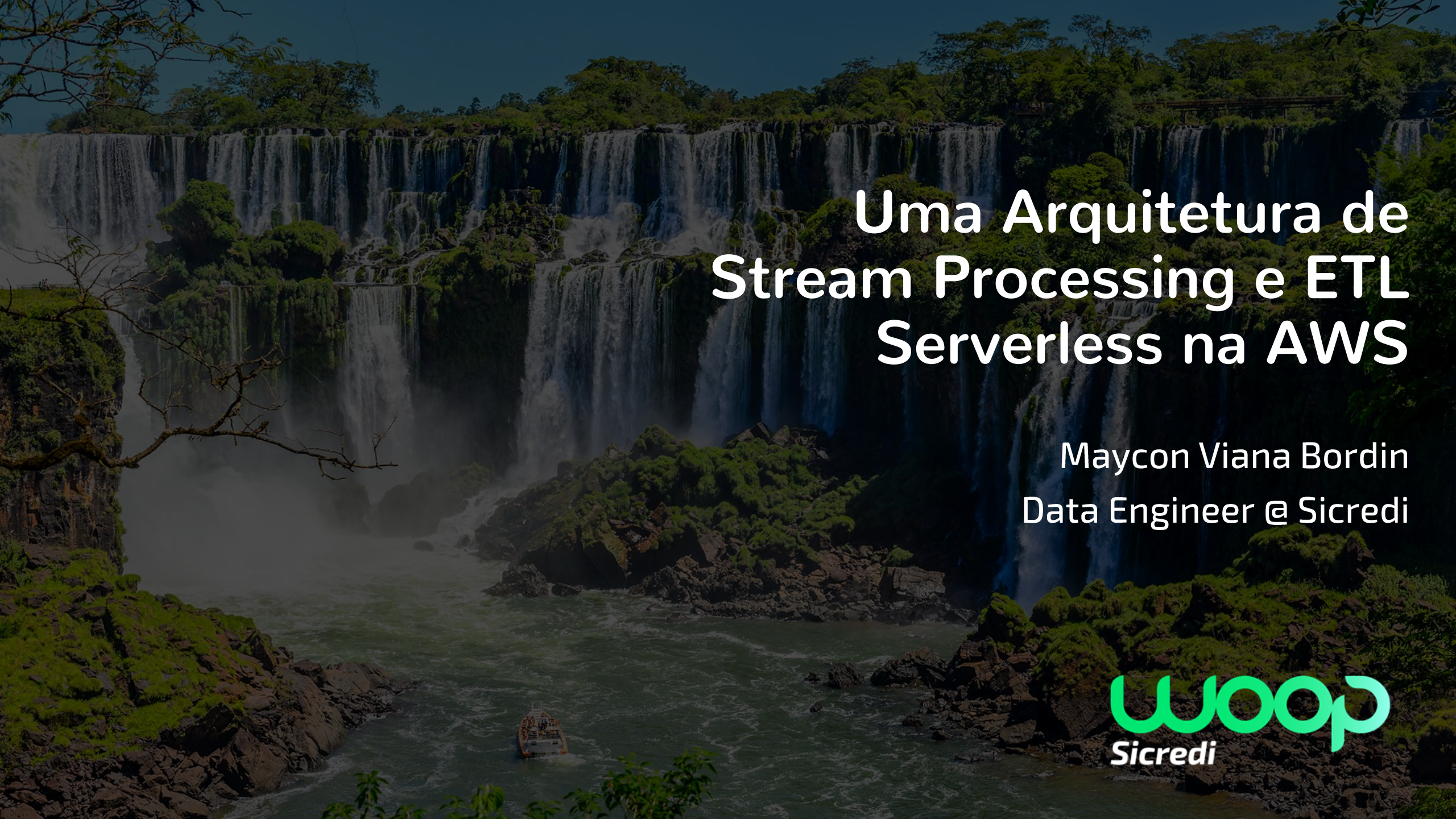


Uma Arquitetura de Stream Processing e ETL Serverless na AWS

Maycon Viana Bordin

Data Engineer @ Sicredi

Woop Sicredi

Sicredi

Uma instituição financeira diferente.

Uma instituição financeira cooperativa.

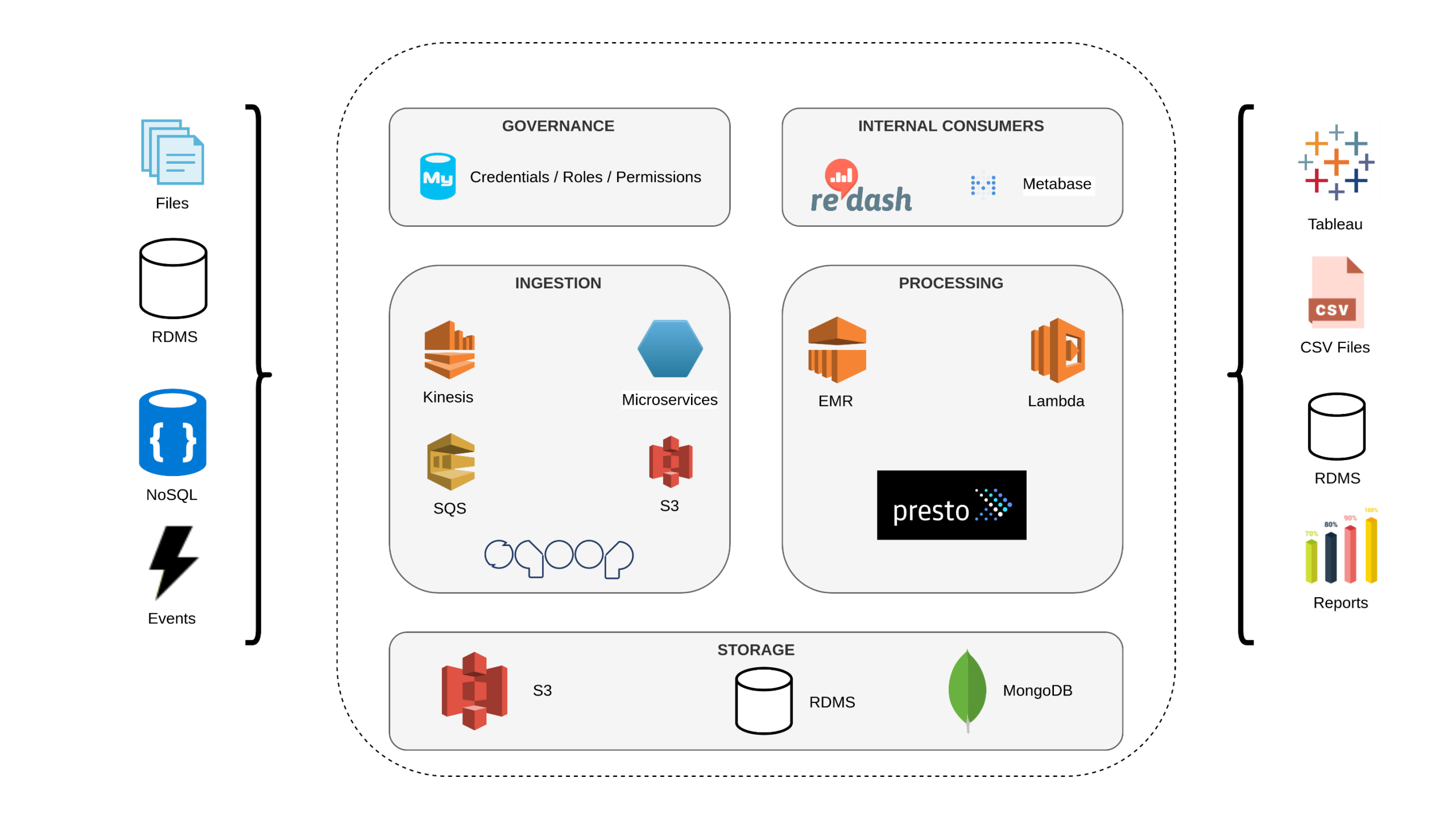
TRANSFORMAÇÃO DIGITAL

Novo Core Bancário

Plataforma para inovação

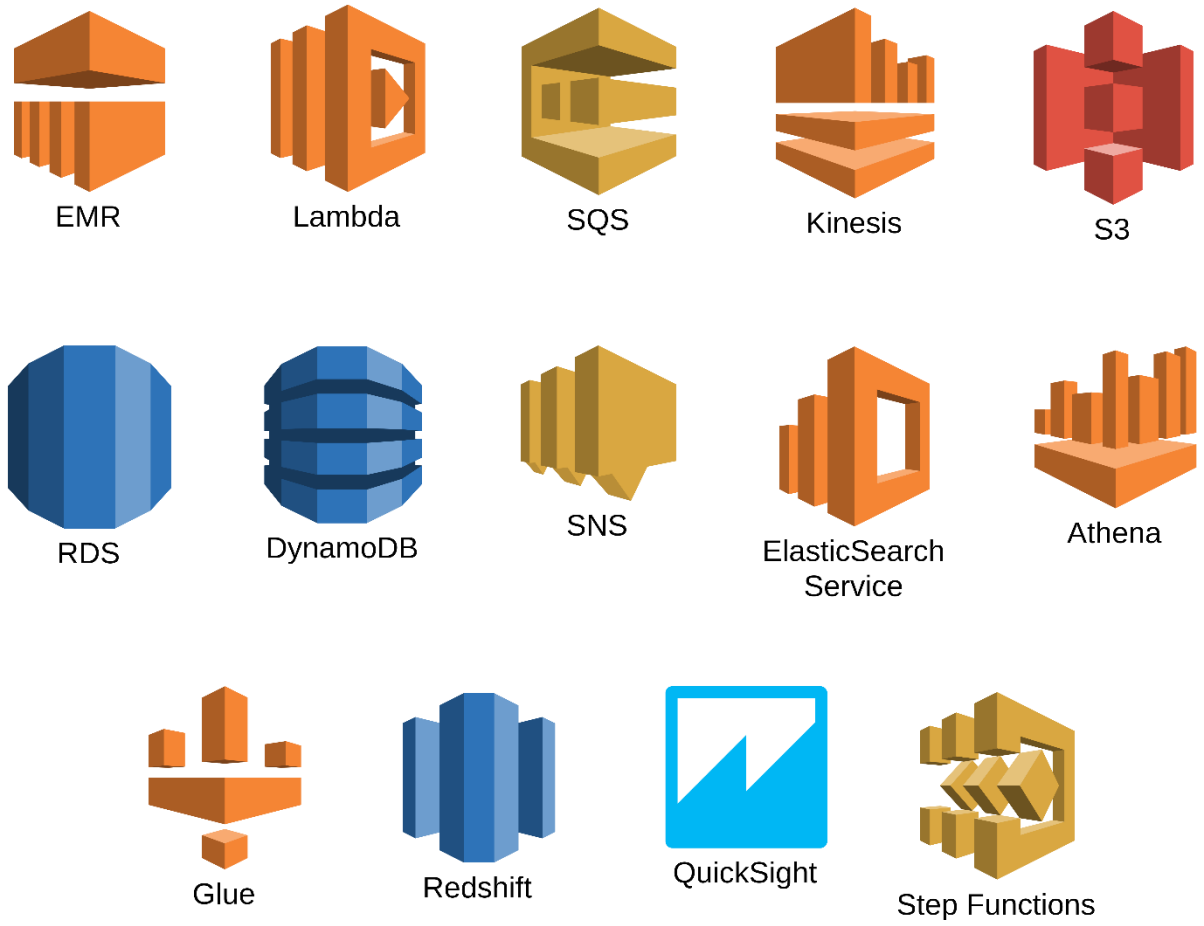
Experiência digital

DATA LAKE do WOOP


Files
RDMS
NoSQL
Events
GOVERNANCE
Credentials / Roles / Permissions
INTERNAL CONSUMERS
re dash
Metabase
INGESTION
Kinesis
Microservices
SQS
S3
PROCESSING
EMR
Lambda
presto
STORAGE
S3
RDMS
MongoDB
Tableau
CSV
CSV Files
RDMS
Reports

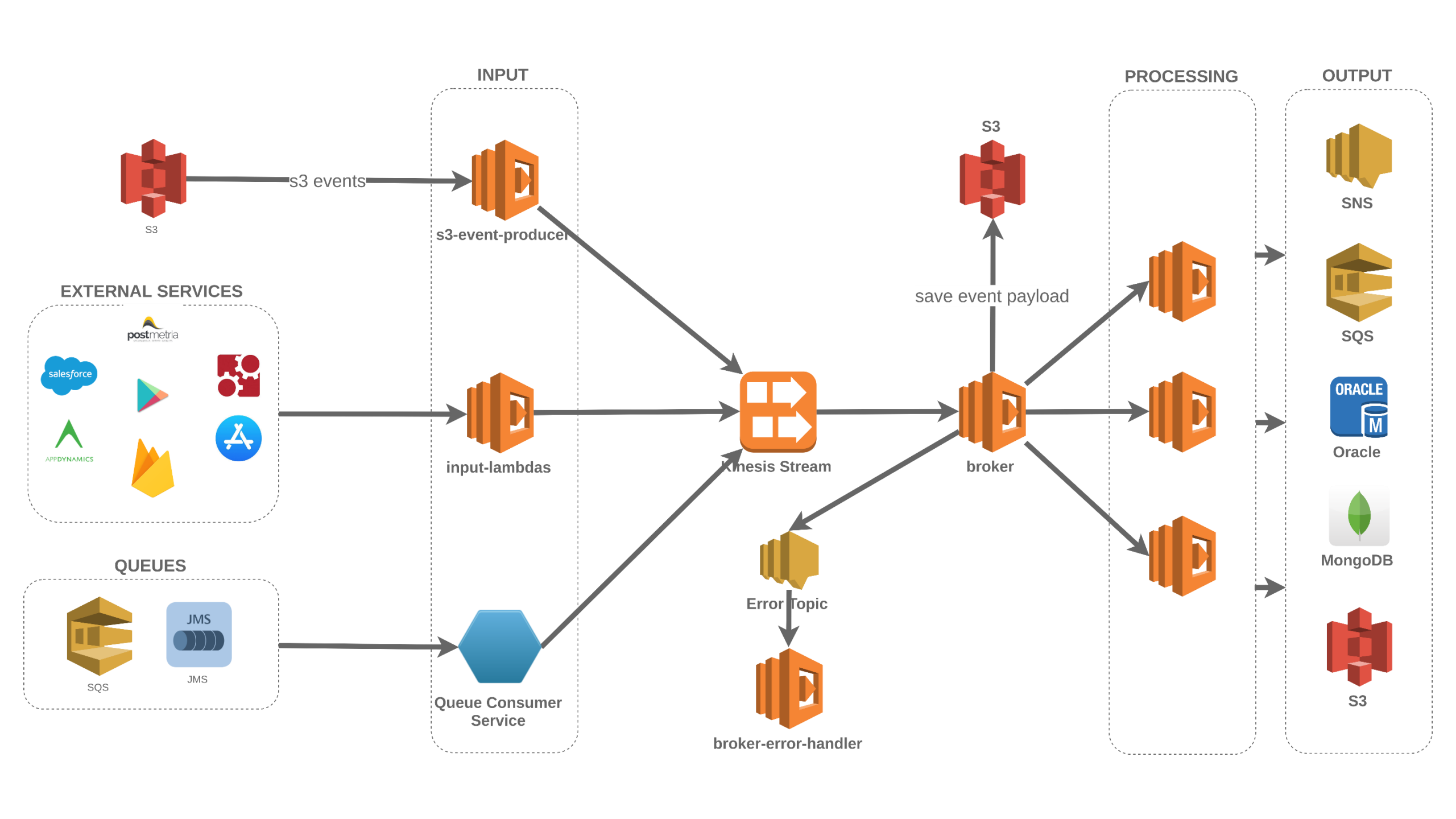
Serverless

Serverless na aws



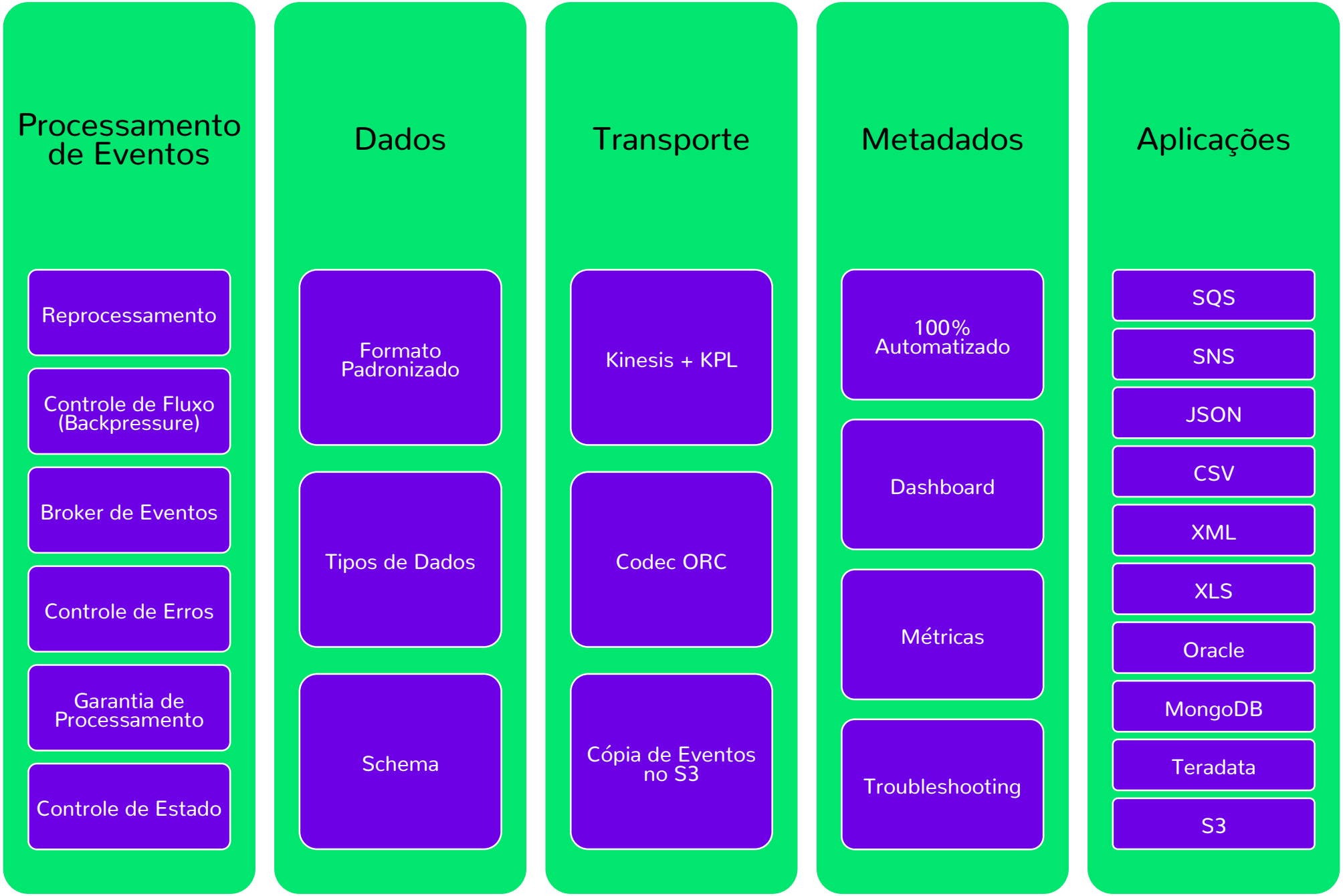
Por que escolhemos Serverless?

A Arquitetura


INPUT
PROCESSING
OUTPUT
S3
s3 events
s3-event-producer
S3
save event payload
SNS
SQS
Oracle
MongoDB
S3
EXTERNAL SERVICES
postmetria
salesforce
APPDYNAMICS
input-lambdas
Kinesis Stream
broker
QUEUES
SQS
JMS
JMS
Queue Consumer Service
Error Topic
broker-error-handler

e tudo escrito em


O Framework
dentro do Framework

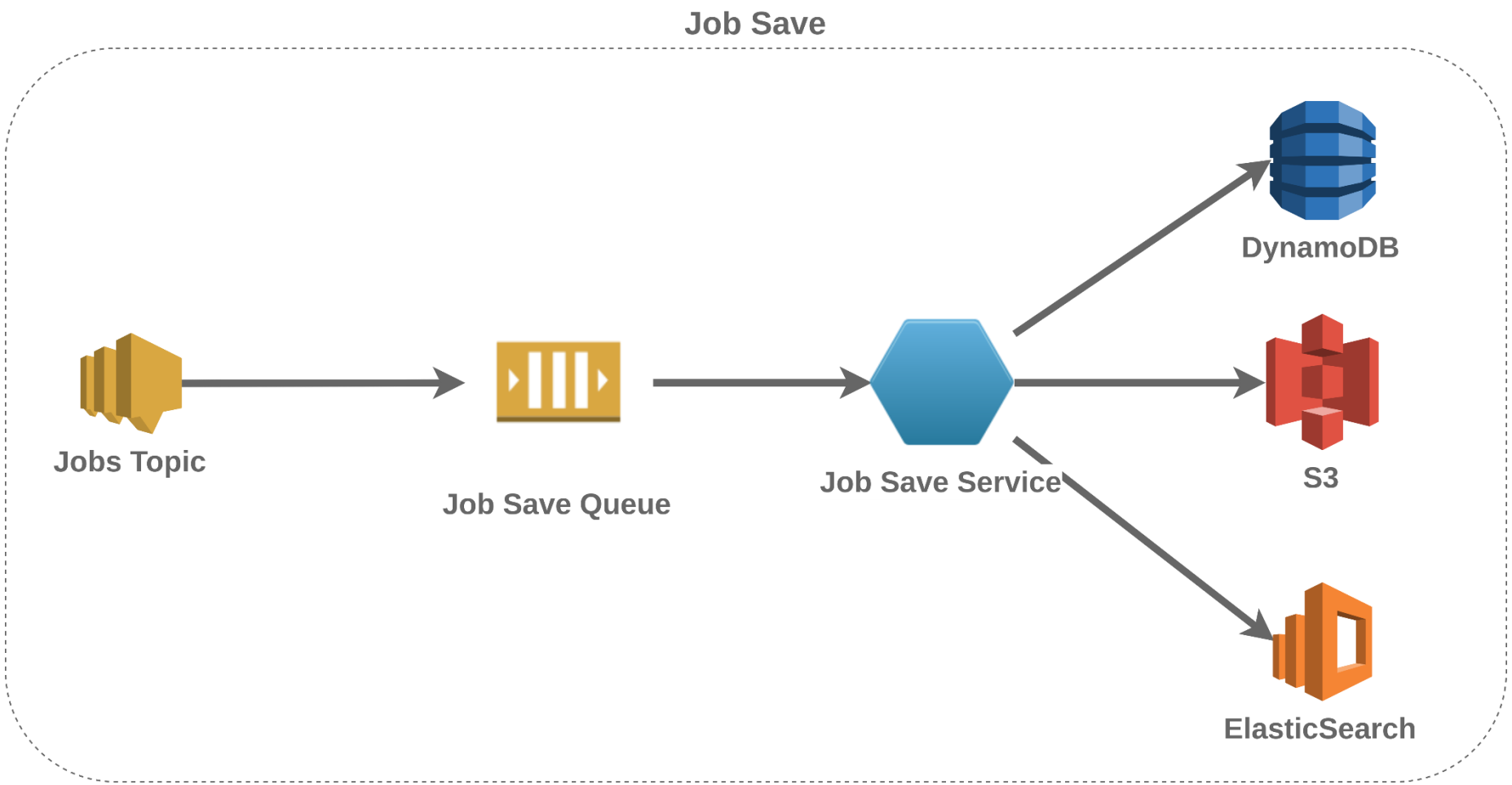

Processamento de Eventos
Reprocessamento
Controle de Fluxo (Backpressure)
Broker de Eventos
Controle de Erros
Garantia de Processamento
Controle de Estado
Dados
Formato Padronizado
Tipos de Dados
Schema
Transporte
Kinesis + KPL
Codec ORC
Cópia de Eventos no S3
Metadados
100% Automatizado
Dashboard
Métricas
Troubleshooting
Aplicações
SQS
SNS
JSON
CSV
XML
XLS
Oracle
MongoDB
Teradata
S3

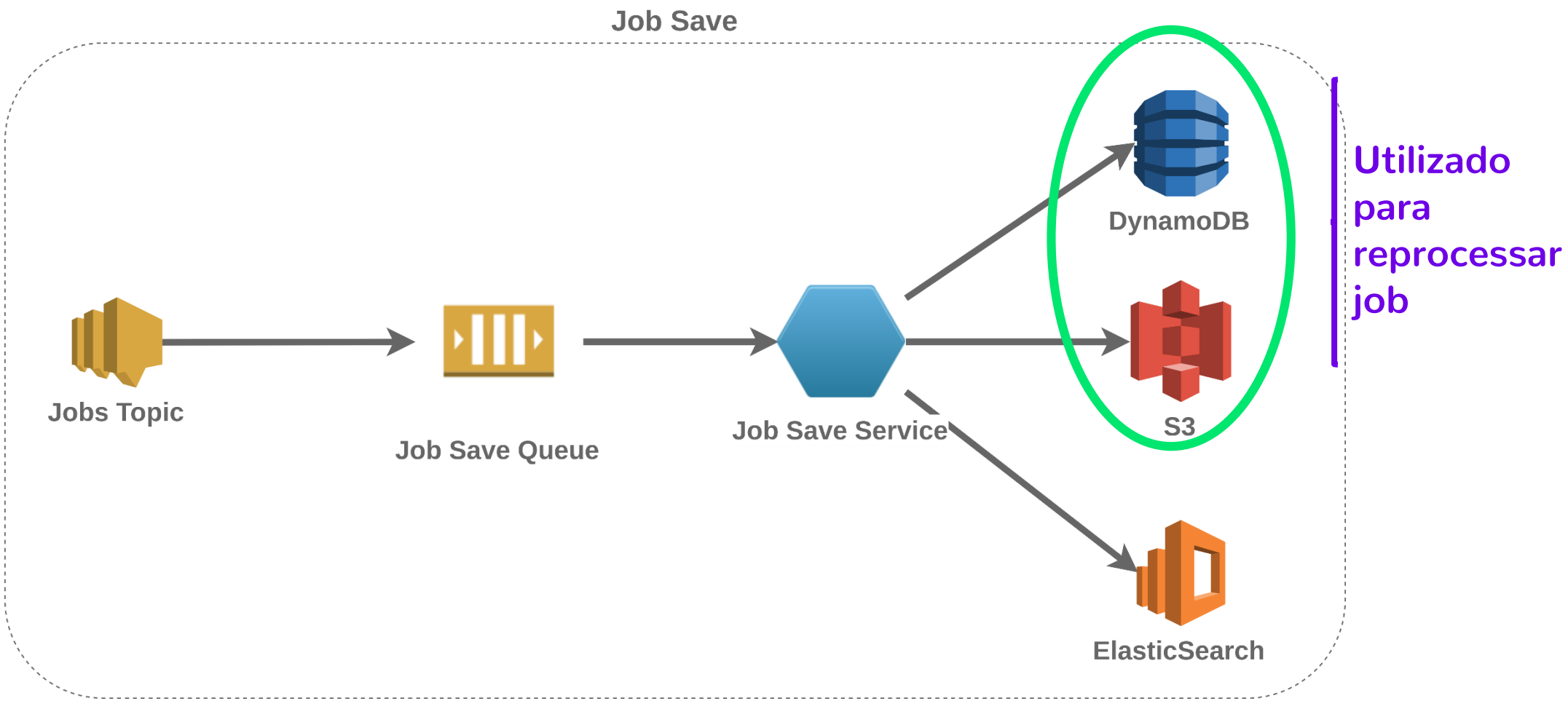
JsonMapProcessor.scala

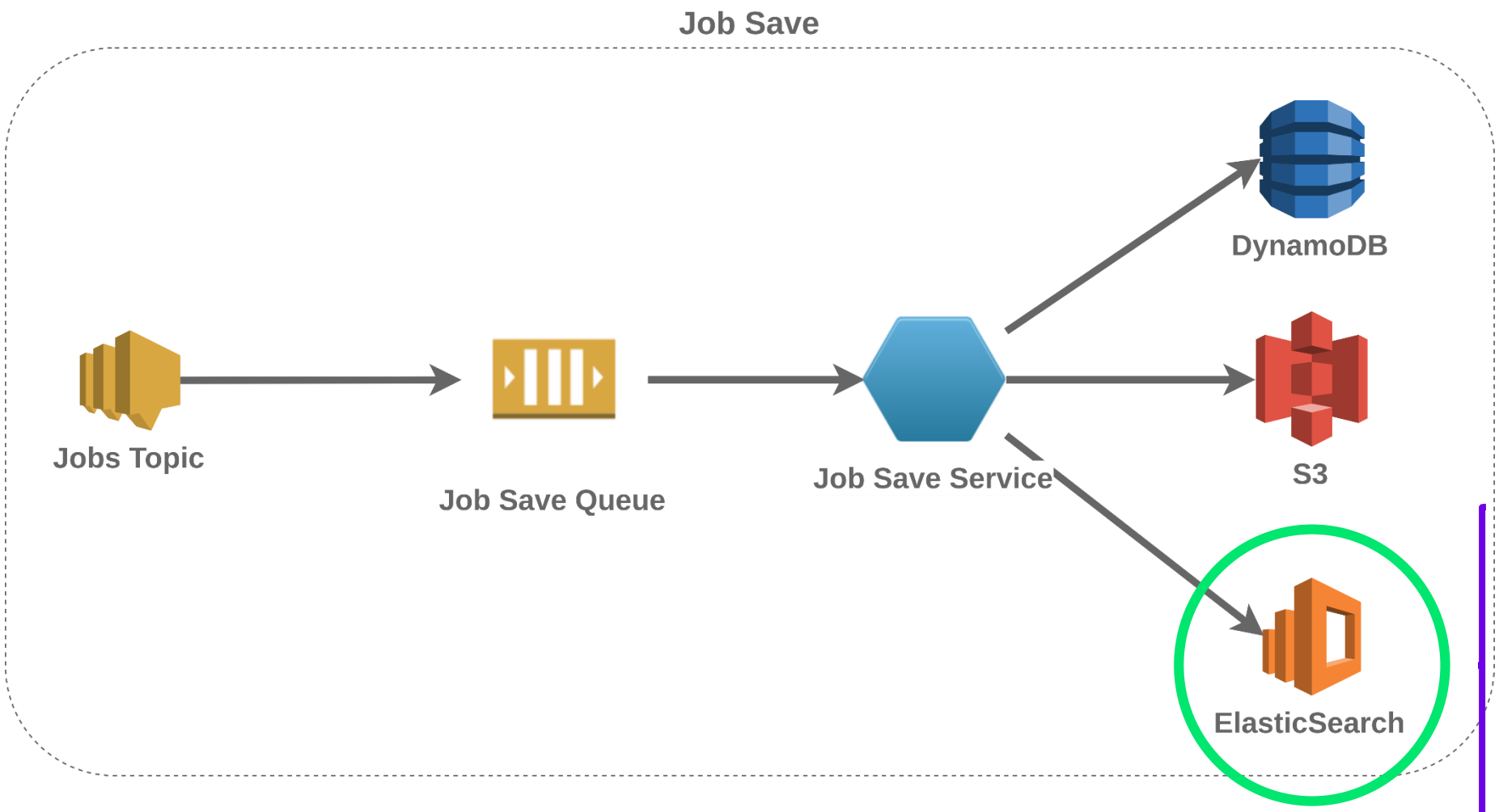
Raw

```scala
class JsonMapProcessor extends BaseProcessor {
  lazy val fields: Map[String, List[String]]  = config.getStringListMap("fields")
  lazy val jsonField: String                  = config.getString("json_field").getOrElse("data")
  lazy val outputStream: String               = config.getOutputStream

  override def process(input: EventRecord): Unit = {
    val rawData = input.getString(jsonField).get

    if (rawData == null || rawData.trim.isEmpty) {
      log.info("Record is empty, ignoring it.")
      return
    }

    val record = fields.map(field => {
      field._1 -> JsonPathUtil.readToStringMultiple(field._2, rawData)
    })

    emit(outputStream, record)
  }
}
```

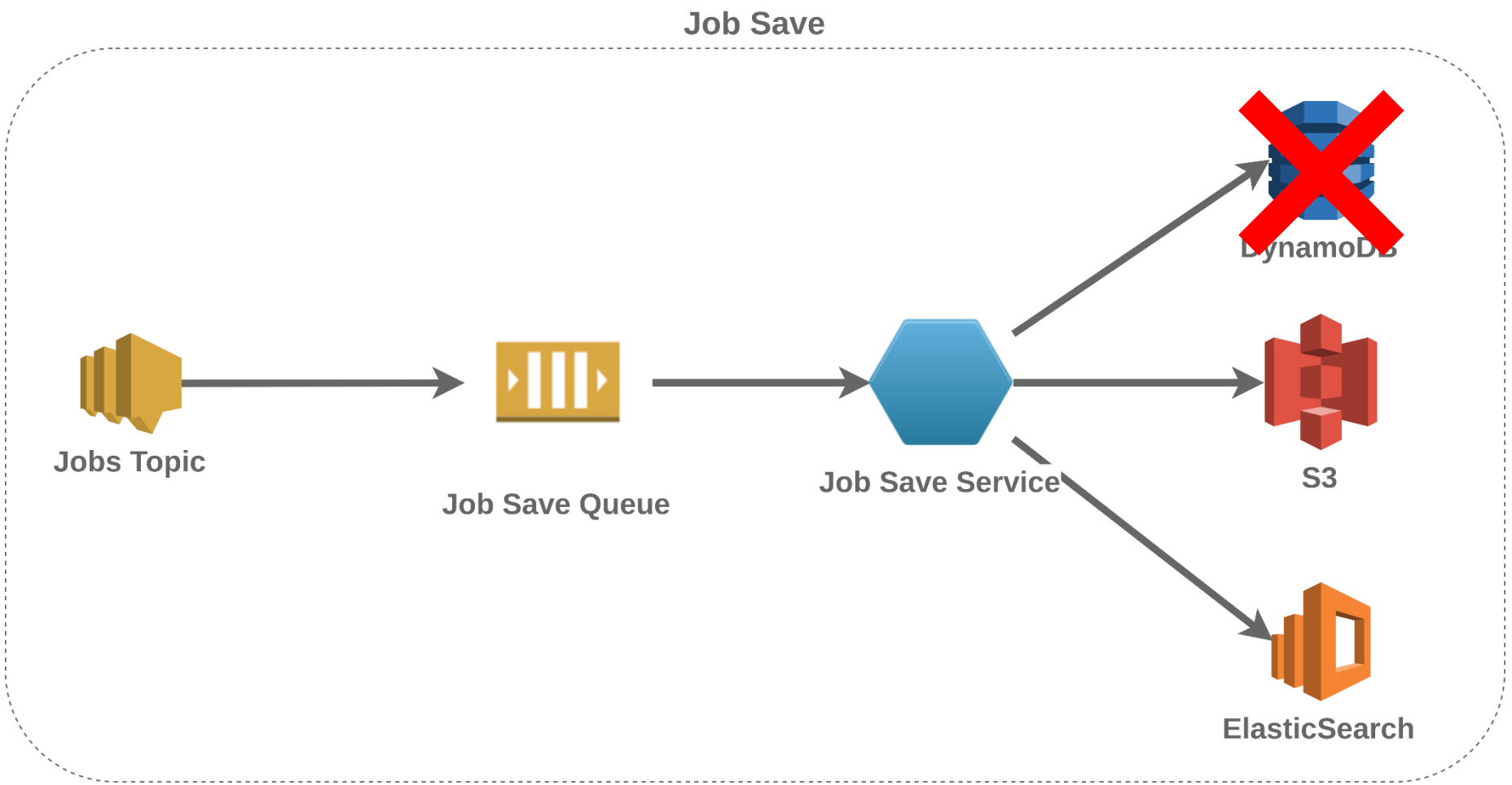
Metadados


Job Save
DynamoDB
Jobs Topic
Job Save Queue
Job Save Service
S3
ElasticSearch

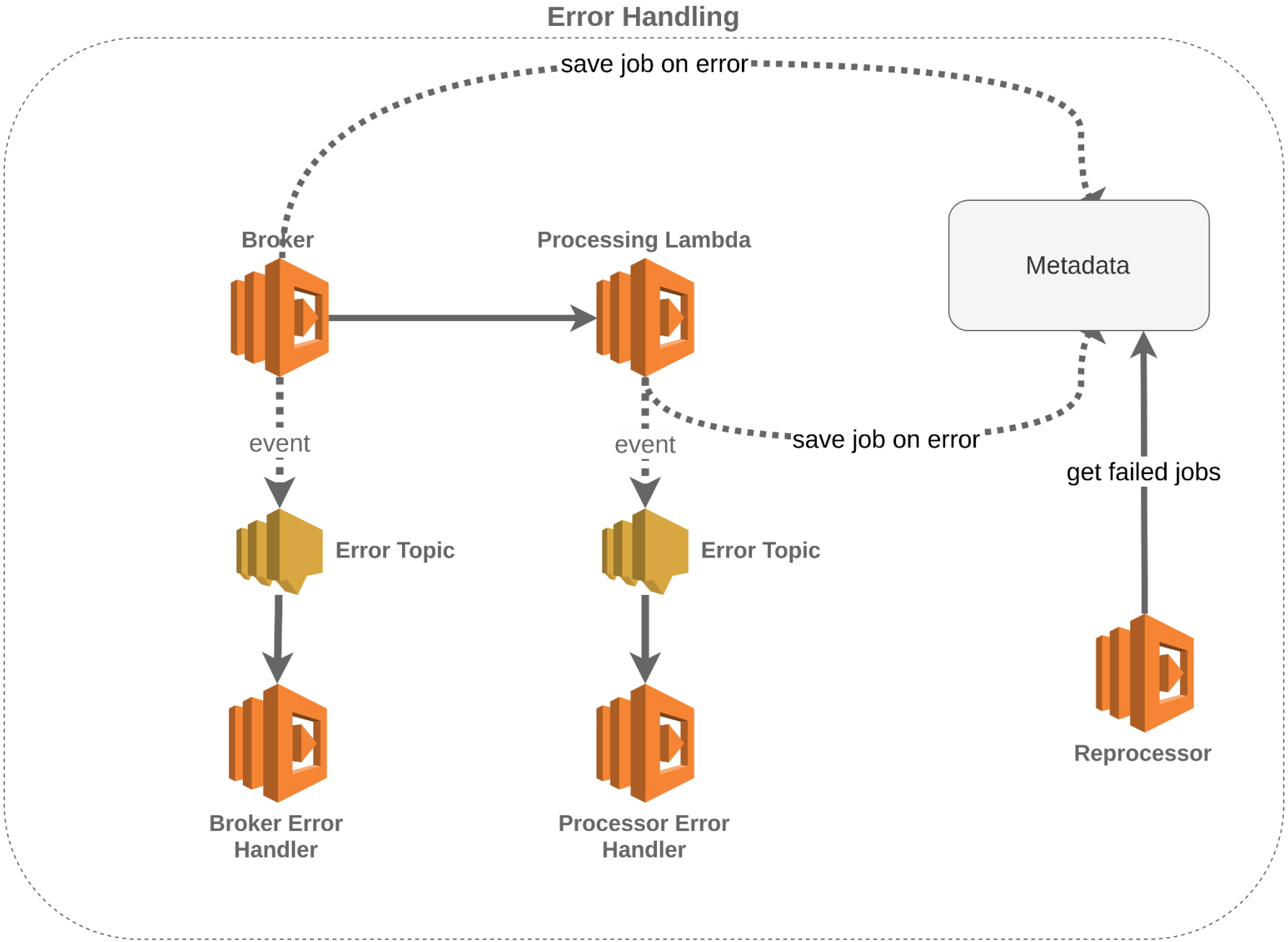

Job Save
DynamoDB
S3
Utilizado para reprocessar job
Jobs Topic
Job Save Queue
Job Save Service
ElasticSearch



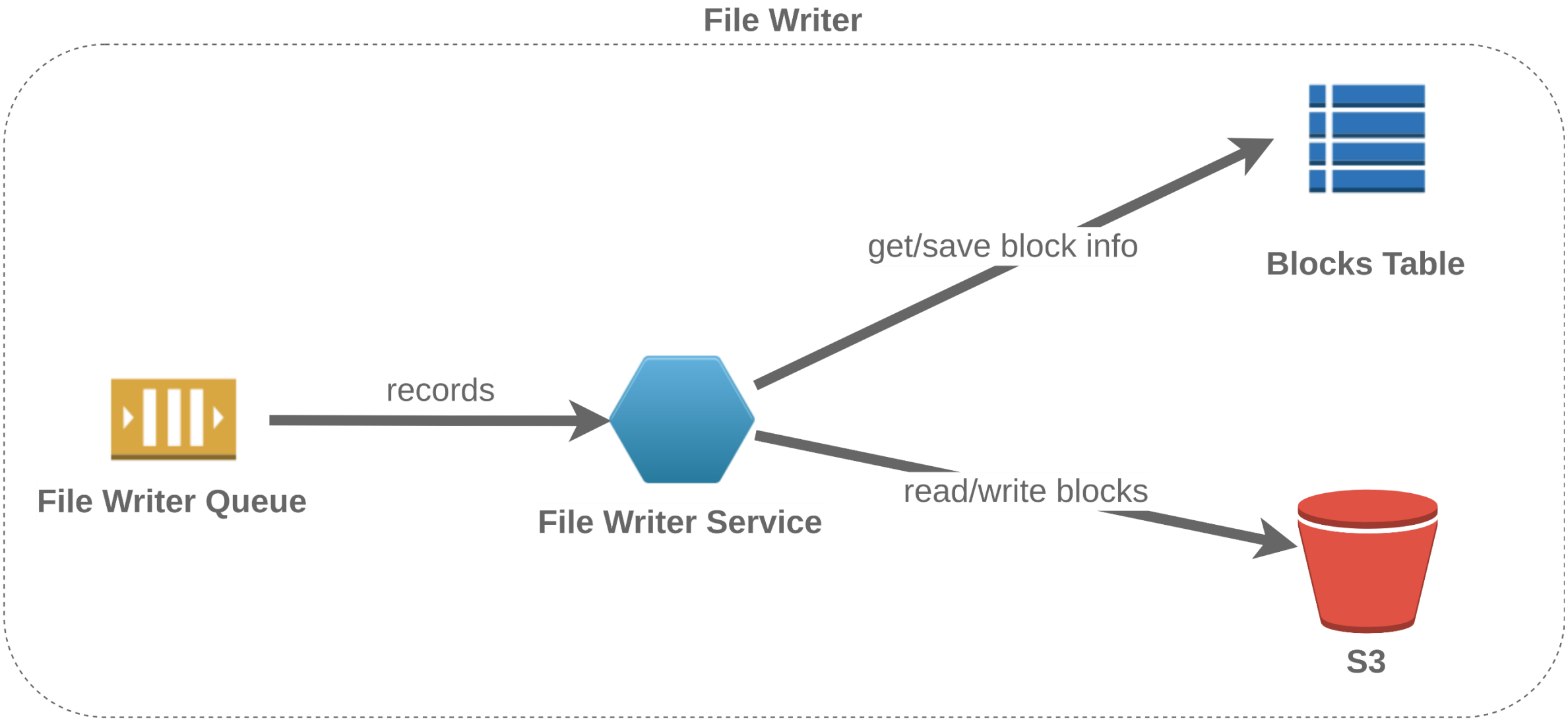
Utilizado para buscas e controle de paralelismo e de falhas


Job Save
DynamoDB
Jobs Topic
Job Save Queue
Job Save Service
S3
ElasticSearch

Lidando com Erros

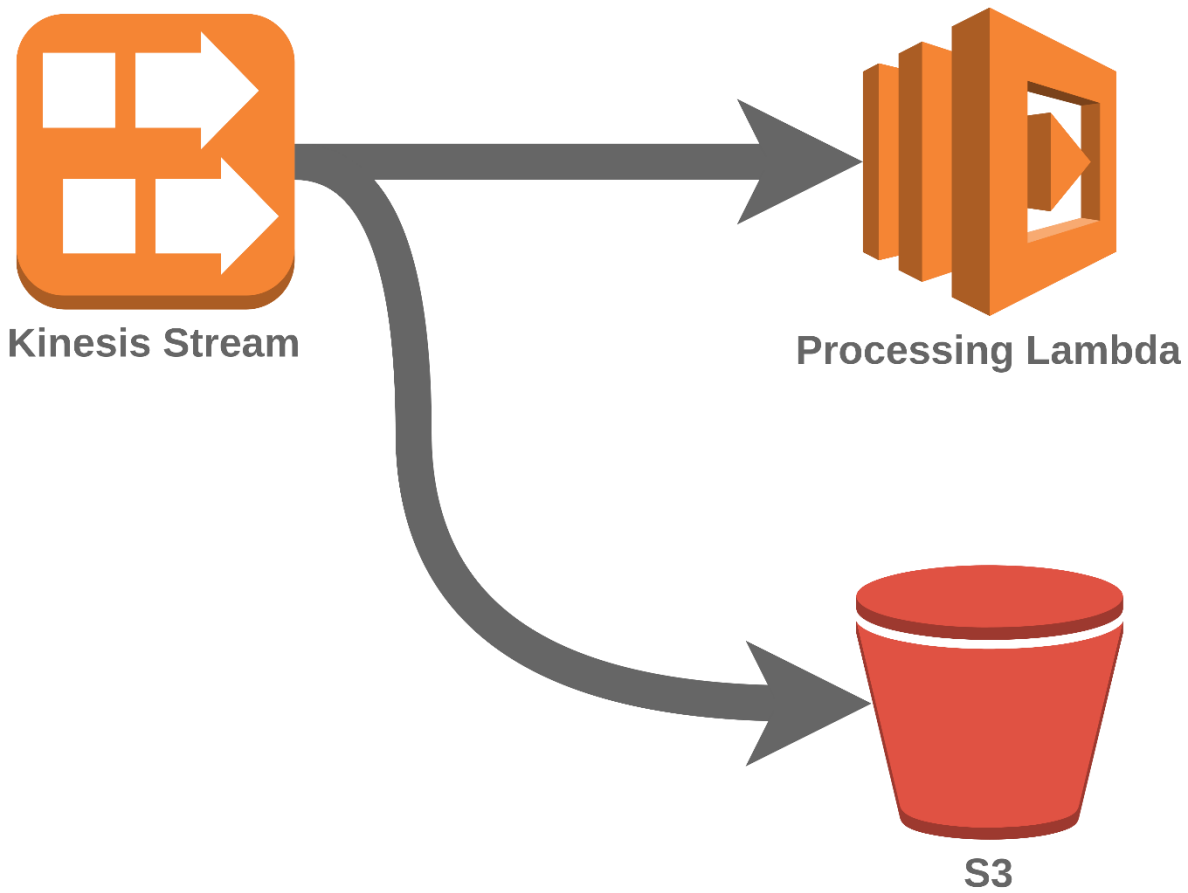

Error Handling
save job on error
Broker
Processing Lambda
Metadata
event
event
save job on error
get failed jobs
Error Topic
Error Topic
Reprocessor
Broker Error Handler
Processor Error Handler

Do Oracle ao (S3 + Presto) em Near Real Time


File Writer
get/save block info
Blocks Table
records
File Writer Queue
File Writer Service
read/write blocks
S3

Amazon Kinesis

Faça Cópia das suas Streams!



Quanto custa uma stream?

1 shard = 1 MB/s ou 1000 registros/s

Você paga pela hora/shard

E por cada carga de PUT de 25KB

Reduzindo os custos do Kinesis

- Faça agregação dos dados com o **Kinesis Producer Library** (KPL)

Reduzindo os custos do Kinesis

- Faça agregação dos dados com o **Kinesis Producer Library** (KPL)
- Dimensione corretamente suas streams

Reduzindo os custos do Kinesis

- Faça agregação dos dados com o **Kinesis Producer Library** (KPL)
- Dimensione corretamente suas streams
- Uma stream X N Streams

Como Utilizamos o Kinesis

- **Utilizamos apenas uma stream no Kinesis** com streams "lógicas" identificadas pela partitioning key.

Como Utilizamos o Kinesis

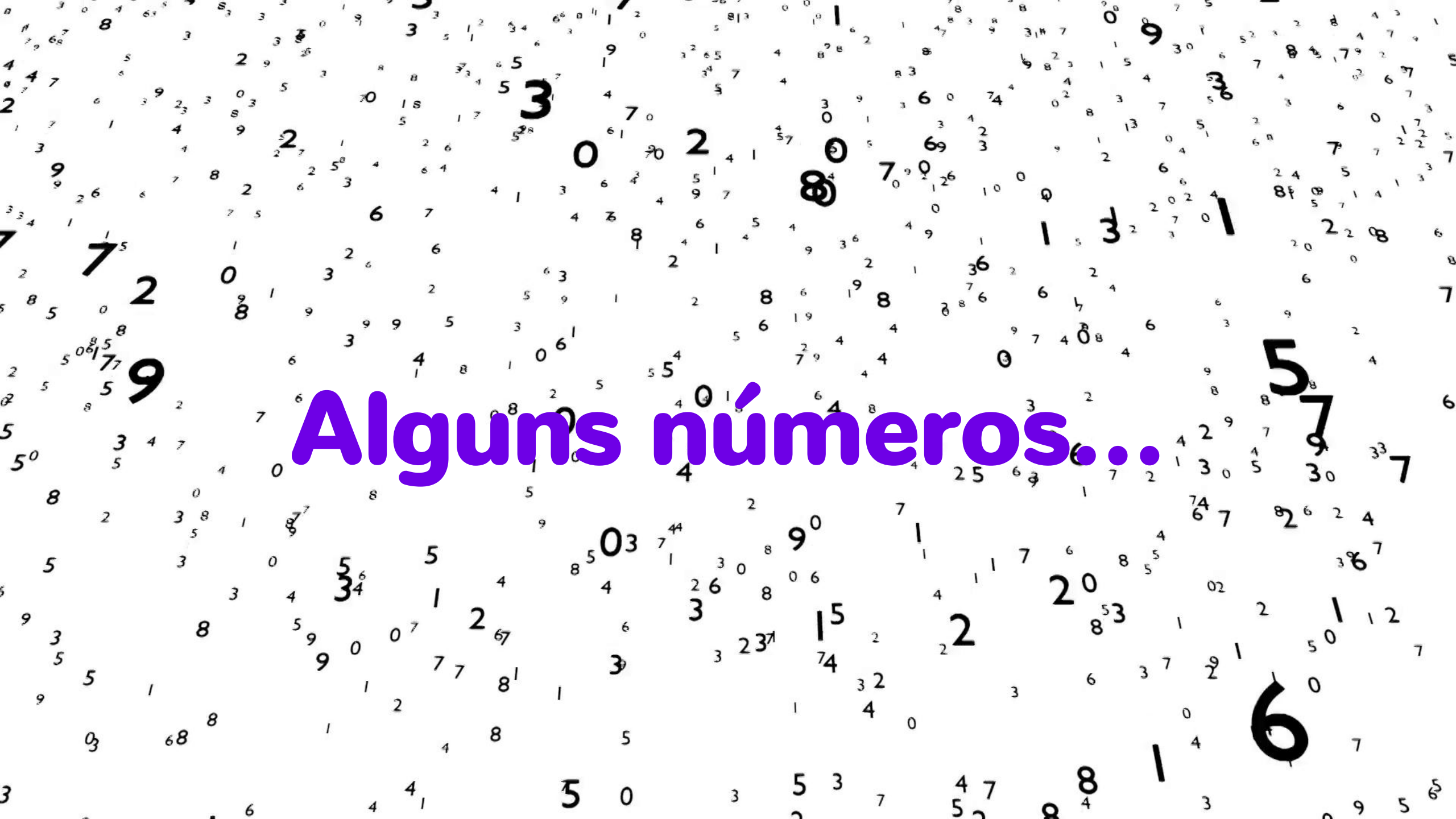
- **Utilizamos apenas uma stream no Kinesis** com streams "lógicas" identificadas pela partitioning key.
- **Agregamos os dados em micro-batches**, e ainda utilizamos o KPL.

Como Utilizamos o Kinesis

- **Utilizamos apenas uma stream no Kinesis** com streams "lógicas" identificadas pela partitioning key.
- **Agregamos os dados em micro-batches**, e ainda utilizamos o KPL.
- **Codificamos os dados em ORC** com strings em binário.

Como Utilizamos o Kinesis

- **Utilizamos apenas uma stream no Kinesis** com streams “lógicas” identificadas pela partitioning key.
- **Agregamos os dados em micro-batches**, e ainda utilizamos o KPL.
- **Codificamos os dados em ORC** com strings em binário.
- **Todas streams possuem schema**, mantemos o caos na RAW.


Alguns números...

12 Milhões
jobs executados

192 Milhões
registros processados

+120 GB
dados processados


PONTOS FORTES
PONTOS FRACOS

PONTOS FORTES

- Pouco gerenciamento
- Integração simples entre serviços da AWS
- Estabilidade dos serviços
- Facilidade em escalar serviços

PONTOS FRACOS

- Custos podem crescer rapidamente
- Subutilização de recursos (ex. Kinesis)
- Explosão de chamadas Lambda
- Saiba quando sua Lambda deve virar um microserviço ou EC2


Nossa arquitetura está em constante evolução. Começamos pequeno e crescemos de forma orgânica, na medida em que os cenários mudam.

Obrigado!

mayconbordin

mayconbordin@gmail.com

Sicredi