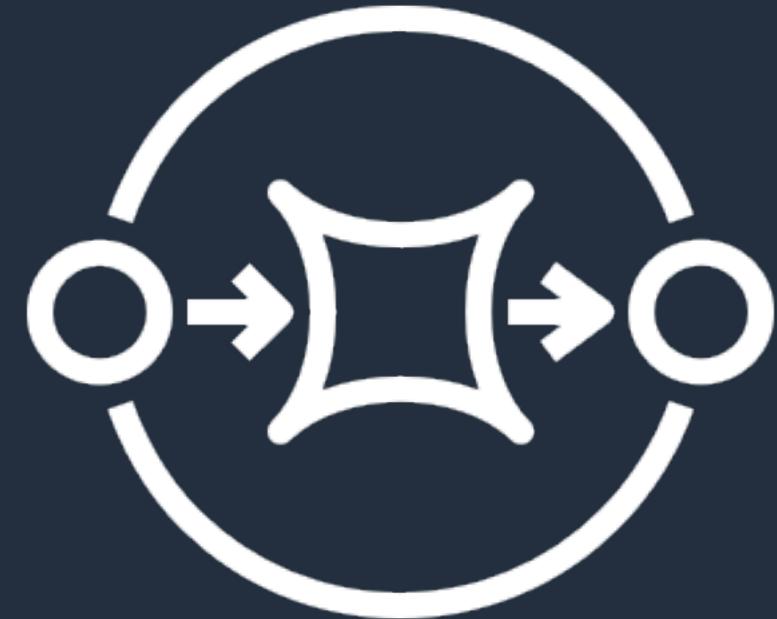
Streaming:

- DynamoDB streams
- Amazon Kinesis Data Streams

Fase 1 – Arquitetura com tratamento de erros



DLQ – Dead-Letter Queue



Amazon Simple Queue Service
(SQS)

Destino para mensagens que não podem ser processadas (consumidas) com sucesso. Dead-letter queues são úteis para depurar seu aplicativo ou sistema de mensagens porque permitem que você isole mensagens problemáticas para determinar porque seu processamento não é bem-sucedido.
A atuação na DLQ pode ser automática ou manual

Fase 2

Implementando novas APIs e backends

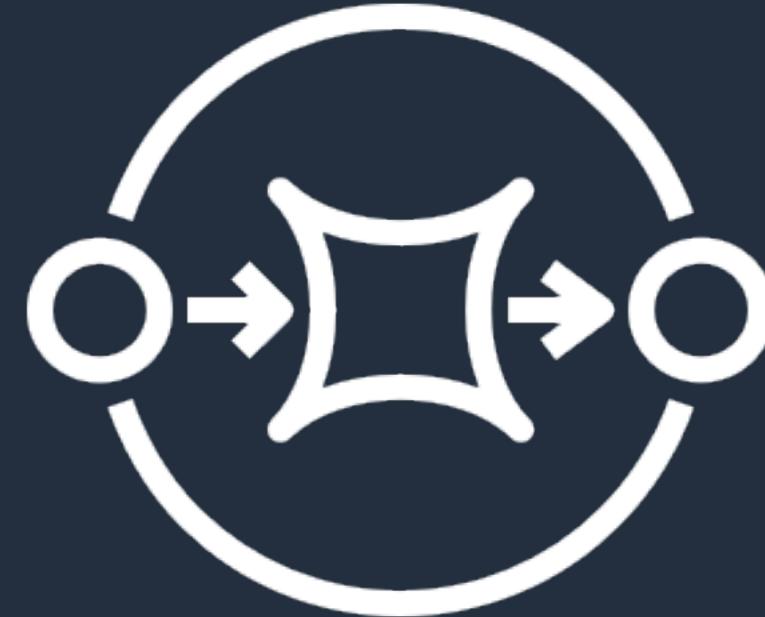
Fase 2 – Implementando novas APIs e backends



Amazon API gateway

- Request/response
- Métodos HTTP como GET e POST
- Comunicação short-lived
- Stateless

Fase 2 – Implementando novas APIs e backends



Amazon Simple Queue Service
(SQS)

- Filas escaláveis e de fácil implementação
- Taxa de transferência ilimitada
- Entrega pelo menos uma vez
- Melhor ordenação possível
- Filas e mensagens ilimitadas
- Criptografia no lado do servidor (SSE)

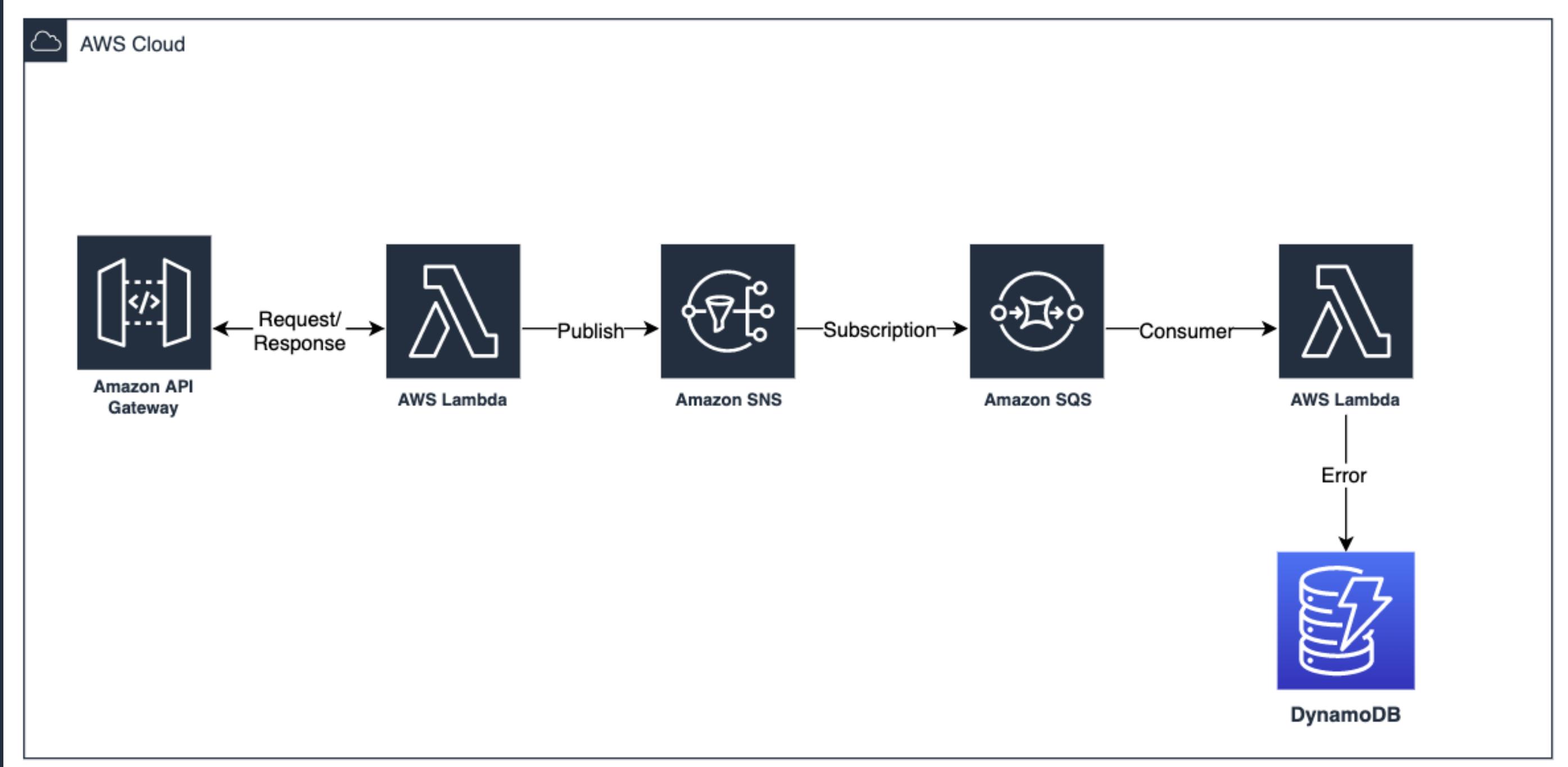
Fase 2 – Implementando novas APIs e backends



Amazon DynamoDB

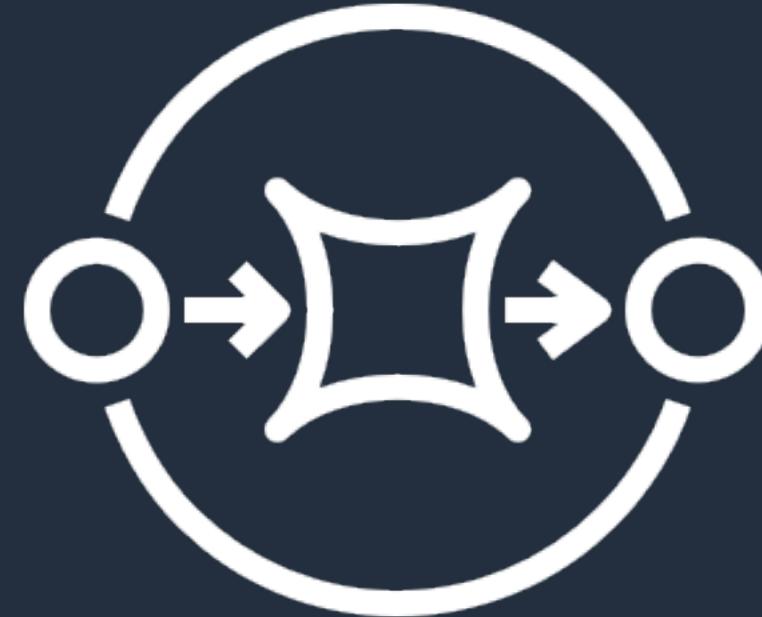
- Totalmente gerenciado, Serviço de banco de dados cloud-native NoSQL
- Projetado para OLTPs críticos:
- Banco de dados operacional que provem:
 - Escalabilidade horizontal extrema
 - Consistência de performance em escala

Fase 2 – Arquitetura com erros



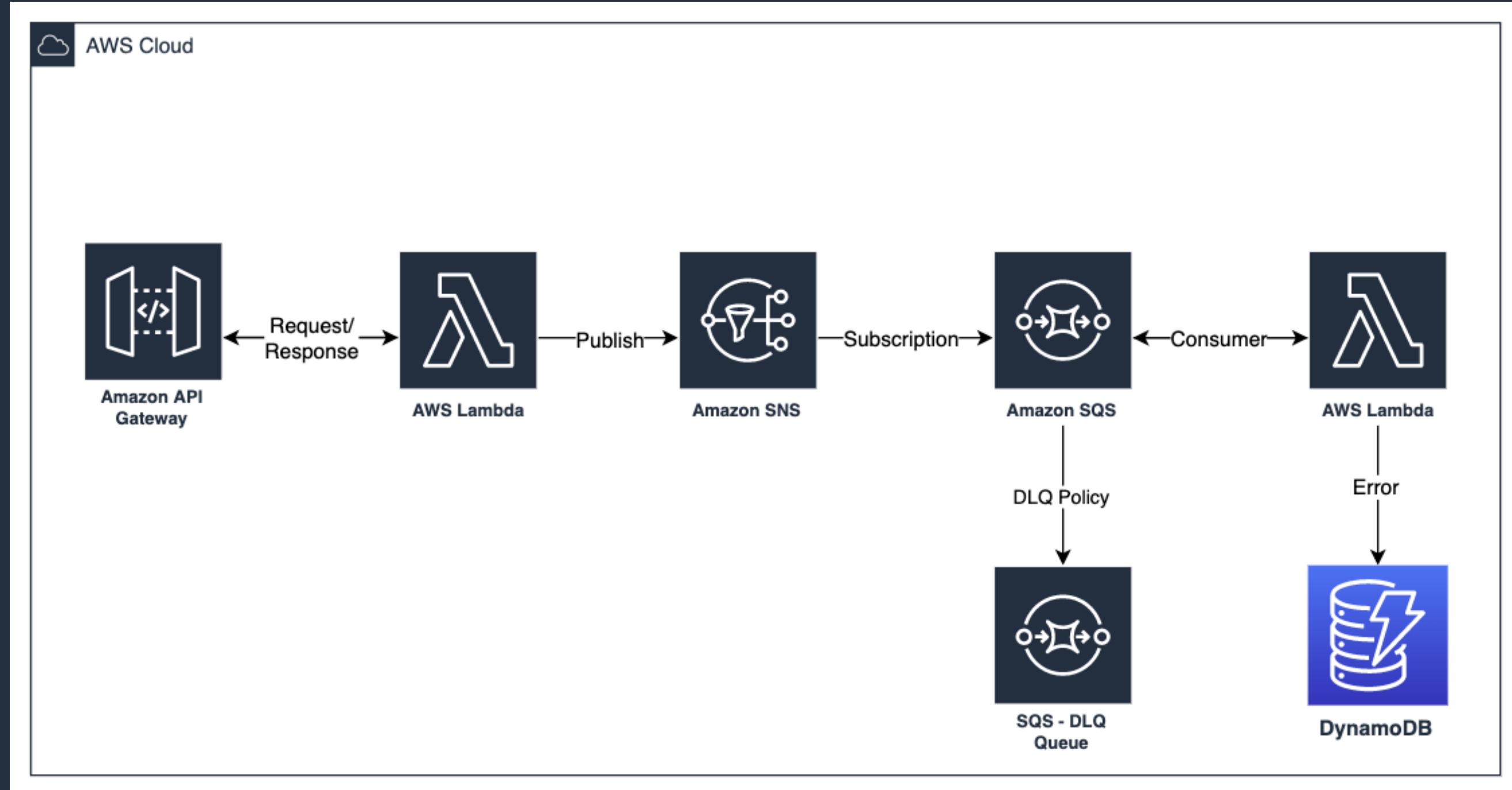
Propriedades + DLQ – Dead-Letter Queue

- Atraso de entrega
- Fila de mensagens morta
- Tempo limite de visibilidade



Amazon Simple Queue Service
(SQS)

Fase 2 – Arquitetura com tratamento de erros



Fase 3

Implementando sincronia de dados com Data Lake

Fase 3 – Implementando sincronia de dados com Data Lake



Amazon DynamoDB Streams

- Ler conteúdo criado/alterado/excluído da tabela
- Processado por lambda
- Próximo de tempo real
Entrega exatamente uma vez

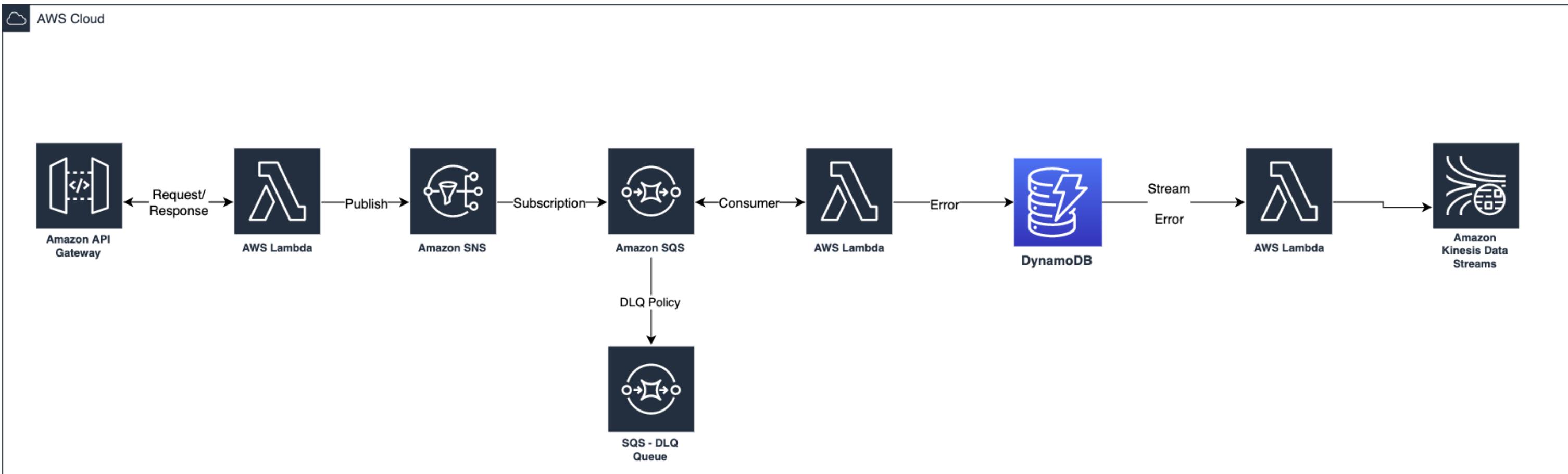
Fase 3 – Implementando sincronia de dados com Data Lake



Amazon Kinesis Stream

- Performance em tempo real
- Multiplos consumidores
- Particionado
- Escalável
- 1 Mb de entrada e 2 Mb de saída por shard provisionado
- Enhanced Fan-Out

Fase 3 – Arquitetura com erros

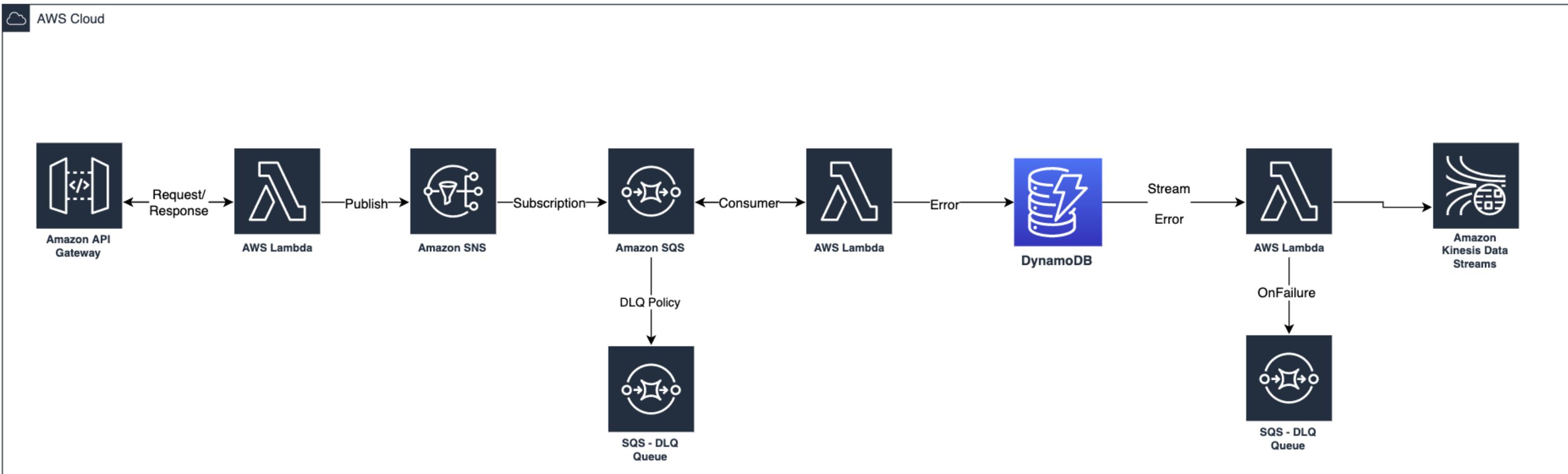


Fase 3 – Lambda Destinations

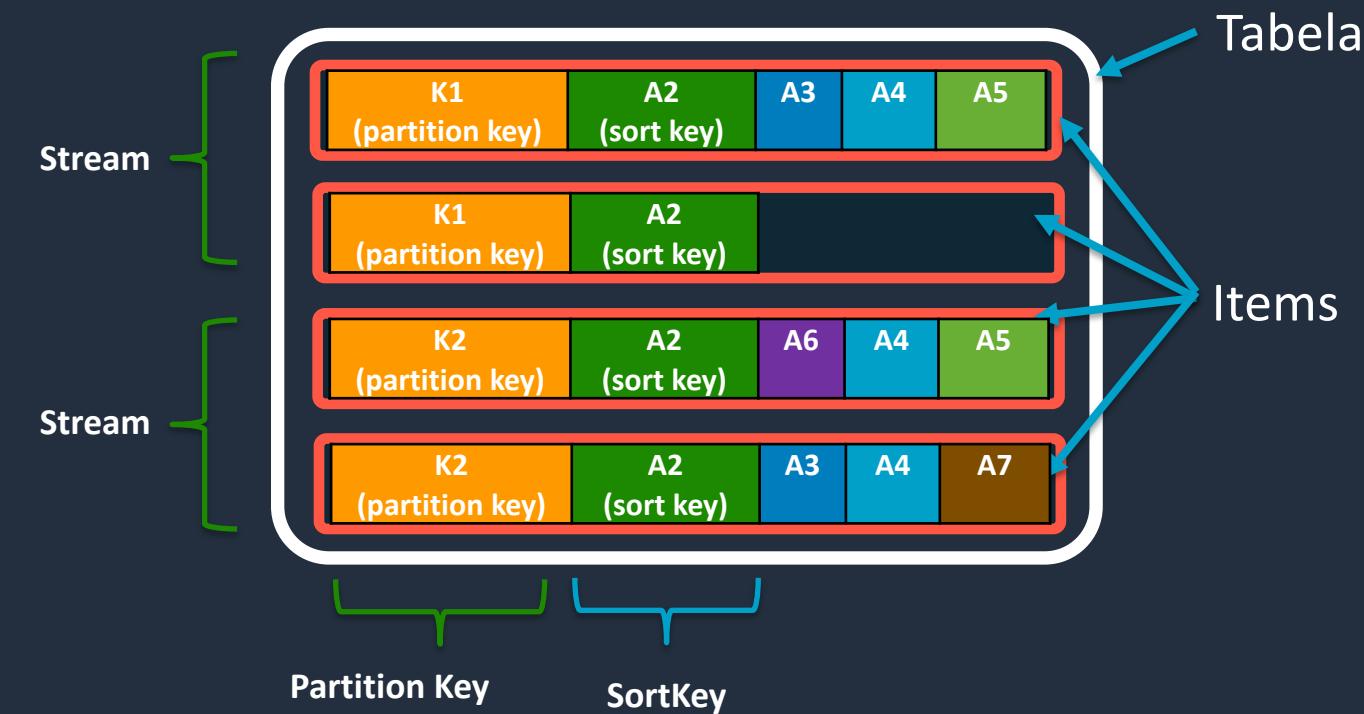


- Em sucesso ou falha do registro após quantidade definida de tentativas pode ser enviada uma notificação para:
 - AWS Lambda
 - Amazon SNS
 - Amazon SQS
 - Amazon EventBridge

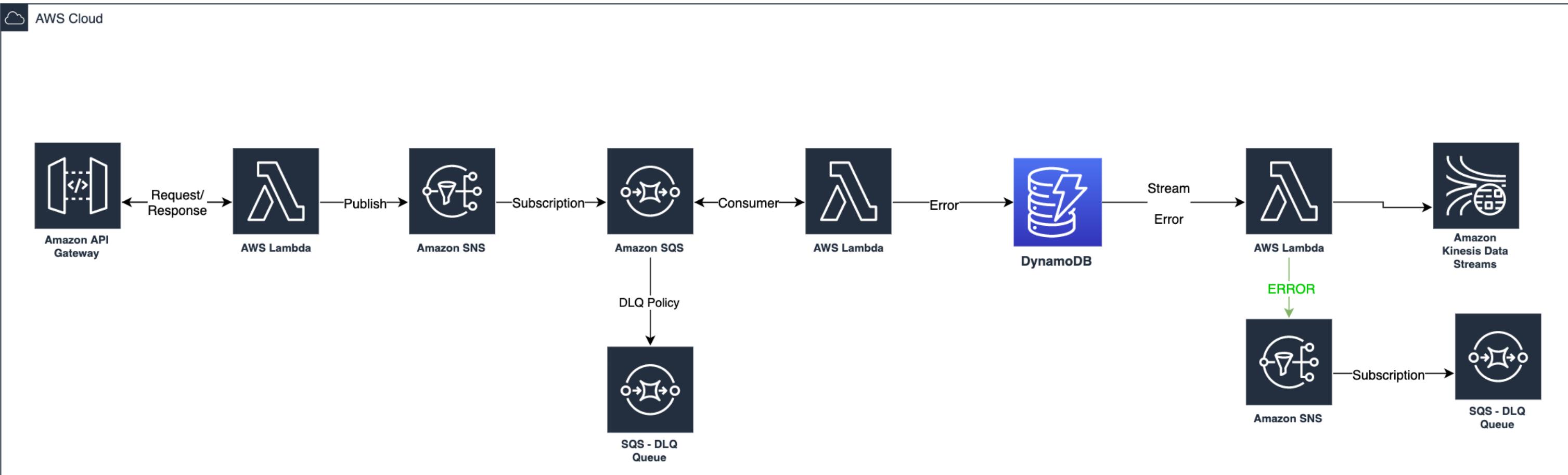
Fase 3 – Arquitetura com tratamento de erros(Opção 1)



Fase 3 – Implementando sincronia de dados com Data Lake



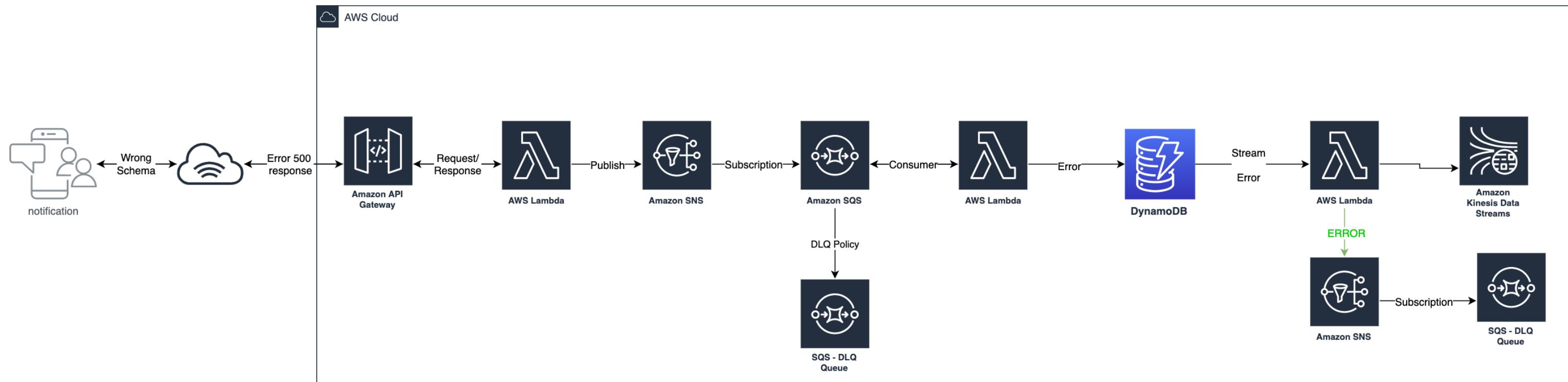
Fase 3 – Arquitetura com tratamento de erros(Opção 2)



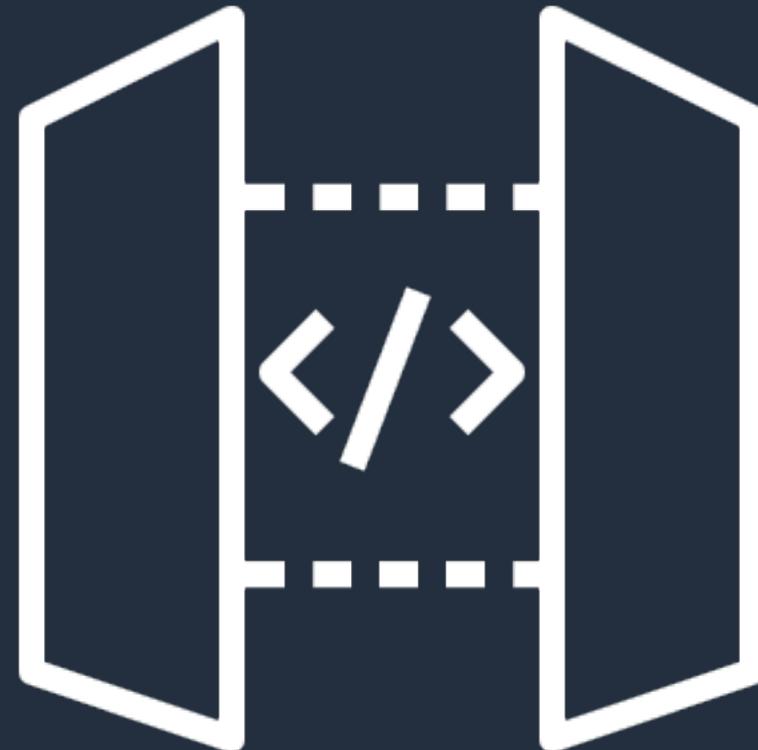
Fase 4

Necessidade de validação de corpo de mensagem

Fase 4 – Arquitetura com erros



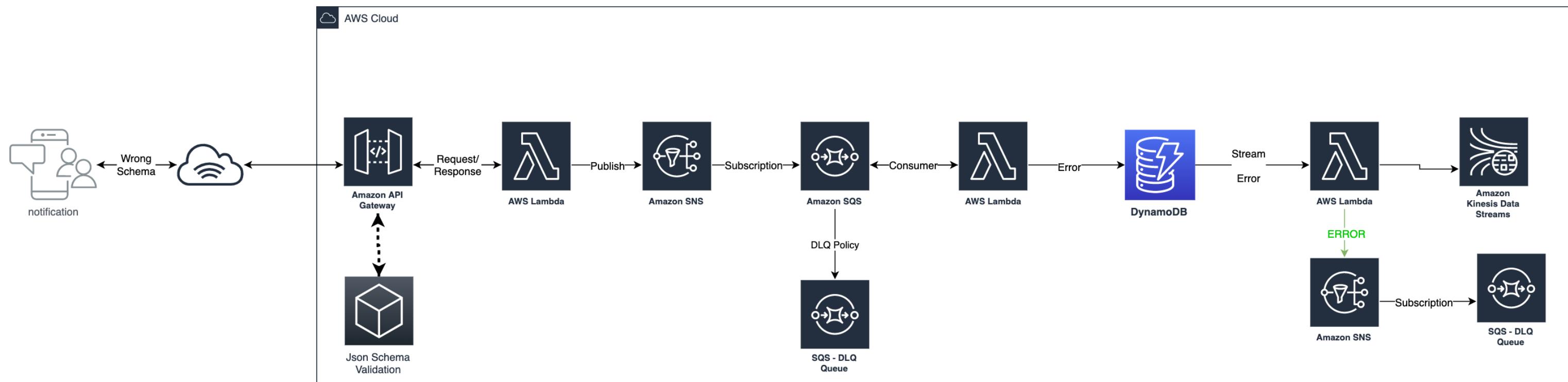
Fase 4 – Checagem de Schema Json



Amazon API gateway

- Validar corpo das mensagens
- Validar query string
- Validar headers
- Mensagem de resposta personalizada

Fase 4 – Arquitetura com tratamento de erros



Resumo



Resumo

- Valide o tipo de integração:
 - Assíncrono
 - Síncrono
- Conheça todas as opções assim como seus benefícios.
- Sempre consulte a documentação
- Invente e simplefique



Perguntas?

Rafael Barbosa

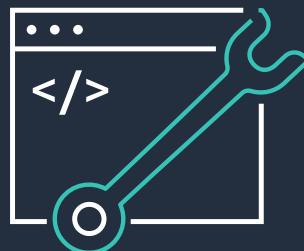
Aprenda serverless no AWS Training and Certification

Recursos criados por especialistas da AWS para ajudá-lo a aprender o desenvolvimento de aplicativos modernos



Cursos gratuitos, sob demanda sobre serverless, incluindo

- Introdução a desenvolvimento Serverless
- Entrando na mentalidade serverless
- Fundamentos de AWS Lambda
- Amazon API Gateway para aplicações serverless
- Amazon DynamoDB para arquitetura serverless



Visite: <https://aws.training>

Obrigado!

Rafael Barbosa