📖Capítulo 7 – DynamoDB

Dia 2 – Building a Streaming Data Pipeline

No segundo dia com o **DynamoDB**, vamos explorar como integrá-lo a outros serviços da AWS para construir um **pipeline de dados em tempo real**. O objetivo é processar e analisar continuamente os eventos que chegam às tabelas.

🧮DynamoDB Streams

O **DynamoDB Streams** captura alterações feitas em uma tabela (inserções, atualizações e exclusões) em tempo real.

- Cada evento contém:
- Nome da tabela.
- Tipo da operação (INSERT, MODIFY, REMOVE).
- Imagem antes e depois do item (se configurado).
- Retém eventos por até **24 horas**.

📌Exemplo de ativação via CLI:

```
aws dynamodb update-table
  --table-name Pedidos
  --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

🕐Integração com AWS Lambda

- O **AWS Lambda** pode ser configurado como consumidor do DynamoDB Streams.
- Cada alteração na tabela dispara uma função Lambda.
- Ideal para:
- Atualizar índices secundários.
- Notificar outros sistemas.
- Processar dados em tempo real (ex.: analytics, detecção de fraude).

📌Exemplo simples de função Lambda (Node.js):

```
exports.handler = async (event) => {
  event.Records.forEach(record => {
    console.log('Evento:', record.eventName);
    console.log('Novo item:', JSON.stringify(record.dynamodb.NewImage));
  });
};
```

📊Integração com Kinesis e Analytics

O DynamoDB Streams pode enviar dados para o **Kinesis Data Streams** ou **Kinesis Firehose**, que então alimentam: - **S3** → armazenamento. - **Redshift** → análise com SQL. - **Elasticsearch/OpenSearch** → busca avançada. - **Kinesis Analytics** → consultas em tempo real com SQL.

Exemplo de arquitetura:

```
DynamoDB → Streams → Lambda → Kinesis Firehose → S3/Redshift/ElasticSearch
```

🛒Caso de Uso: E-commerce em Tempo Real

Imagine um e-commerce com milhões de pedidos por dia. - Cada novo pedido é gravado na tabela `Pedidos`. - O DynamoDB Streams captura o evento. - Uma função Lambda processa e envia para: - **Kinesis** (para dashboards em tempo real). - **S3** (para armazenamento histórico). - **SNS/SQS** (para notificação e fila de processamento).

📌Resultado: acompanhamento em tempo real de vendas, sem sobrecarregar a tabela principal.

⚖️Vantagens

- Processamento assíncrono e escalável.
- Baixa latência.
- Fácil integração com outros serviços da AWS.
- Permite separar a carga de leitura/escrita da carga analítica.

🔗Resumo do Dia 2

- Ativamos e entendemos o **DynamoDB Streams**.
- Conectamos Streams a funções **Lambda**.
- Exploramos a integração com **Kinesis, S3, Redshift e Elasticsearch**.
- Criamos um pipeline para processar pedidos em tempo real em um sistema de e-commerce.

🔔No próximo passo: **Dia 3 – Building an Internet of Things (IoT) System Around DynamoDB**, onde veremos como o DynamoDB pode ser usado como base para arquiteturas de IoT.