

# Serverless Search



## 이상현 / Kurt Lee

- CTO, Catchfashion Inc
- Former Technical Leader, Vingle Inc
- AWS Serverless Hero

iOS / Frontend / Backend / Serverless /  
Data Pipeline / API / Microservices

[breath103@gmail.com](mailto:breath103@gmail.com)












<https://github.com/breath103>

<https://medium.com/@kurtlee>

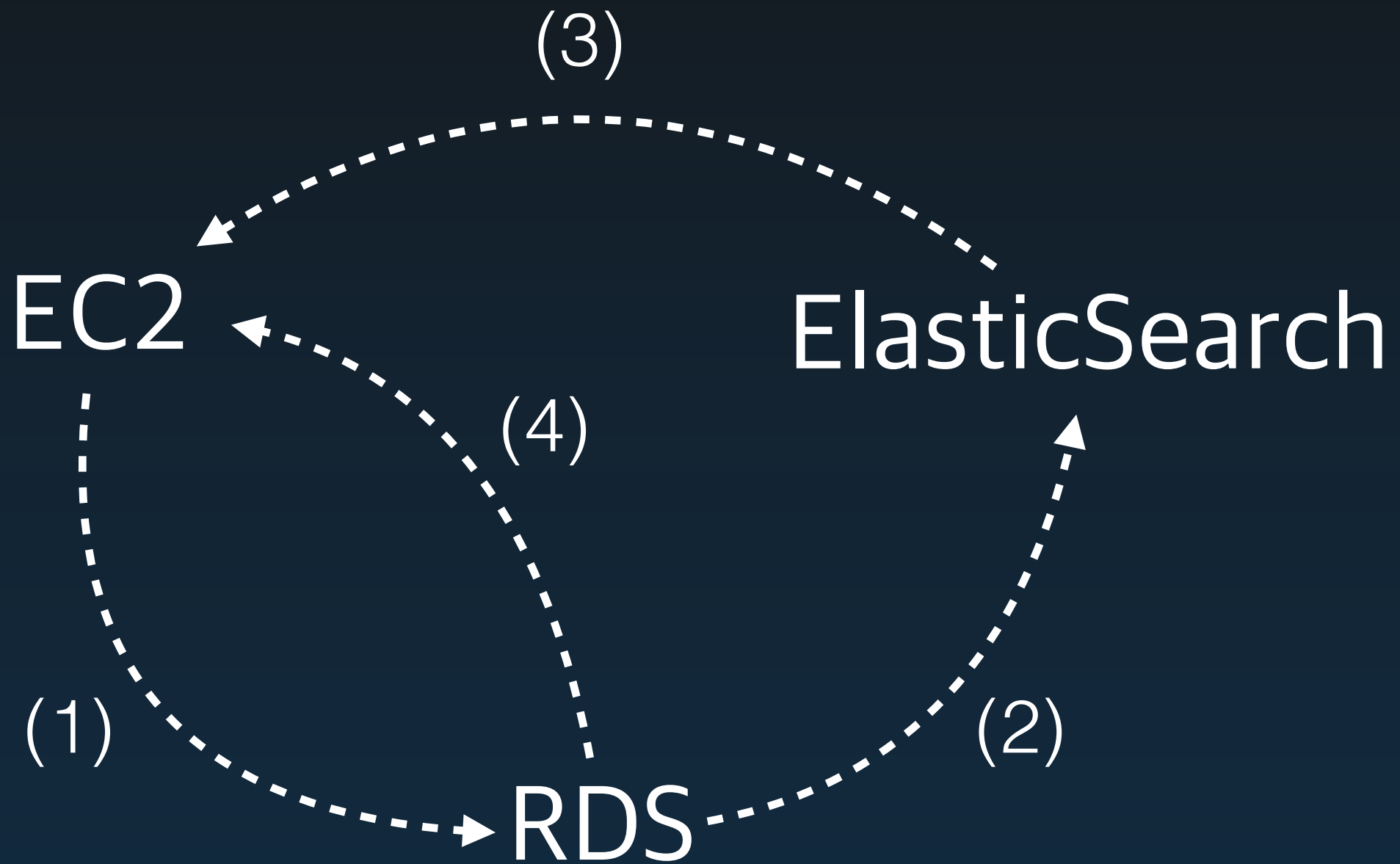
# Q. 왜 Elasticsearch를 써야하는지?

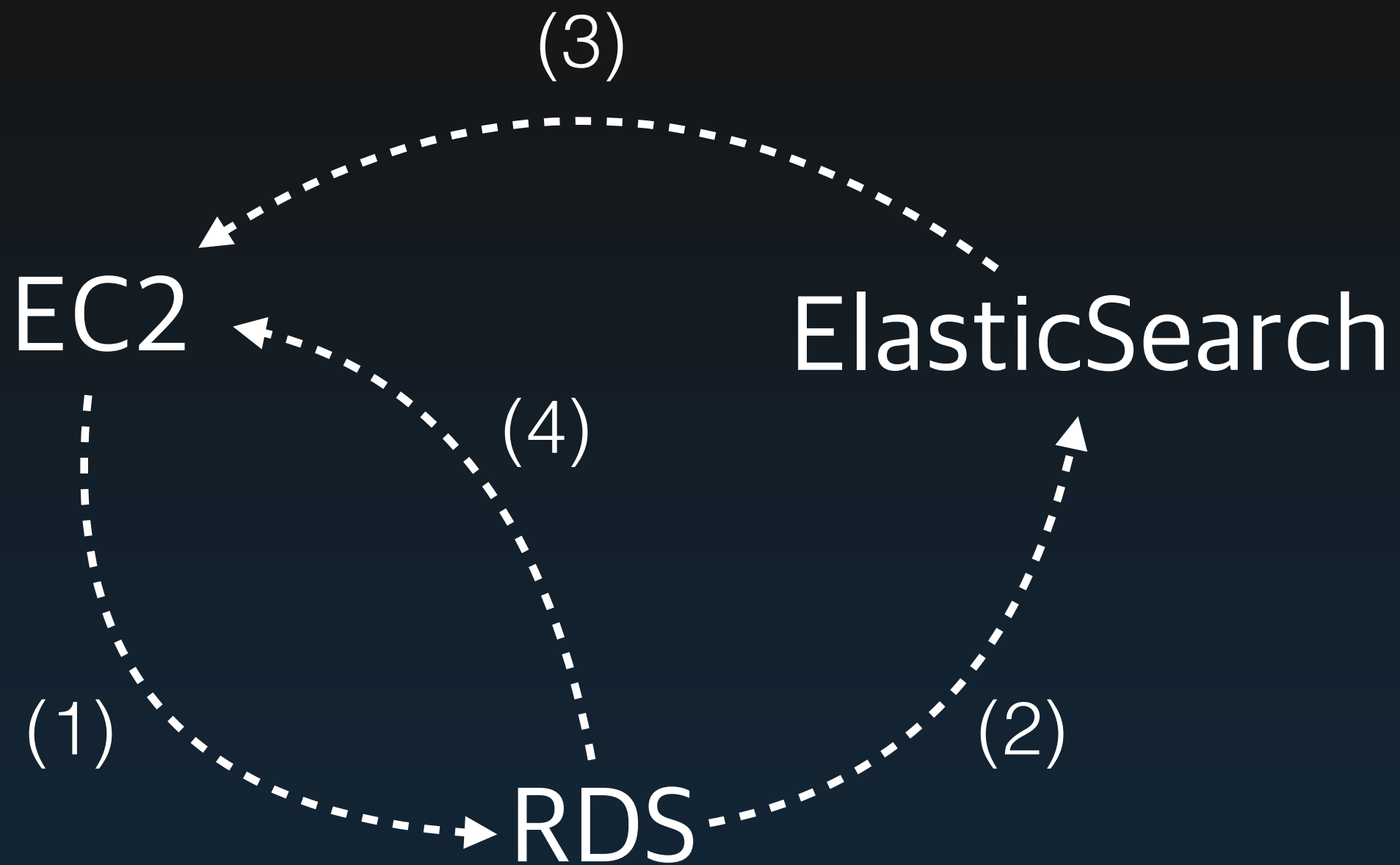
## A. 일반적인 RDS는 검색을 위한 Query에 적합하지 않습니다.

### Database services

Database type	Use cases	AWS service
Relational	Traditional applications, ERP, CRM, e-commerce	 <a href="#">Amazon Aurora</a>  <a href="#">Amazon RDS</a>  <a href="#">Amazon Redshift</a>
Key-value	High-traffic web apps, e-commerce systems, gaming applications	 <a href="#">Amazon DynamoDB</a>
In-memory	Caching, session management, gaming leaderboards, geospatial applications	 <a href="#">Amazon ElastiCache for Memcached</a>  <a href="#">Amazon ElastiCache for Redis</a>
Document	Content management, catalogs, user profiles	 <a href="#">Amazon DocumentDB</a>
Wide column	High scale industrial apps for equipment maintenance, fleet management, and route optimization	 <a href="#">Amazon Managed Apache Cassandra Service</a>
Graph	Fraud detection, social networking, recommendation engines	 <a href="#">Amazon Neptune</a>
Time series	IoT applications, DevOps, industrial telemetry	 <a href="#">Amazon Timestream</a>
Ledger	Systems of record, supply chain, registrations, banking transactions	 <a href="#">Amazon QLDB</a>

# 일반적인 검색 아키텍처





1. App은 RDS에 데이터를 write
2. RDS는 데이터를 ElasticSearch에 Sync
  1. 이때 `updated_at > (now() - 10 minutes)` 로 Cron job을 돌린다던가...
3. App은 검색 요청이 들어오면 ElasticSearch에서 검색 결과물로, 원하는 테이블의 id를 받음
4. App은 (3)에서 받은 테이블의 id로 RDS 에서 다시 record를 검색

# 문제점

1. App은 RDS에 데이터를 write
2. RDS는 데이터를 ElasticSearch에 Sync
  1. 이때 `updated_at > (now() - 10 minutes)` 로 Cron job을 돌린다던가...
    1. 오직 이 작업을 위해 `updated_at` 에 index를 만들어야..
    2. cron주기가 10분이다 -> MD들이 상품정보 바꾸면 10분뒤에 반영?;
    3. distributed system architecture가 아님.
      1. 평상시엔 10분마다 변경되는 상품이 100개다가, 어느날 갑자기 1만개가 변경되면?  
Cron process가 memory 터지거나, failover가 일어나야함.  
심지어 failover를 해도, 그냥 같은 timewindow로 재시도 하면 똑같이 메모리 폭발
3. App은 검색 요청이 들어오면 ElasticSearch에서 검색 결과물로, 원하는 테이블의 id를 받음
4. App은 (3)에서 받은 테이블의 id로 RDS에서 다시 record를 검색

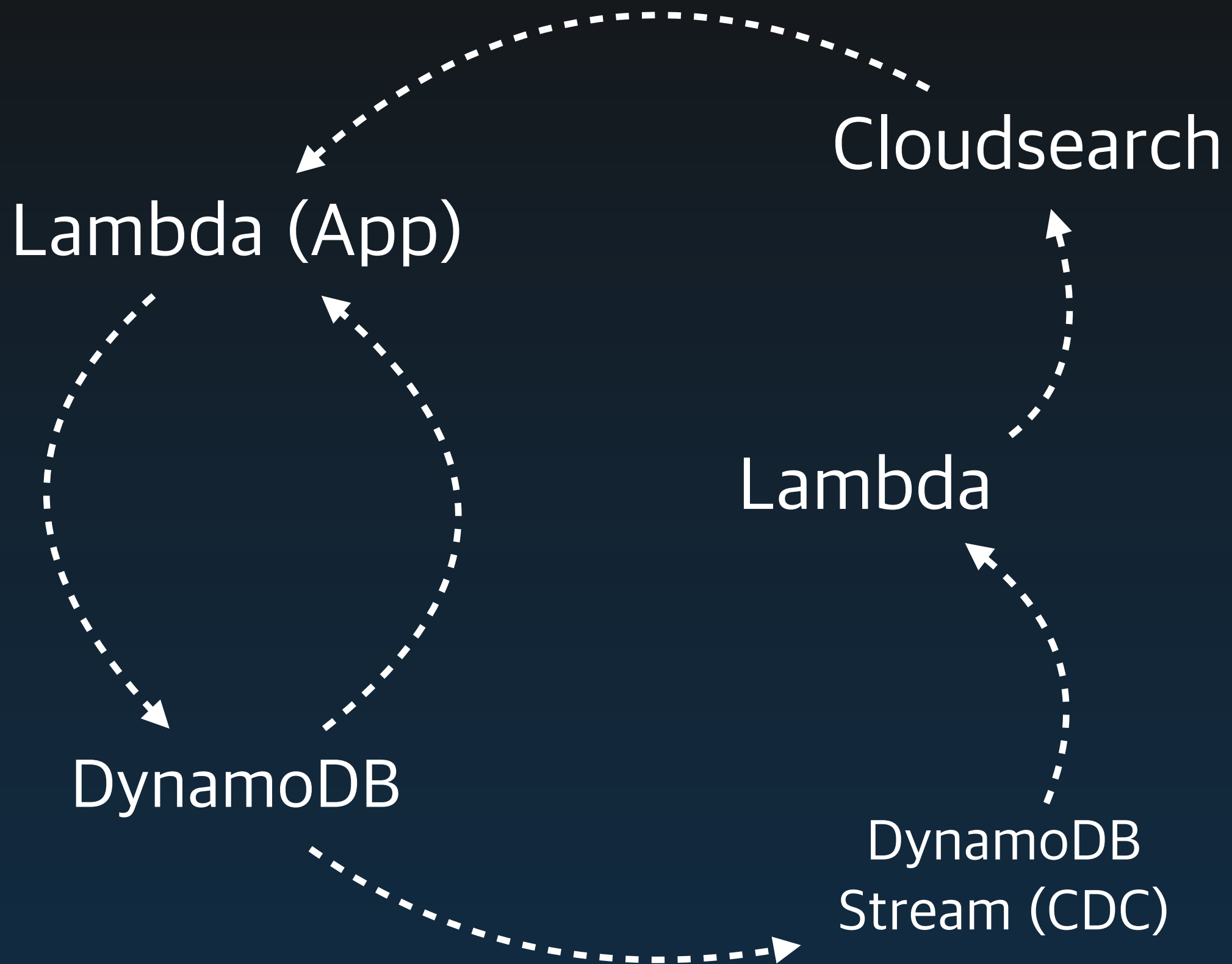
Better Architecture?

- Serverless
- Distributed System
- Scalable
- Failover & Retry



1.

100% Pure Serverless  
Solution



# Amazon Cloudsearch?

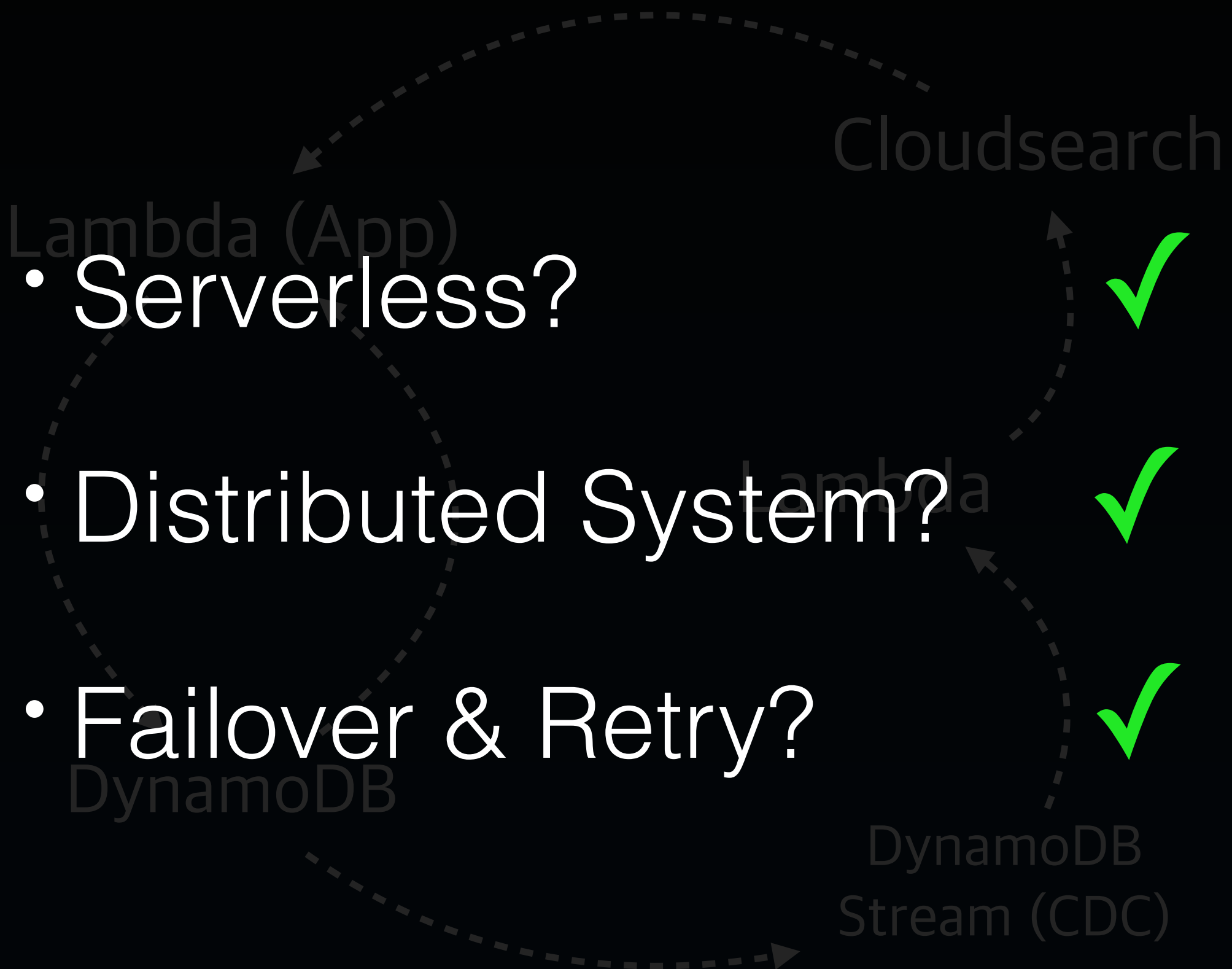
Solr 기반 Fully Managed Search Service

# DynamoDB Stream

Fully managed, Change Data Capture + Stream Service  
= DynamoDB 변경사항을 읽어서 Kinesis Stream에 넣어준다.

자세한 가이드는

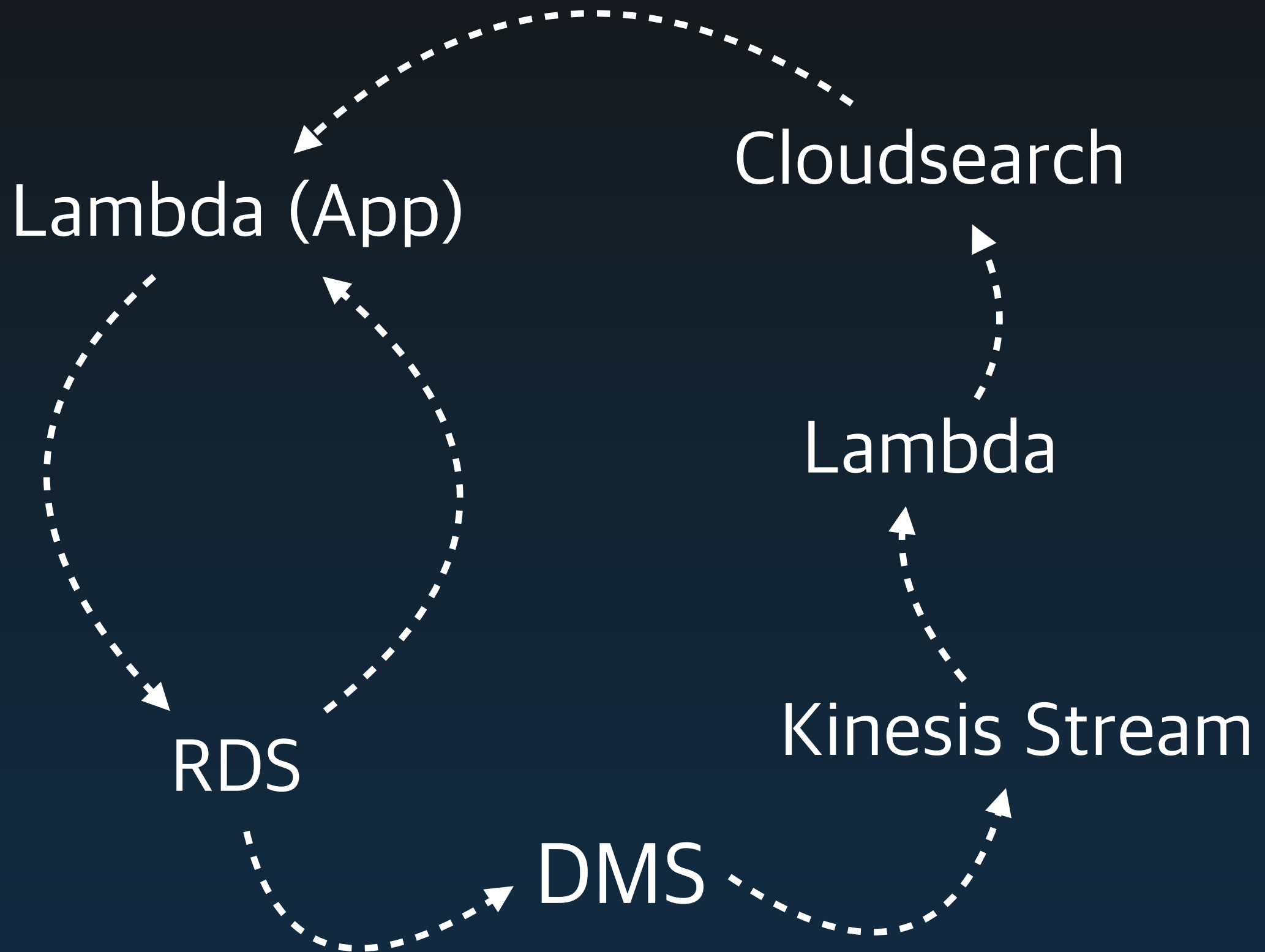
<https://medium.com/vingle-tech-blog/do-you-know-dynamodb-stream-14b284bf38d5>



하지만 DynamoDB로는  
X가 안되어 썩썩 ㅠㅠ

2.

# 반쯤 Serverless Solution





# Amazon Data Migration Service

각종 DB → DB CDC를 이용한 Migration을 해주는 서비스.

## ☒ Source endpoint

A source endpoint allows AWS DMS to read data from a database (on-premises or in the cloud), or from other data source such as Amazon S3.

aurora

aurora-serverless

aurora-postgresql

docdb

dynamodb

kinesis

redshift

s3

elasticsearch

mariadb

sqlserver

mysql

## ☐ Target endpoint

A target endpoint allows AWS DMS to write data to a database, or to other data source.

aurora

aurora-postgresql

s3

db2

mariadb

azuredb

sqlserver

mongodb

mysql

oracle

postgres

sybase

"Replicator Instance"를 Provisioning 하긴 해야합니다.  
하지만 fully managed.

# Kinesis Stream -> Lambda

Fully managed data stream processing

• Serverless?

• Distributed System?

• Failover & Retry?

