
AWS
re:Invent
DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

SVS321

AWS Lambda and Apache Kafka for real-time data processing applications

Julian Wood
(he/him)
Principal Developer Advocate
AWS

What are we talking about today?

01 Understanding data streaming

02 Streaming data on AWS

03 Streaming architecture

04 Processing streaming data

05 AWS Lambda and Kafka

06 Managing performance

Resources

s12d.com/svs321-24

About me

Julian Wood

Principal Developer Advocate – AWS Serverless

Recovering server“more” infrastructure engineer

- Enterprises and startups

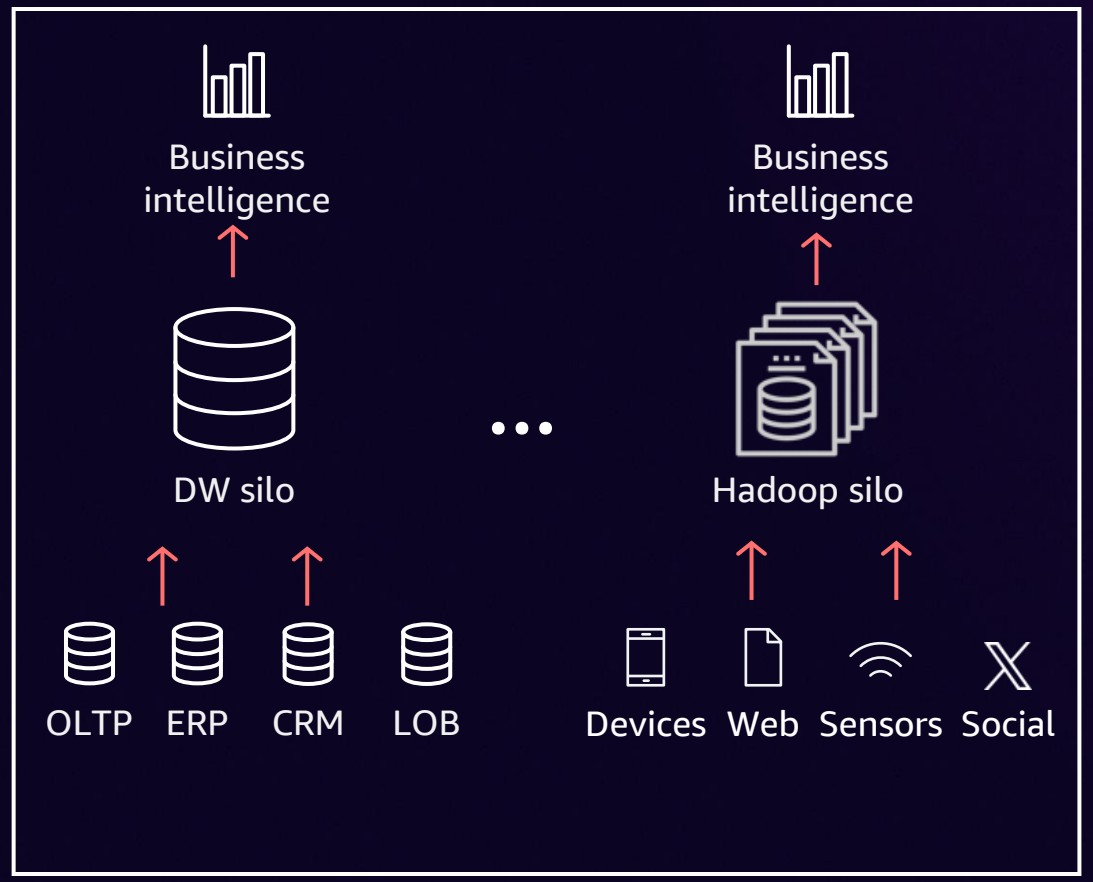
You can’t scare me, I have twin girls!

From Cape Town via London

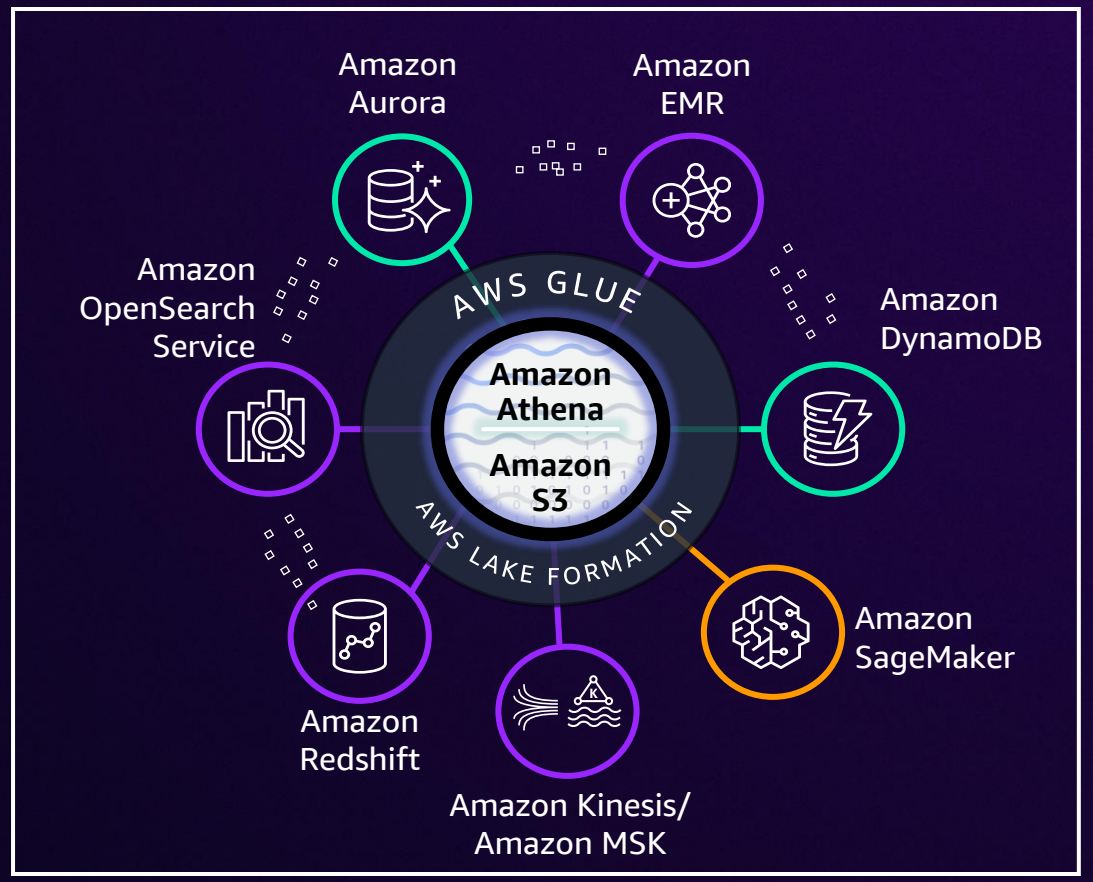
Understanding data streaming

Data processing is evolving . . .

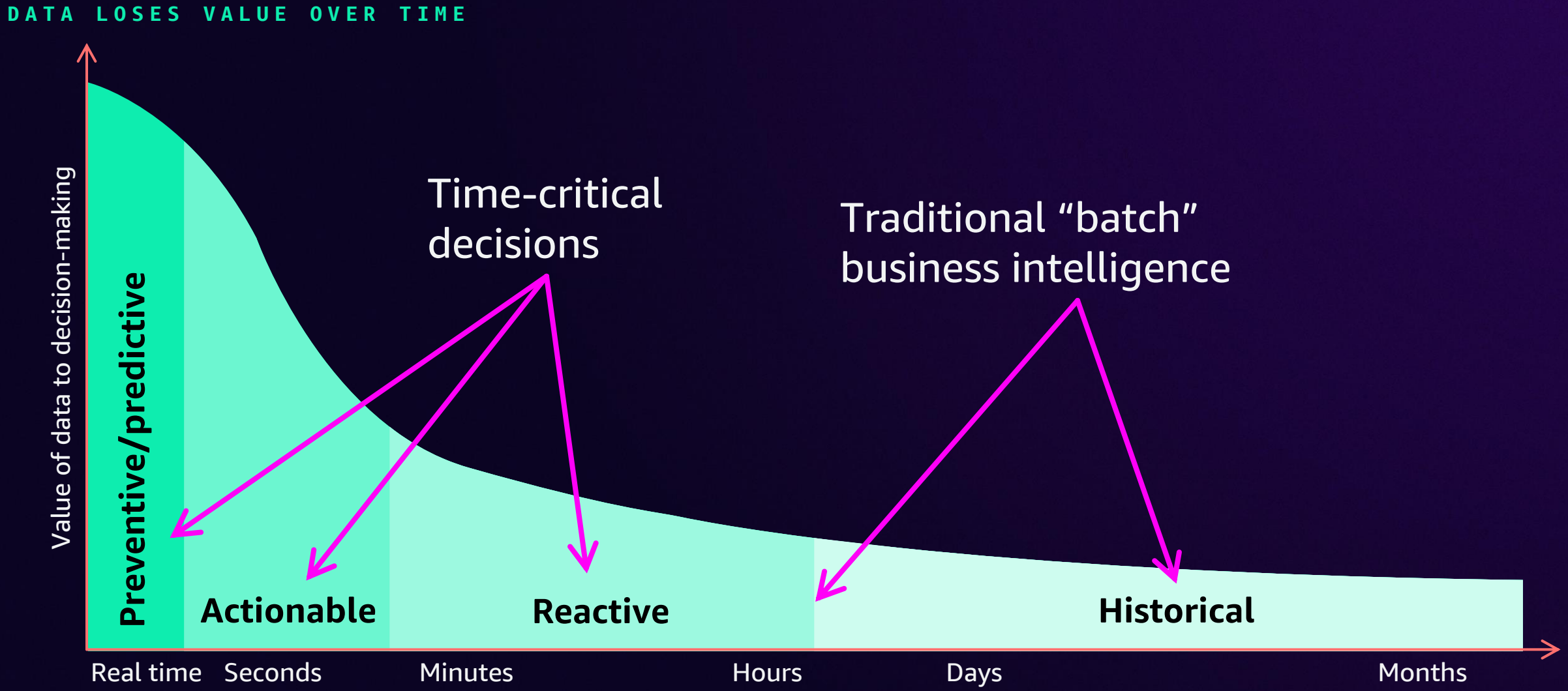
Siloed data



Modern data architectures



Streaming enables real-time processing



Source: Mike Gualtieri, Forrester, *Perishable Insights*

Streaming enables real-time processing

Batch processing

Hourly server logs

Weekly or monthly bills

Daily website clickstream

Daily fraud reports

Stream processing

Real-time metrics

Real-time spending alerts and caps

Real-time clickstream analysis

Real-time detection

Common uses of streaming data

Industrial automation

Online gaming interactions

IoT device monitoring

Clickstream data

Data lakes

Log analytics

Streaming data pipeline

Stream sources

IoT sensors

Apps

Social media

[Wed Oct 11 14:32:52 2023] [error] [client /live/ap/htdocs/test

Logs

Stream ingestion

AWS IoT Core

Kinesis Agent

AWS SDK

AWS Database Migration Service (AWS DMS)

MSK Connect

Stream storage

kafka

Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Stream analytics & integration

Stream analytics

Amazon Managed Service for Apache Flink

AWS Lambda

Amazon EMR

AWS Glue

Stream integration

Amazon Data Firehose

Amazon S3

Amazon Redshift

Amazon OpenSearch Service

Outcome

Automatic decision

Interactive dashboard

Alerting

Real-time ML inference

Streaming data characteristics

High volume

Organizations that collect clickstream data about user behavior can often reach up to 2–3 TB every day

Nearly continuous rather than discrete

IoT devices like sensors collect data continuously and at regular intervals

Ordered

A chat application makes little sense with messages out of order

Time-sensitive

Fraud detection must be done as quickly as possible to prevent loss

Streaming data on AWS

Apache Kafka

What is Apache Kafka?

Apache Kafka

Is it like a service bus?

Is it like REST but async?

Is it like a database?

What is Apache Kafka?

Apache Kafka

Streaming platform

Is it like a service bus?

Is it like REST but async?

Is it like a database?

Running Apache Kafka

- Self-managed (on premises/cloud)
- Amazon MSK

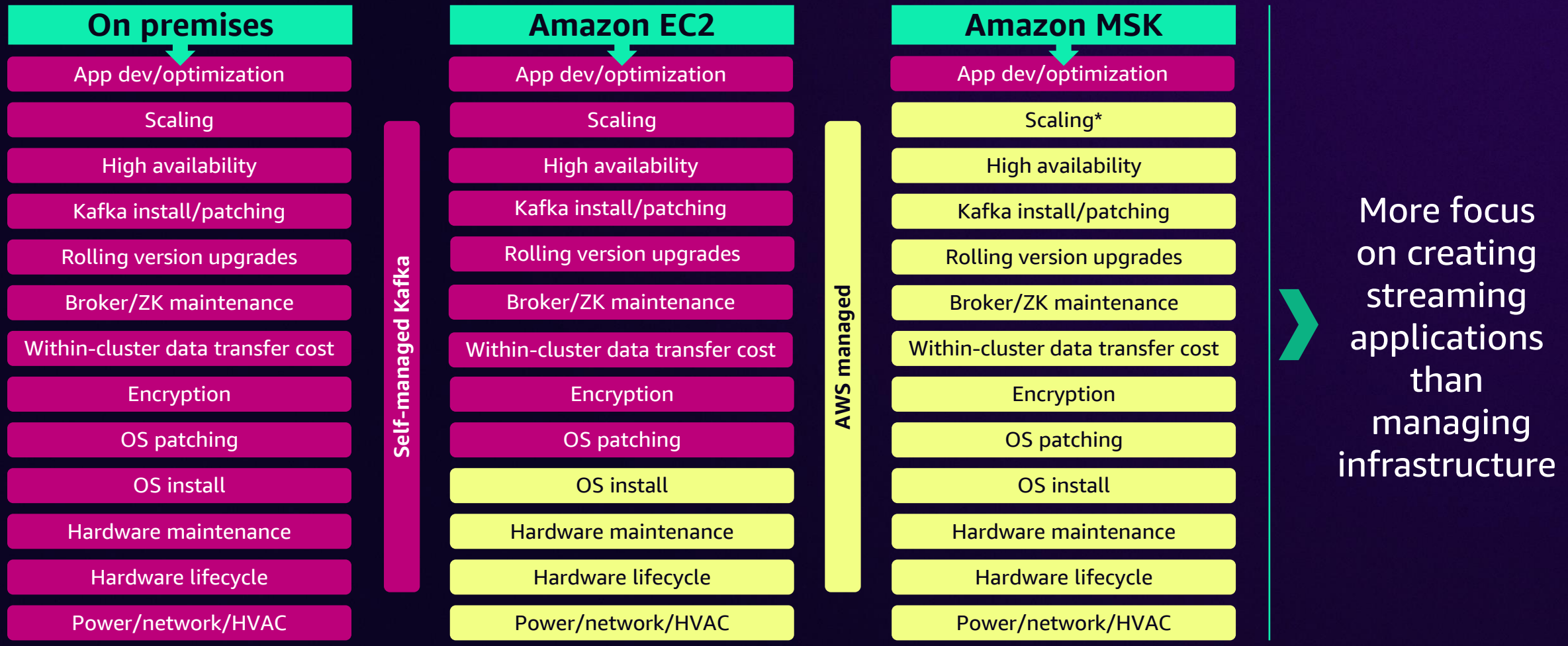
- Confluent Cloud (SaaS)
- Confluent Platform (on premises/cloud)
- Redpanda
- WarpStream

Amazon MSK

Securely stream data with a fully managed, highly available Apache Kafka service

- Automate provisioning, configuring, and tuning
- Fully compatible with open source Apache Kafka
- Highly secure
- Lower cost

Amazon MSK fully managed



*Cluster expansion and scaling storage

Amazon MSK Serverless

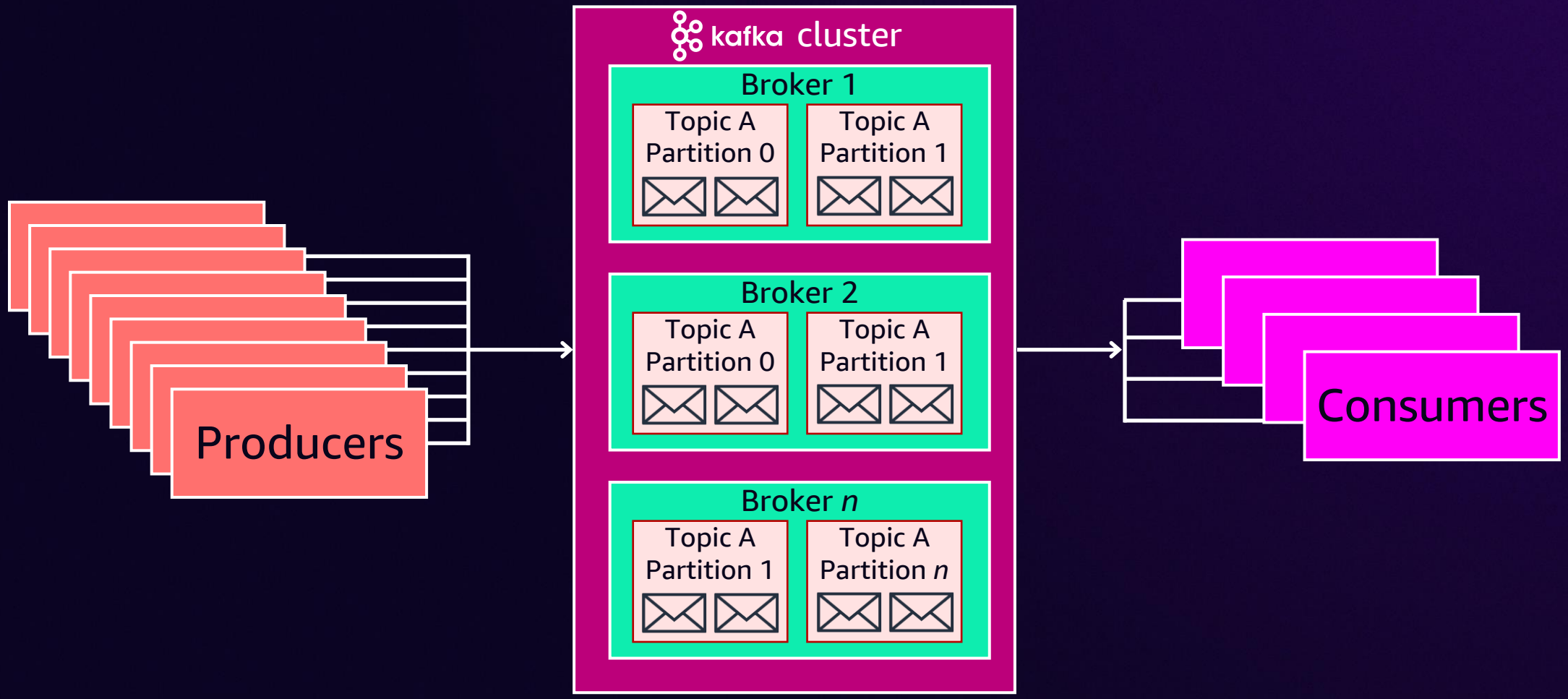
Easily run Apache Kafka clusters without rightsizing cluster capacity

Instantly scale I/O without worrying about scaling capacity up and down or reassigning partitions

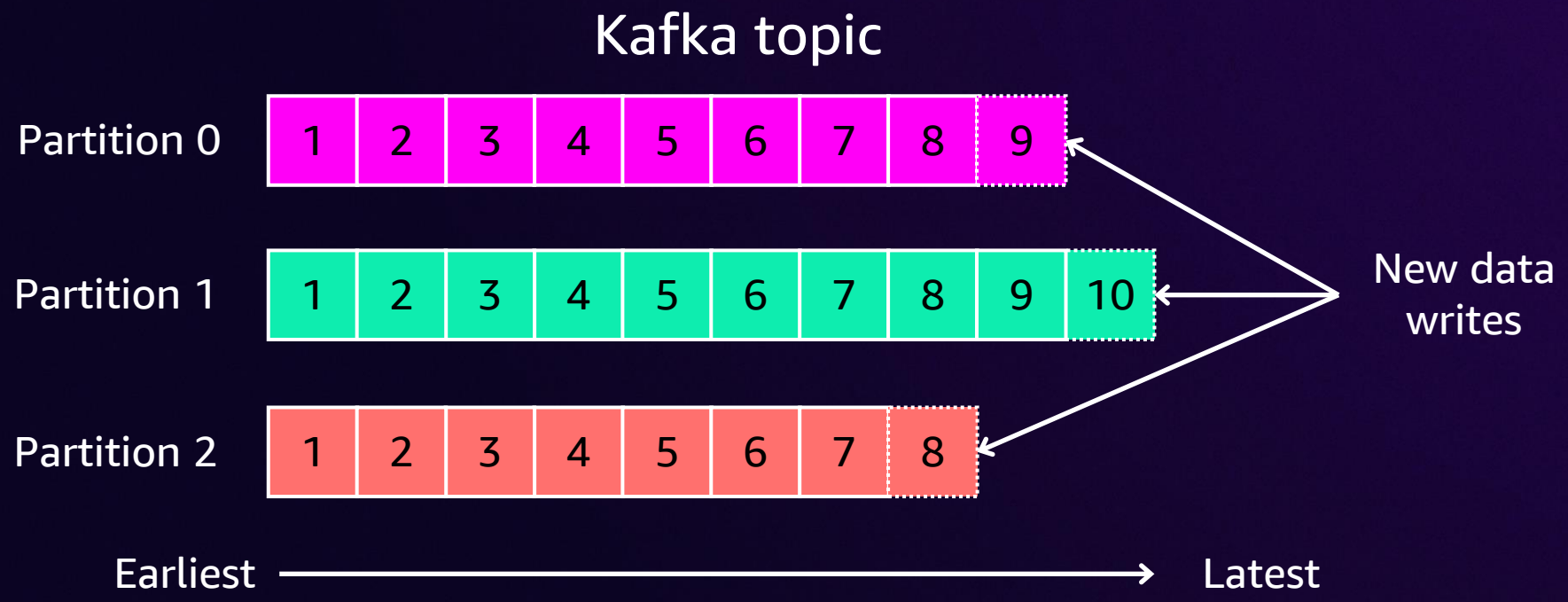
Pay for the data volume you stream and retain

Streaming architecture

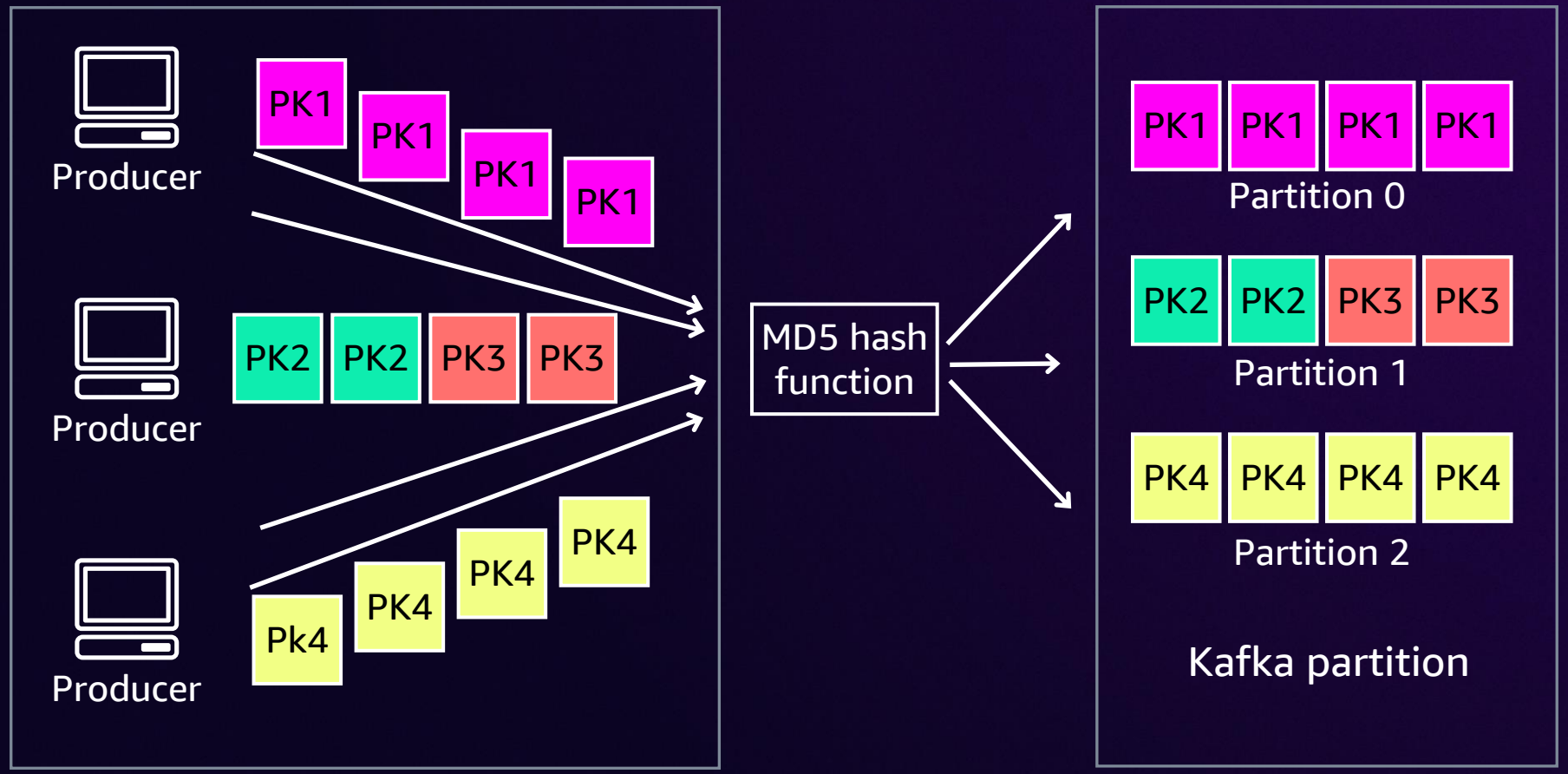
Kafka clusters, brokers, topics, partitions, records



Kafka partitions



Partition keys



Partition assignment

Random hash

- Random hash, records are randomly sent to different partitions
- Load balances records across all available partitions

Time-based hash

- Groups of records sent to a single partition if they arrive at the same time
- Identical timestamp results in an identical hash

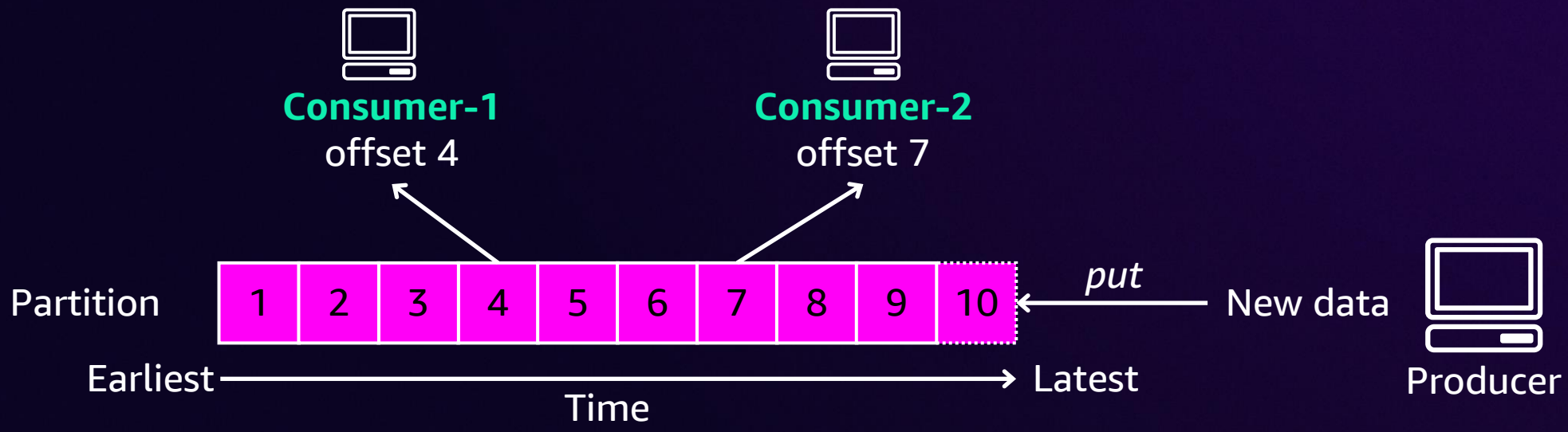
Application-specific hash: Example: customerID as the key

- All records for that customer are routed to the same partition, always in order
- Useful for downstream aggregation logic
- May limit the record capacity per customerID

Kafka offset

Tracks partition record sequence

Consumers store this to keep track of their position in the stream



Processing streaming data

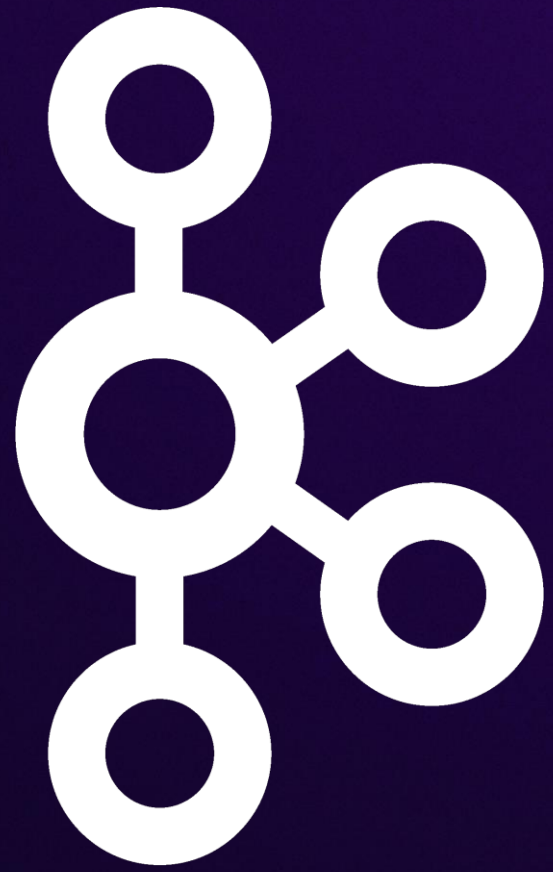
Consuming streaming records

Processing

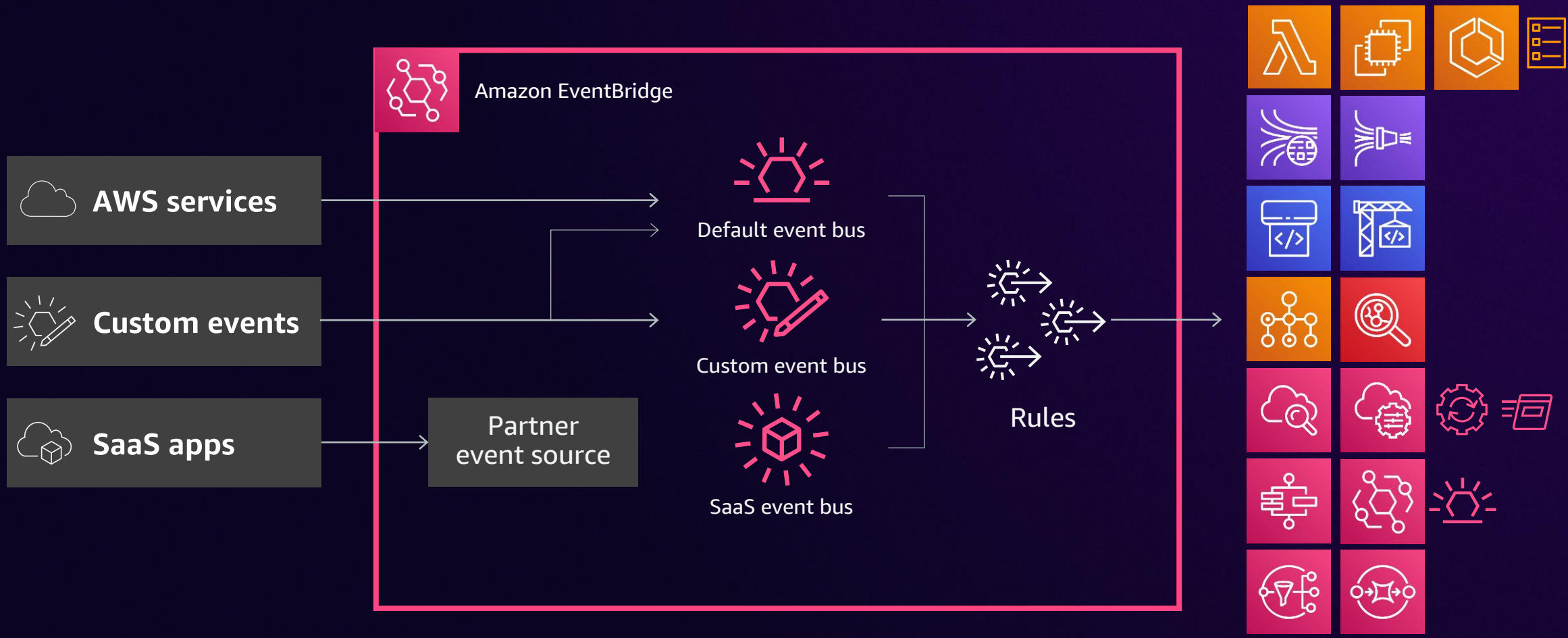
- Amazon EventBridge bus
- Amazon EventBridge Pipes
- Lambda event source mapping
- Kafka Connect

Analytics

- Amazon Managed Service for Apache Flink
- AWS Glue



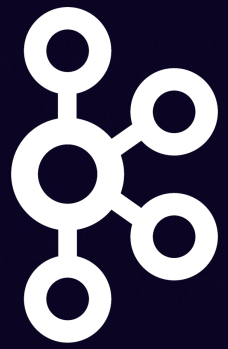
Amazon EventBridge event bus



Kafka and EventBridge bus

Kafka

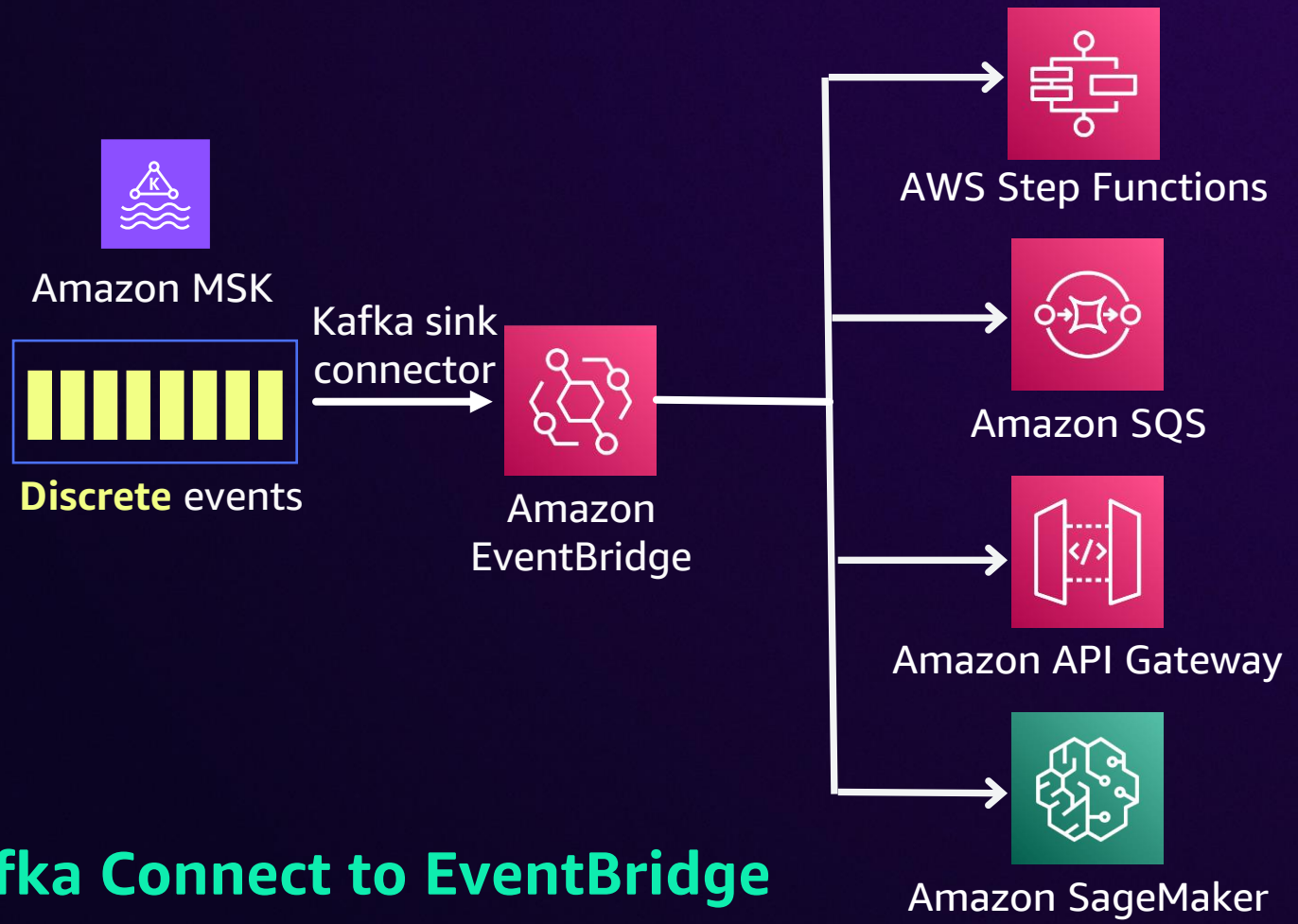
- Stream events (pull)
- Requires smart endpoints
- Open source, run anywhere
- Fully managed (Amazon MSK)



EventBridge

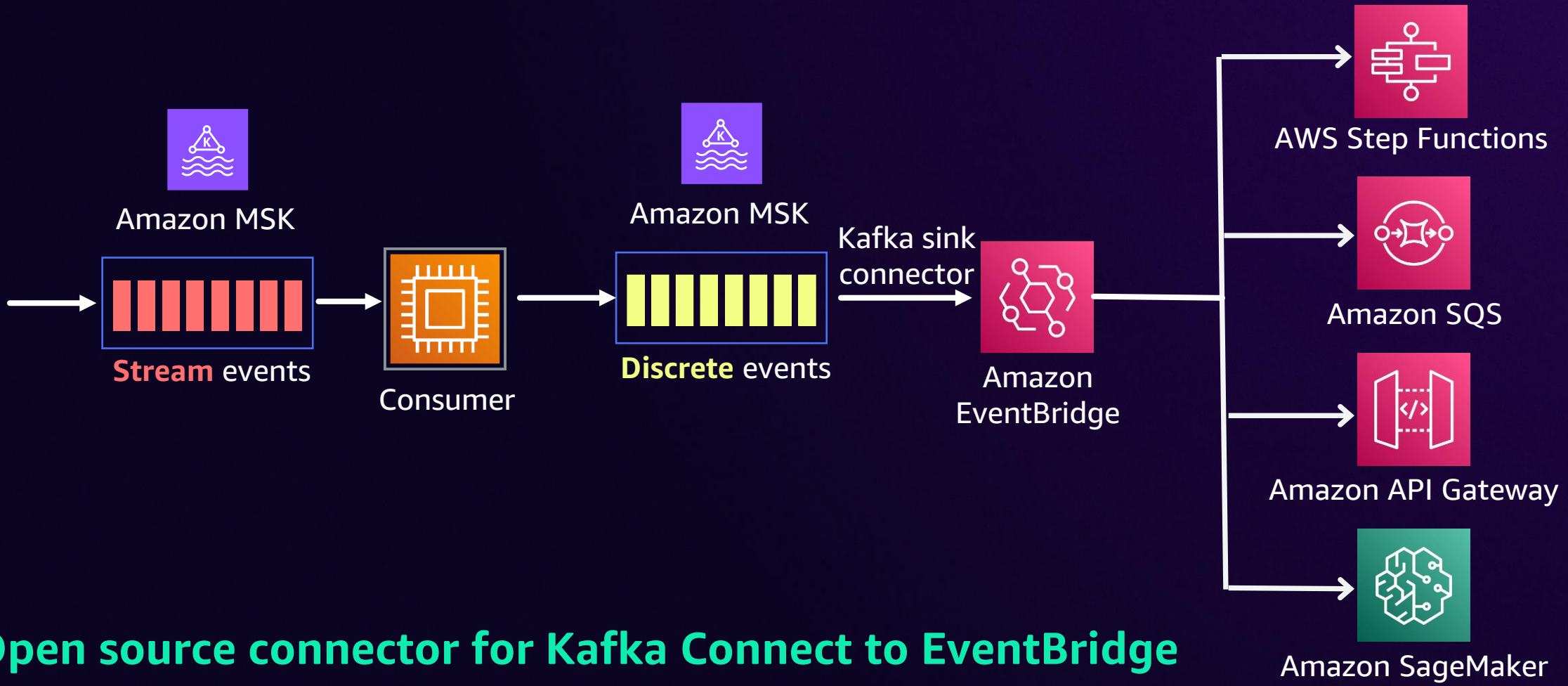
- Discrete events (push)
- AWS and partner integrations
- Low code
- Fully managed

Kafka connector for Amazon EventBridge



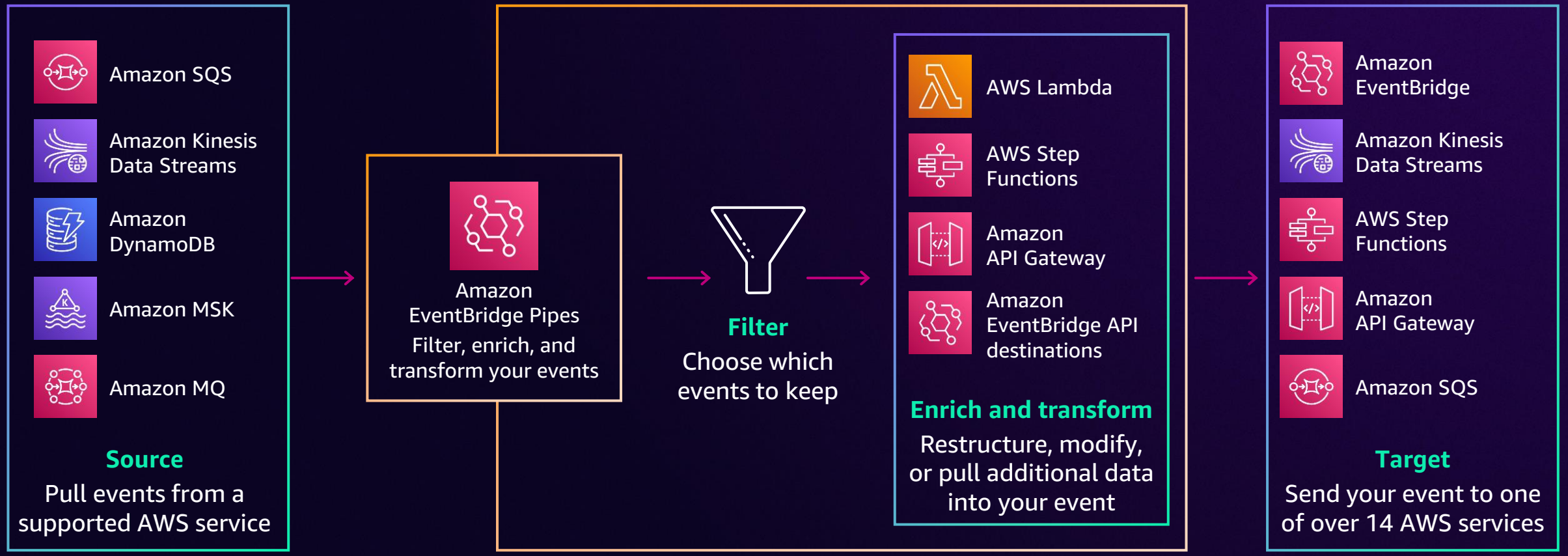
Open source connector for Kafka Connect to EventBridge

Kafka connector for Amazon EventBridge



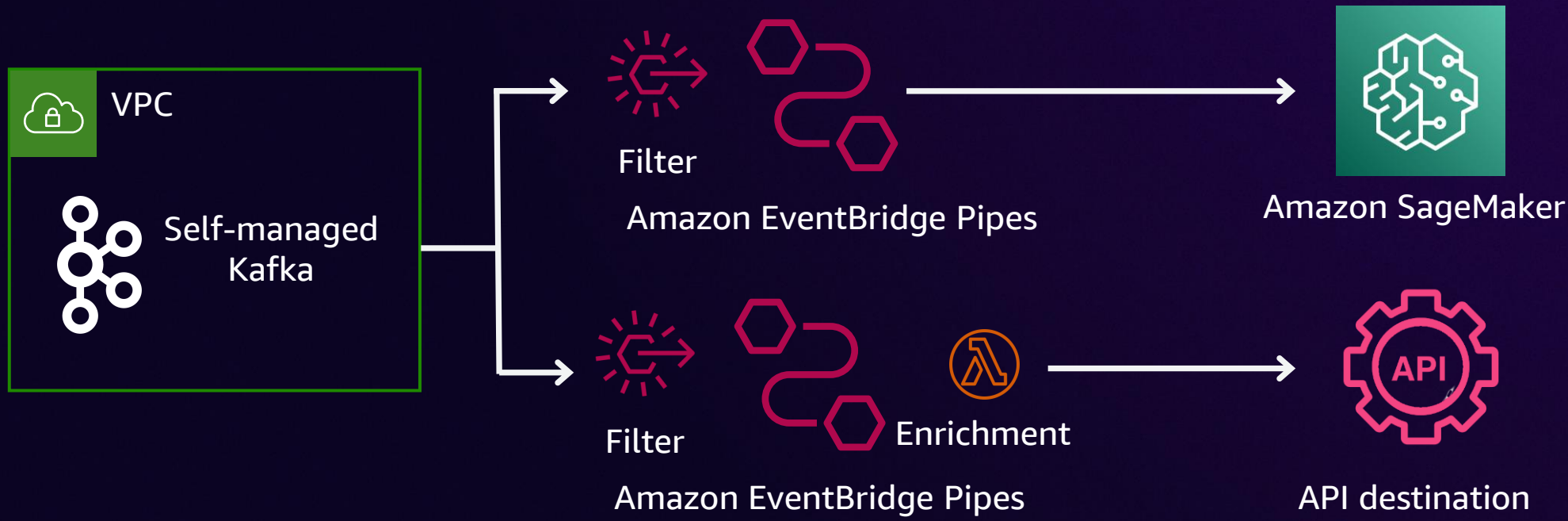
Open source connector for Kafka Connect to EventBridge

EventBridge Pipes: Point-to-point



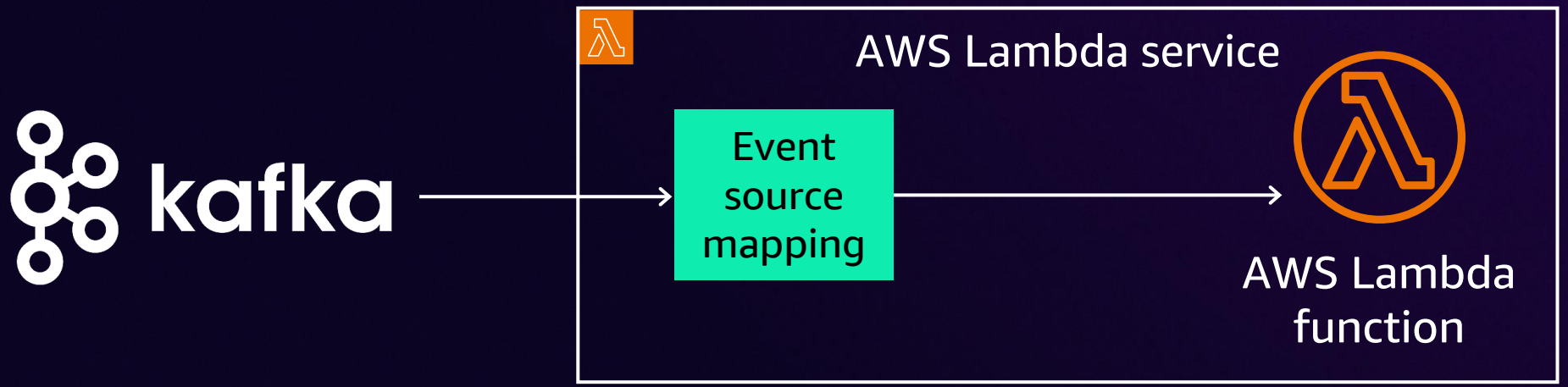
Kafka use case: Connect to API destination

Connect your self-managed Kafka cluster to other AWS services and HTTP APIs without writing any code

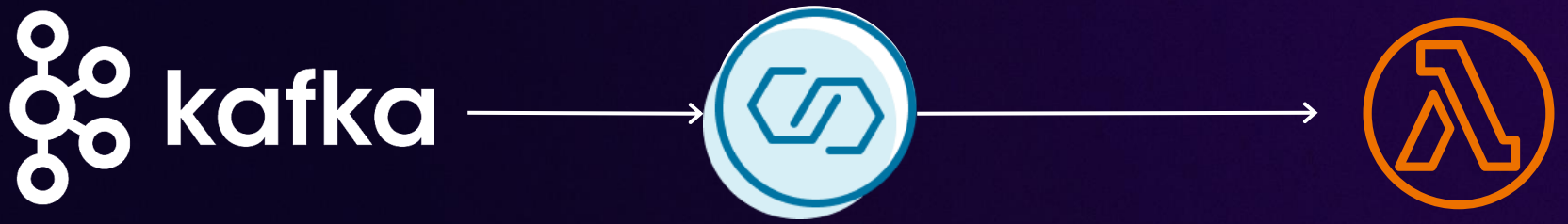


AWS Lambda and Kafka

Kafka Lambda consumer options



Kafka/Confluent Lambda sink connector



Polls Kafka partitions and calls your function synchronously or asynchronously

At-least once semantics

Option to batch records

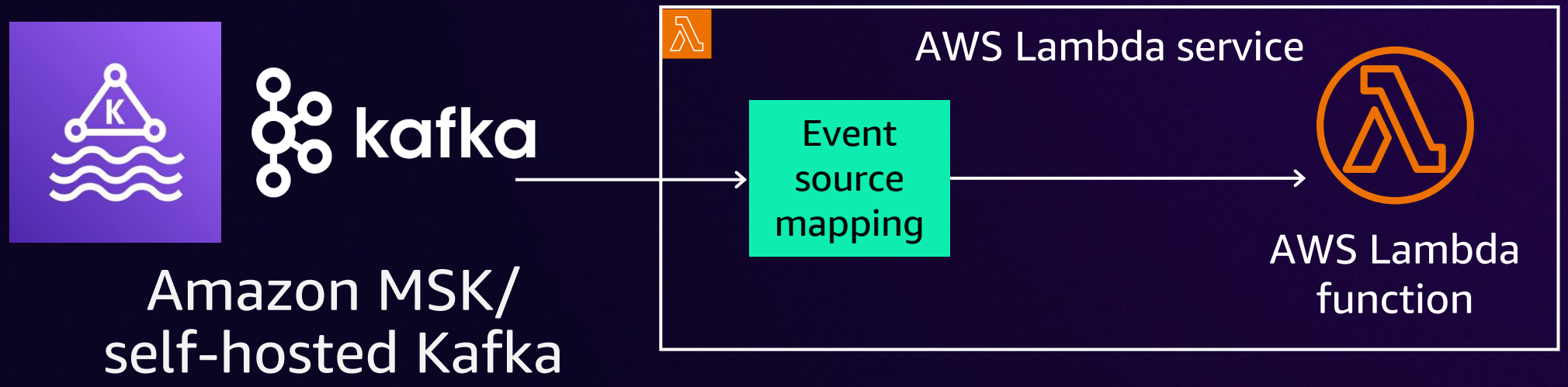
Scales up to soft maximum of 10 connectors

Amazon MSK Connect capacity options: provisioned/auto scaled based on workers

Sink connector error handling

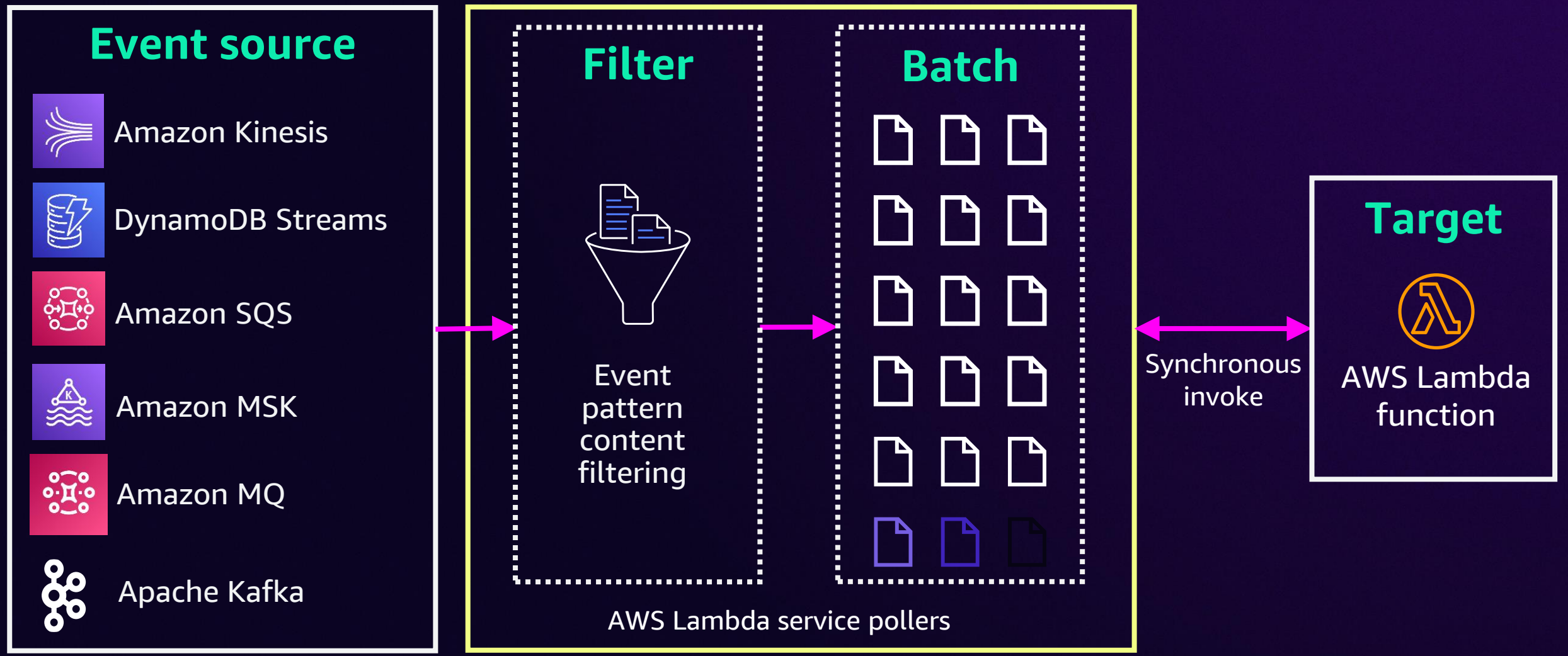
- Connector can send to Kafka dead-letter queue (DLQ)
- Lambda error handling semantics follow async/sync invocations
  - Async: Lambda service retries twice (three total attempts)
    - Send to Lambda DLQ/destinations for failed invocations
  - Sync: By default, fails and stops processing for that partition
    - Option to log to another Kafka topic and continue processing

Lambda event source mapping



Lambda event source mapping

LAMBDA RESOURCE THAT READS FROM AN EVENT SOURCE AND INVOKES A LAMBDA FUNCTION



Lambda event source mapping

Serverless processing using Lambda functions

Lambda service polls Kafka automatically

Custom processing in any language runtime

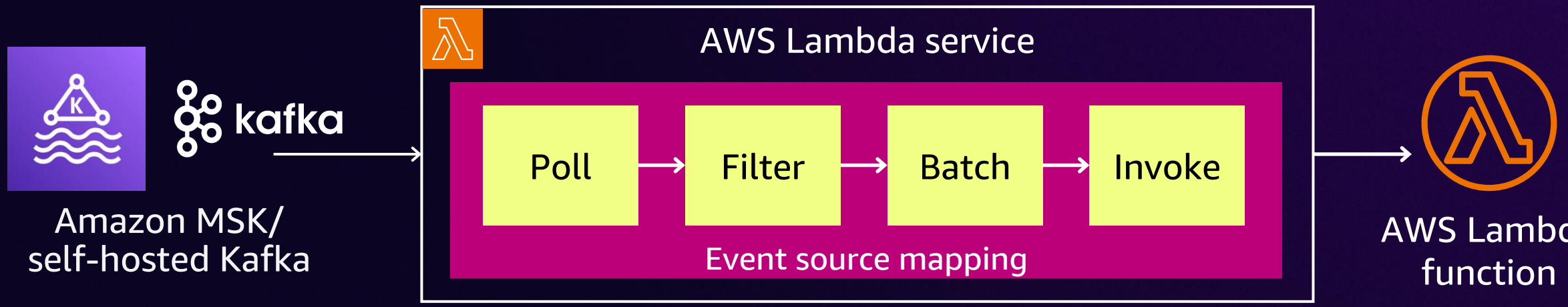
Configurable starting position

Supports optional filtering

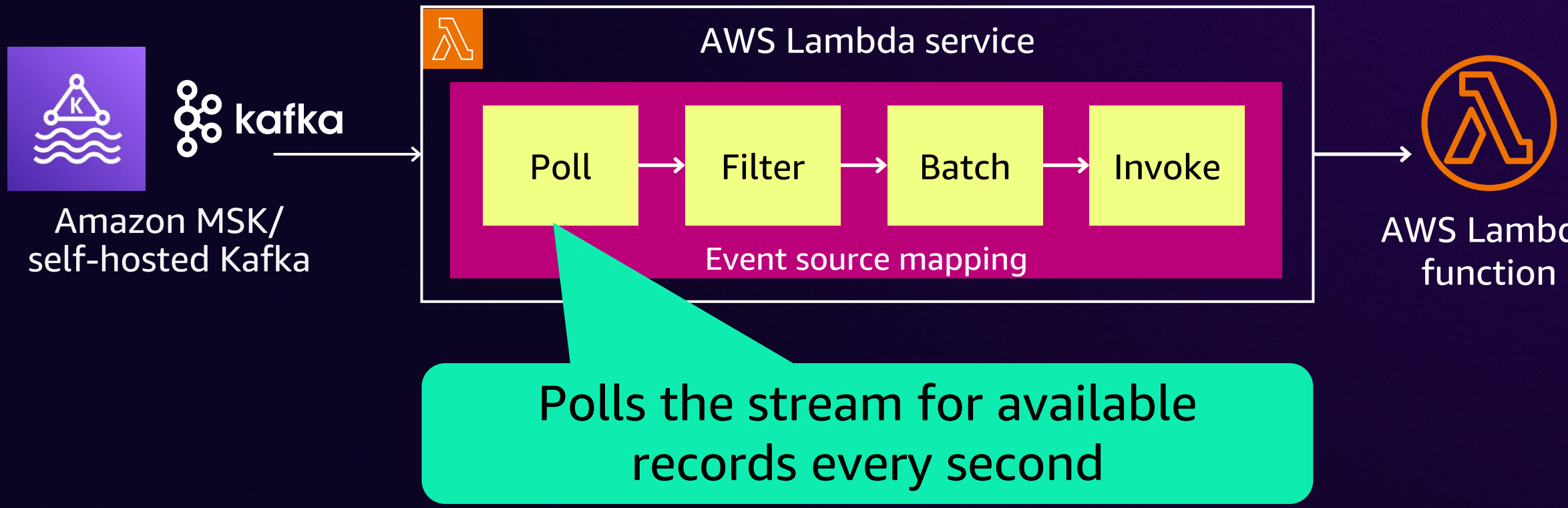
Records delivered in a batch as a payload

AWS Lambda

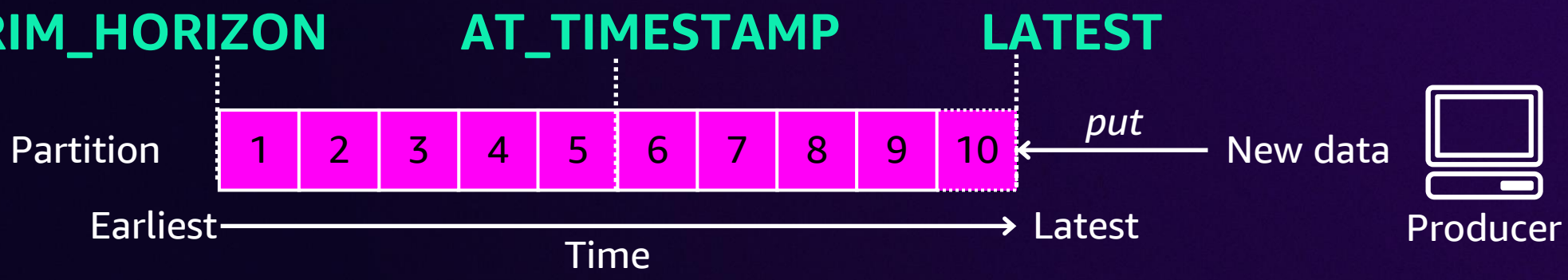
Event source mapping



Event source mapping: Polling



Event source mapping: Starting position



TRIM_HORIZON
Start reading at earliest record in the stream

AT_TIMESTAMP
Start reading at specific timestamp

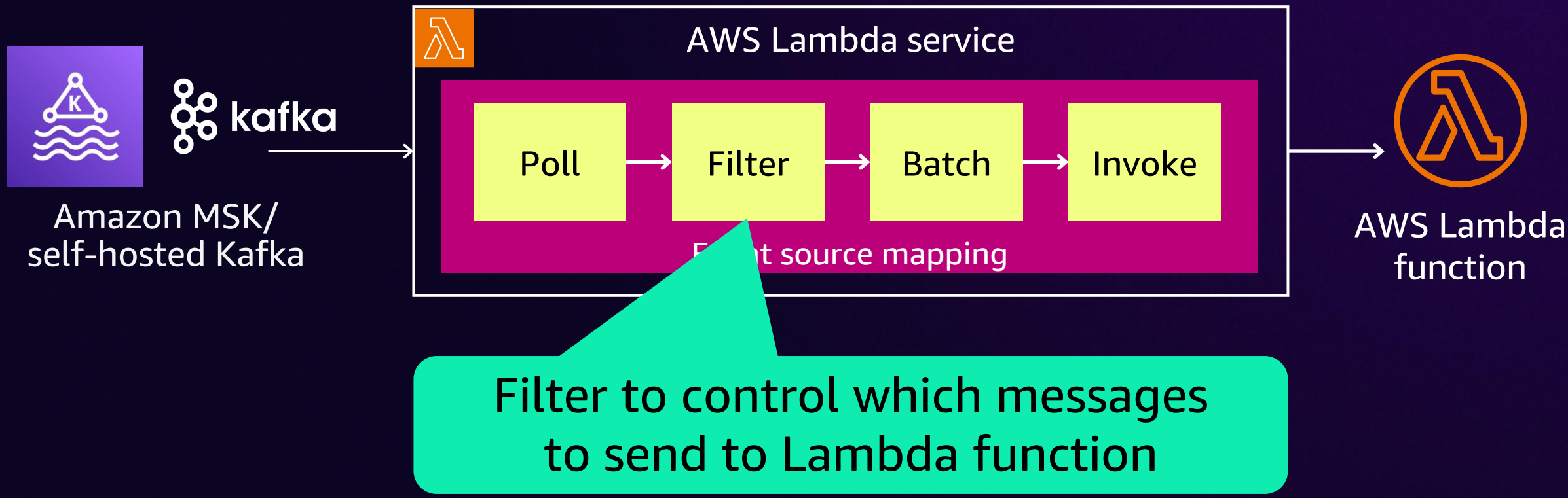
LATEST
Start reading from latest record once Lambda configures the ESM

Possible delay between configuring the ESM and processing the first record

Kafka event source mapping: Consumer group ID

- Any Kafka topic consumer uses a consumer group
- ESM creates this by default, generates randomly unique value
  - Lambda identified as a new consumer to start processing at a specified position
- Can create ESM that uses a custom consumer group ID
  - Connect Lambda to an existing consumer group
  - Consuming starts from where Kafka recorded the consumer group left off
  - Connect Lambda to a topic that is replicated using **MirrorMaker 2**

Event source mapping: Filtering



Event source mapping: Filtering

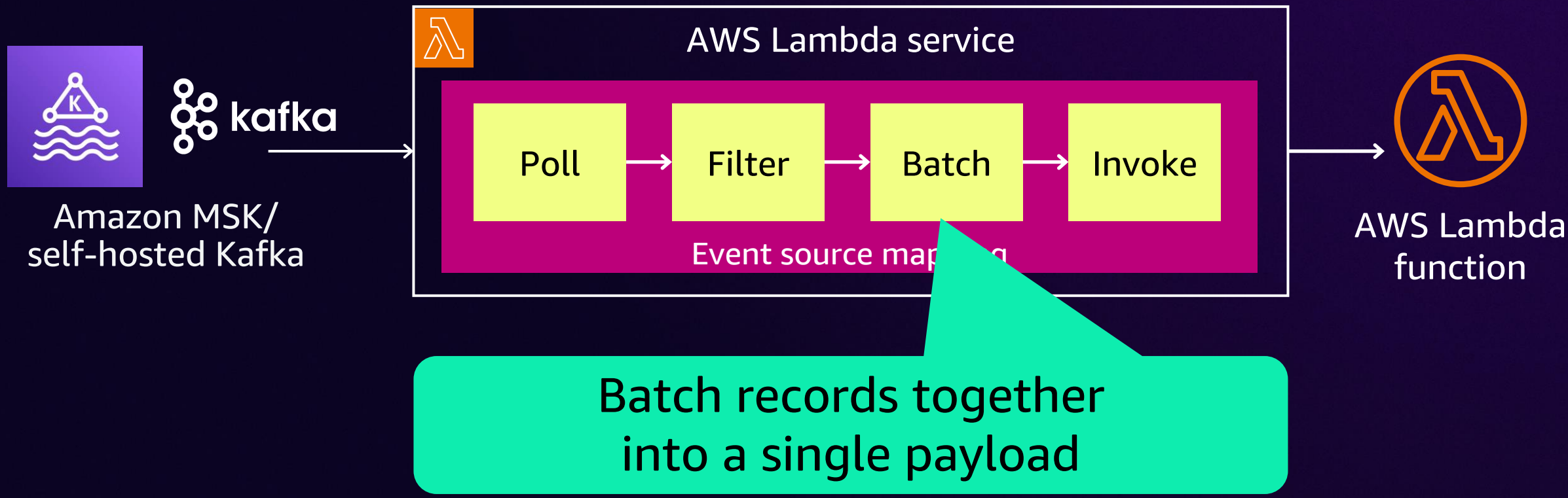
```
FleetTirePressureMapping:
  Type: AWS::Lambda::EventSourceMapping
  Properties:
    FunctionName: fleet-tire-pressure-evaluator
    BatchSize: 100
    StartingPosition: LATEST
    EventSourceArn:

      arn:aws:kafka:...:stream/fleet-telemetry


    FilterCriteria:
      Filters:
        - Pattern: |
          {
            "data": {
              "tire_pressure": [{"numeric": ["<", 32]}]
            }
          }
```

Filter incoming messages **before function invocation**

Reduce traffic to function, **reduce cost**

- Same pattern matching rules as Amazon EventBridge
- Define up to 5 patterns per ESM, up to 2,048 characters each
- Batching applied **after filtering**

Event source mapping: Batching



Event source mapping: Batching

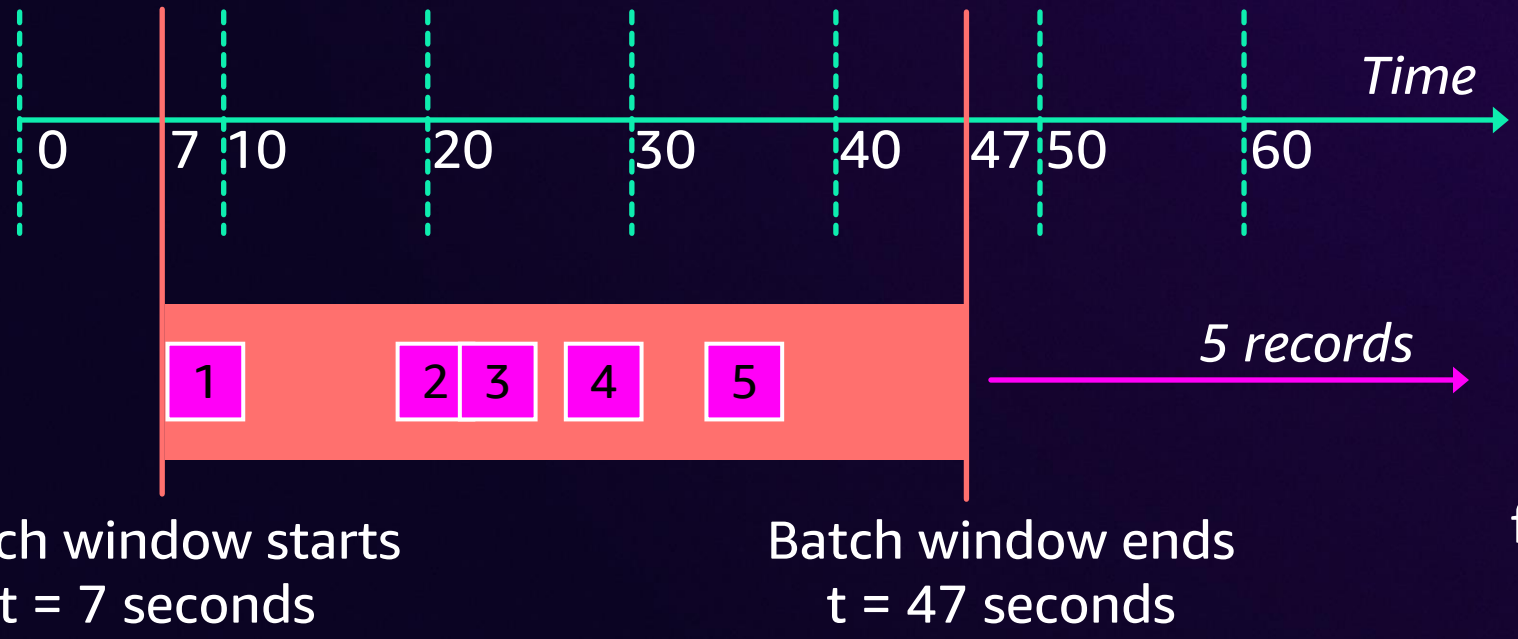
- **Batching window** reaches its maximum value
  - Default batch window is 500 ms
  - Can configure any value from 0 to 300 seconds (5 minutes) in increments of seconds
- **Batch size** is met
  - Minimum/default batch size = 1, maximum = 10,000
- **Payload size** reaches 6 MB
  - Cannot modify this limit, as this is the Lambda invocation payload limit

Batching: Maximum batch window reached

Maximum batching window = 40 seconds

Maximum batch size = 10

Maximum batch bytes = 6 MB

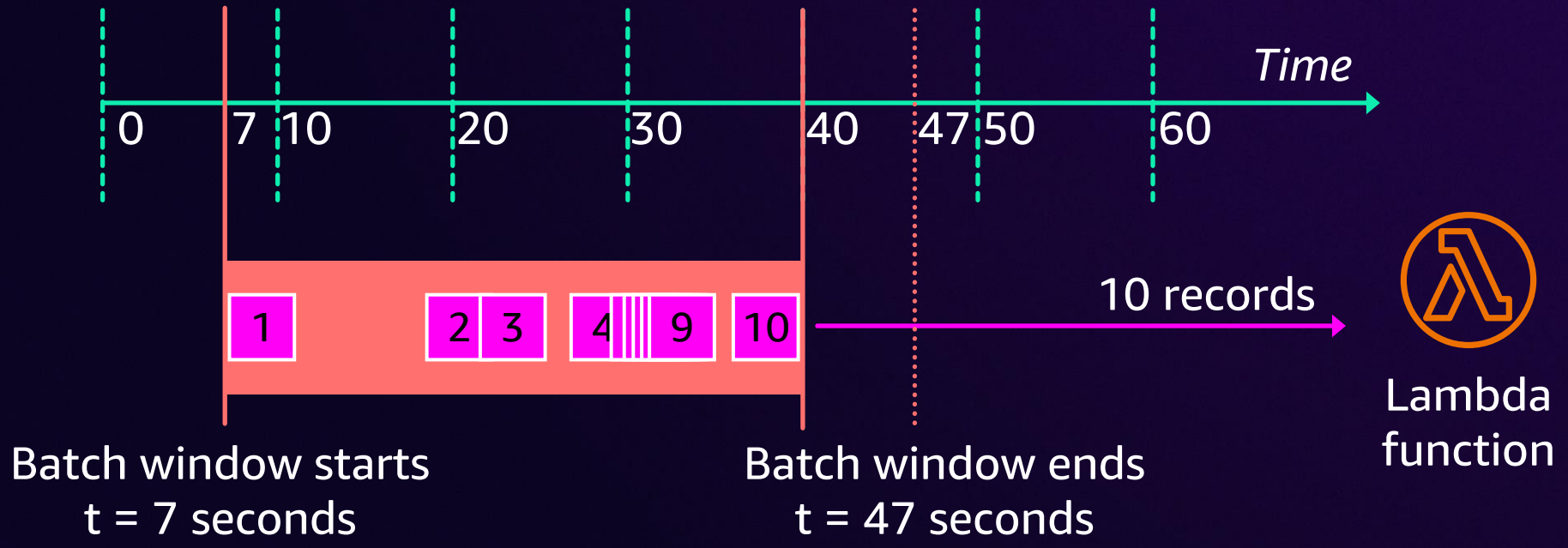


Batching: Maximum batch size reached

Maximum batching window = 40 seconds

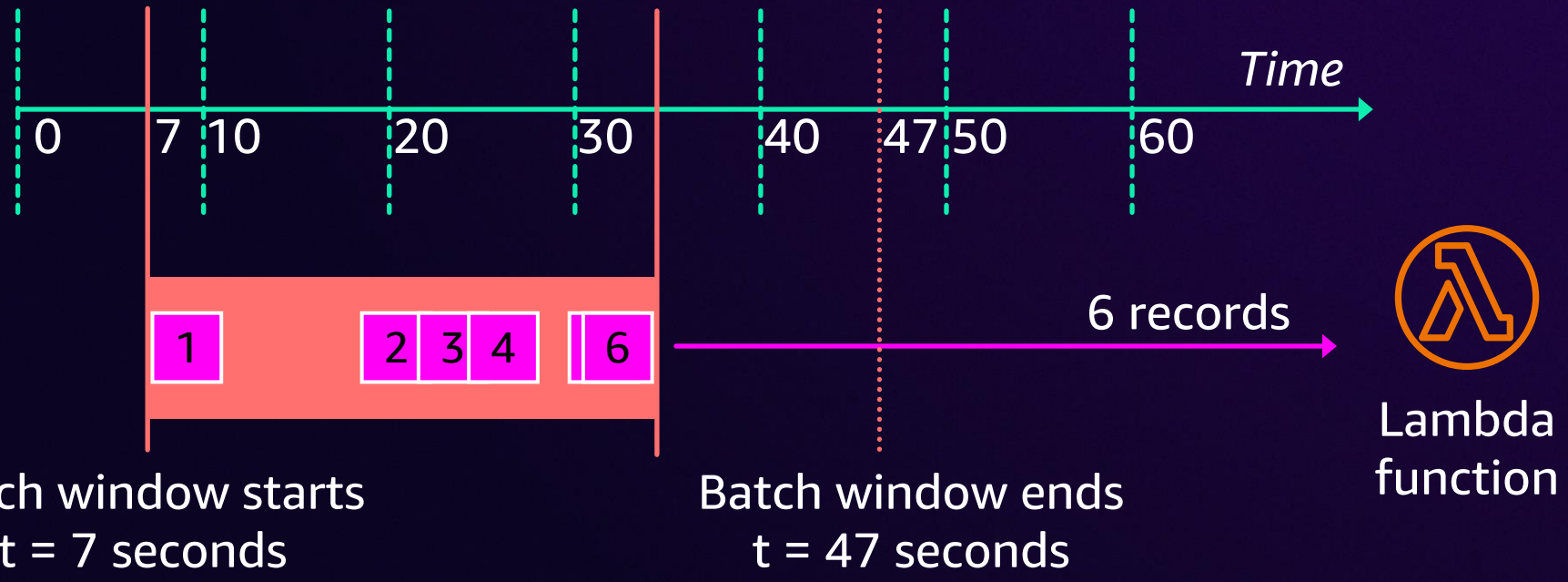
Maximum batch size = 10

Maximum batch bytes = 6 MB

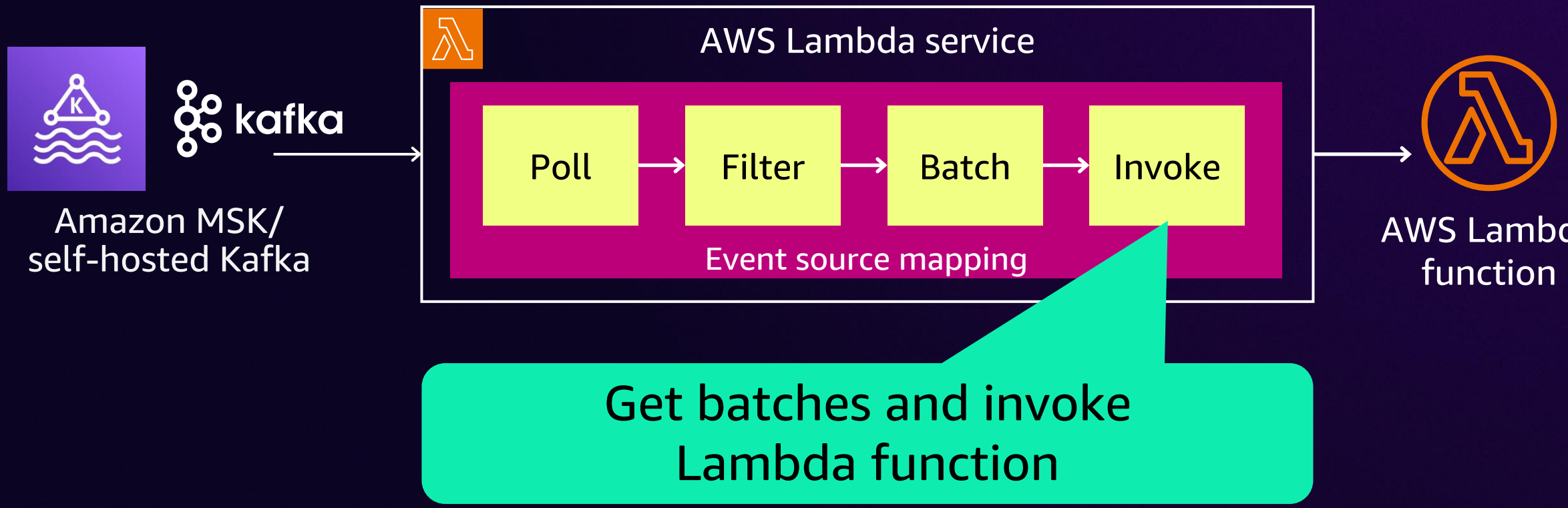


Batching: Maximum batch payload size reached

Maximum batching window = 40 seconds
Maximum batch size = 10
Maximum batch bytes = 6 MB



Event source mapping



Example events

Kafka


```
{
    "eventSource": "aws:kafka",
    "eventSourceArn": "arn:aws:kafka:us-east-1:123456789012:cluster/vpc-2priv-2pub/751d2973-a626-4…
    "records": {
      "mytopic-0": [
          {
            "topic": "mytopic"
            "partition": "0",
            "offset": 15,
            "timestamp": 1545084650987,
            "timestampType": "CREATE_TIME",
            "key":"abcDEFghiJKLmnoPQRstuVWXyz1234==",
            "value": "SGVsbG8sIHRoaXMgaXMgYSBOZXN0Lg==",
            "headers":[
               {
                   "headerKey":[
                      104,
                      101,
                      101
                    ]
                }
            ]
          }
.....
```

Lambda authentication to Kafka

Kafka hosting option	SASL/SCRAM +TLS	SASL/PLAIN	IAM	TLS	Unauthenticated
Self-managed Kafka	Yes	Yes	No	Yes	No
Amazon MSK	Yes	No	Yes	Yes	Yes
Amazon MSK Serverless	No	No	Yes	No	No

SASL: Store credentials in AWS Secrets Manager

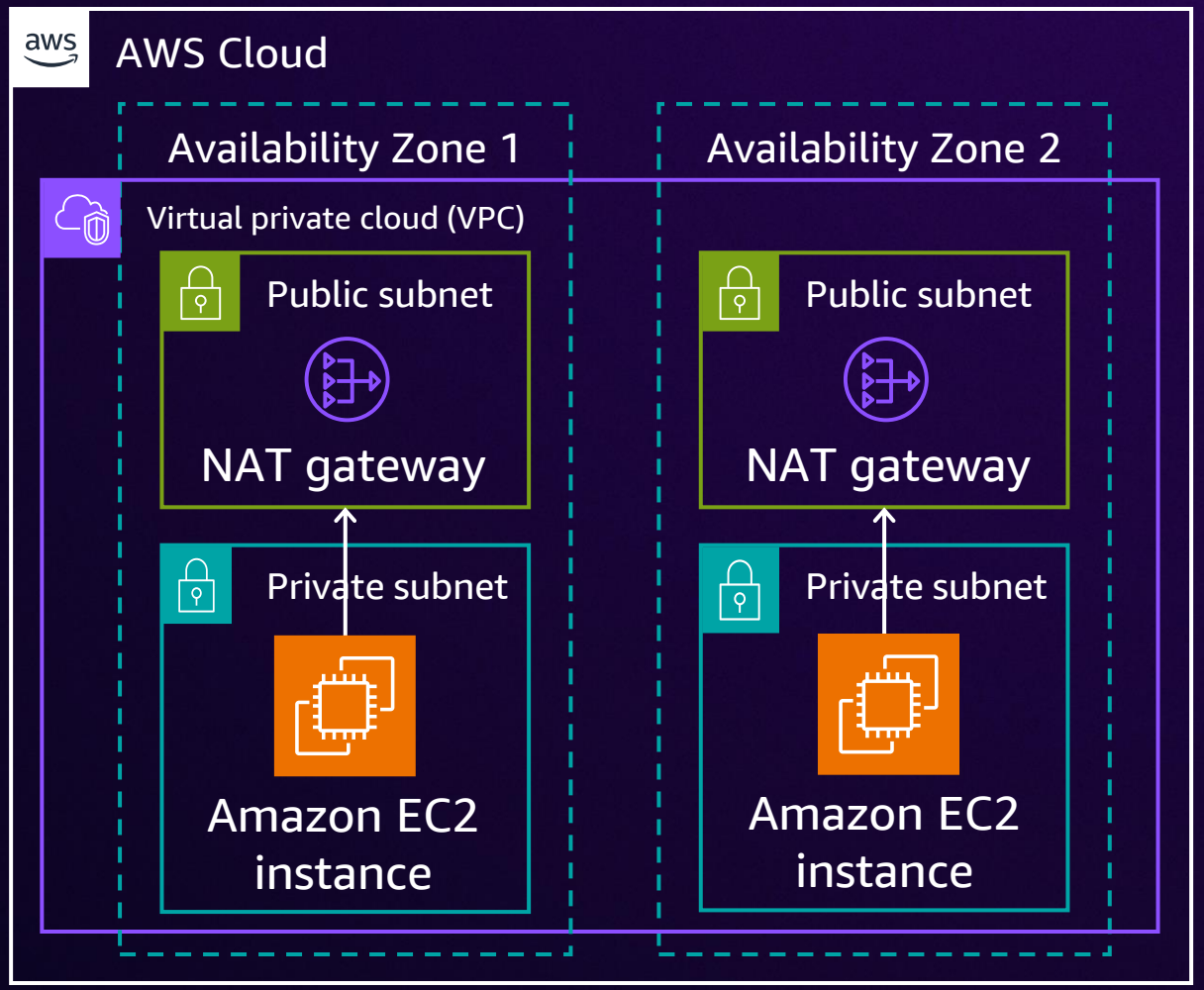
Amazon MSK: Secret name must begin with prefix: *AmazonMSK_*

Amazon MSK Serverless: Only supports IAM

Self-hosted Kafka networking

Lambda ESM connectivity to self-hosted Kafka

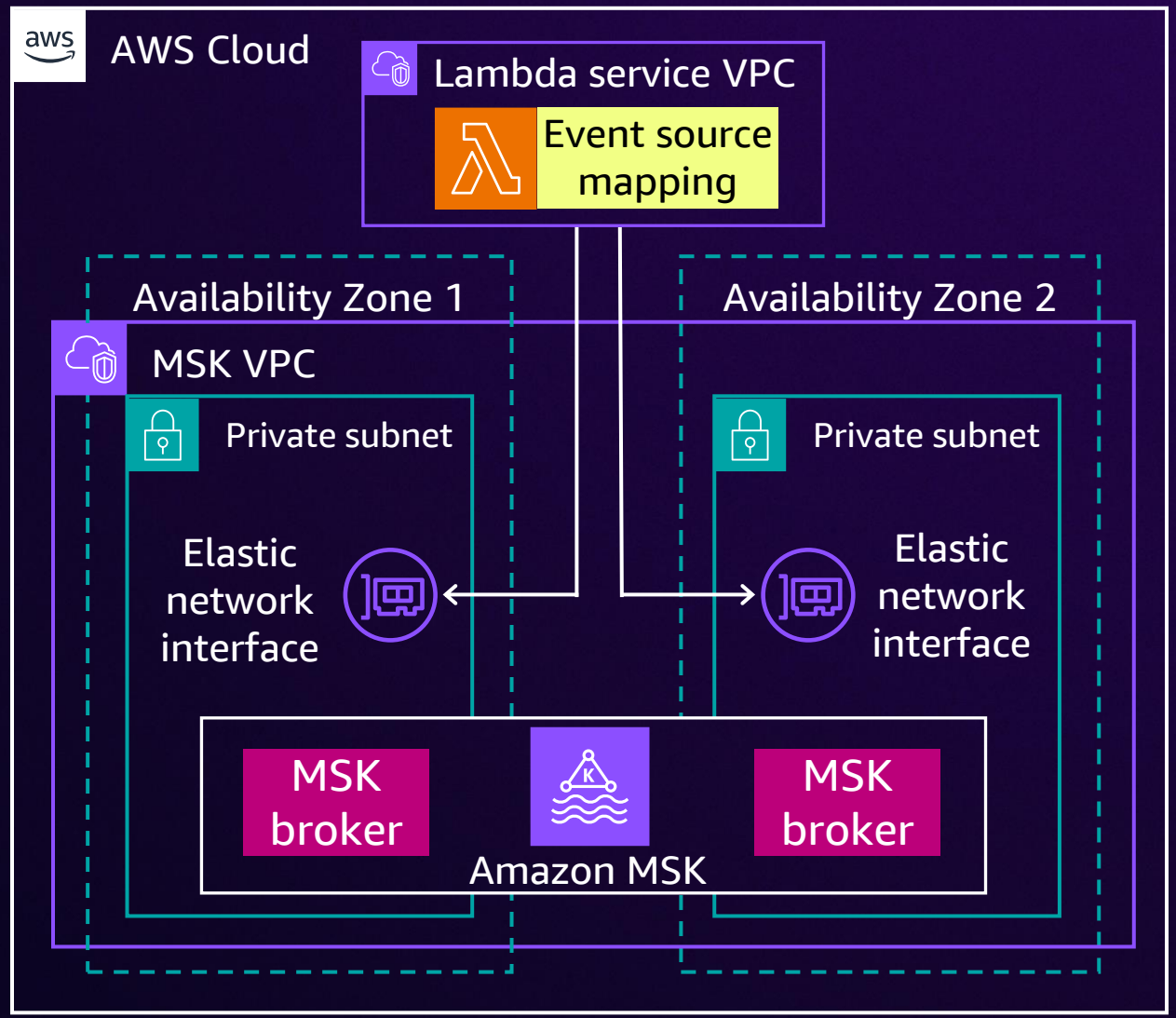
- Accesses Kafka over WAN
- Requires NAT gateway in public subnet(s)
- Configure subnets/security groups for access
- Lambda function and Amazon MSK VPC can be same or different account



Amazon MSK networking: Lambda ESM to MSK

Lambda ESM connectivity to Amazon MSK

- Doesn’t use function VPC settings
- ESM creates an ENI inside each cluster subnet
- Uses subnet/security group settings on target cluster
- Security group rule must grant self inbound/outbound access



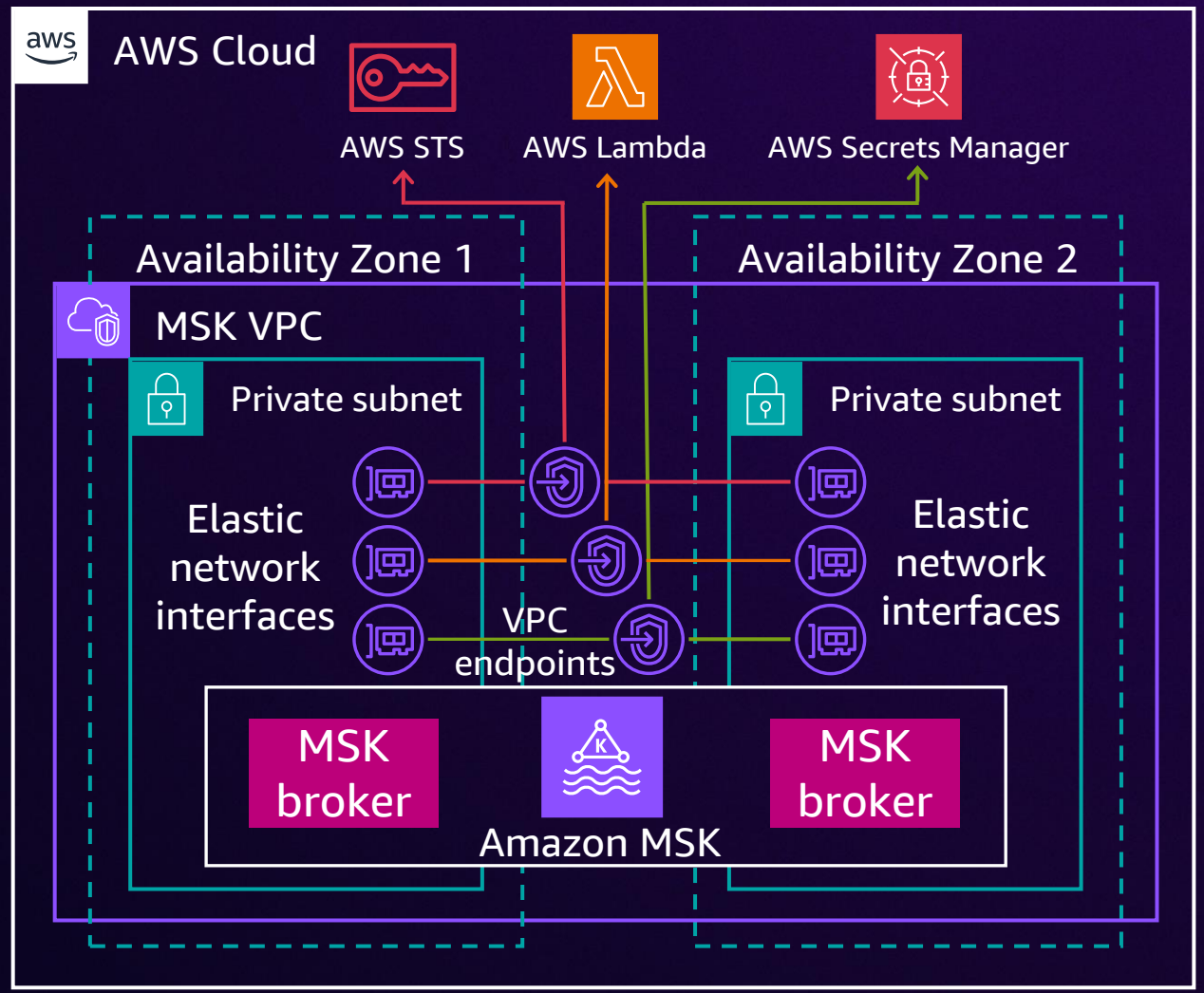
Amazon MSK networking: VPC outbound

MSK VPC outbound connectivity

On-Demand

VPC endpoint option

- MSK VPC requires outbound connectivity to
  - AWS Lambda
  - AWS Security Token Service (AWS STS)
  - AWS Secrets Manager



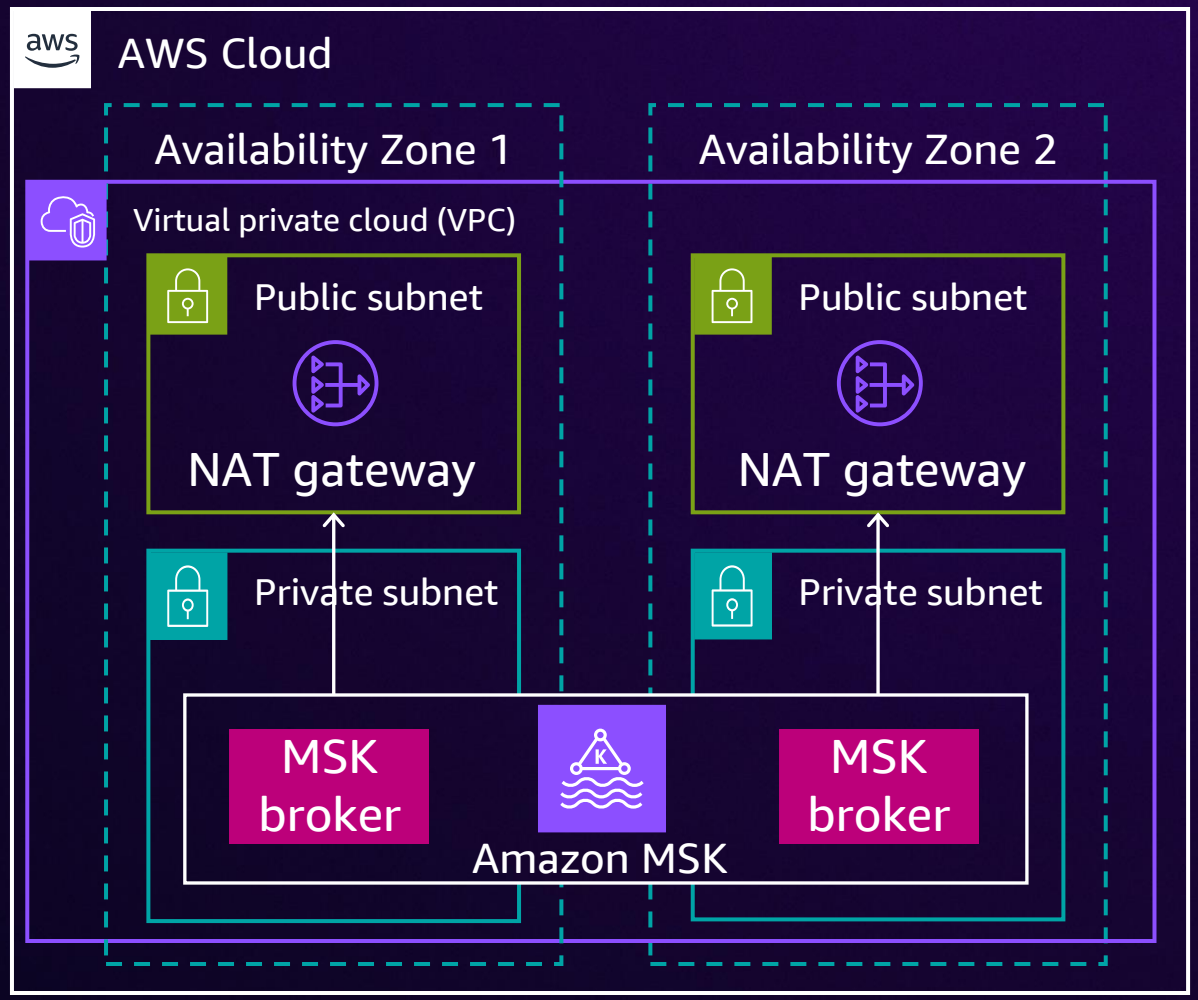
Amazon MSK networking: VPC outbound

MSK VPC outbound connectivity

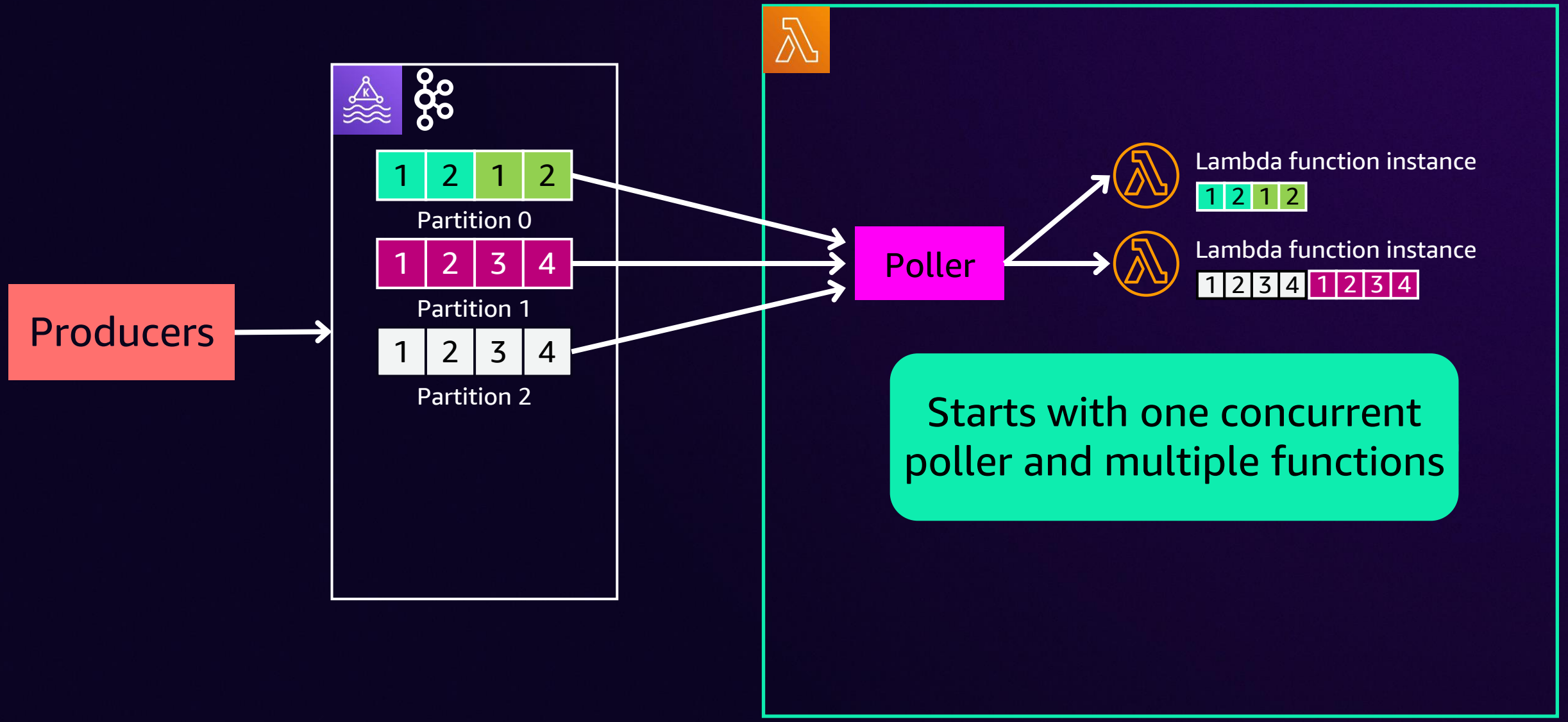
On-Demand

NAT gateway option

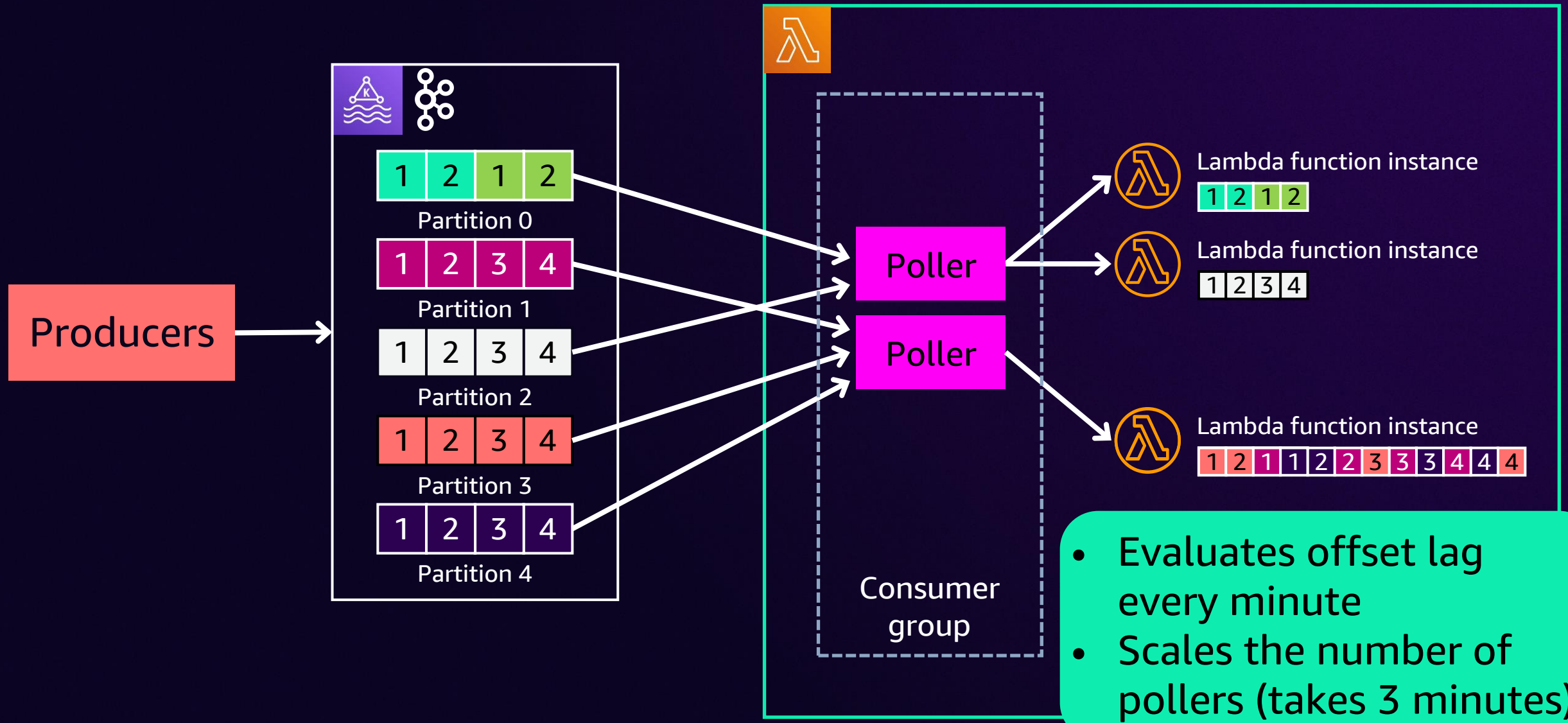
- MSK VPC requires outbound connectivity to
  - AWS Lambda
  - AWS STS
  - AWS Secrets Manager



Amazon MSK to Lambda: Scaling

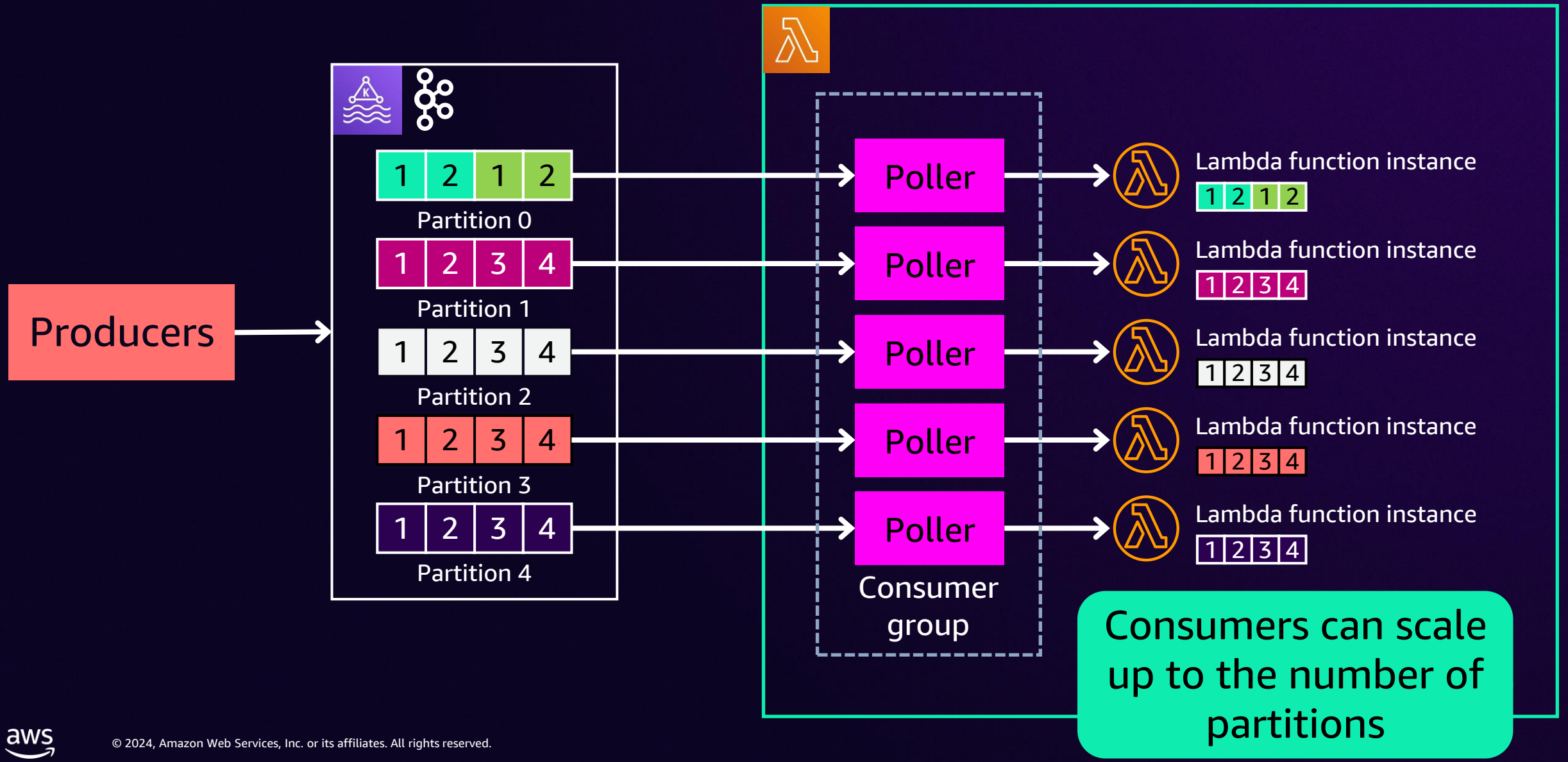


Amazon MSK to Lambda: Scaling

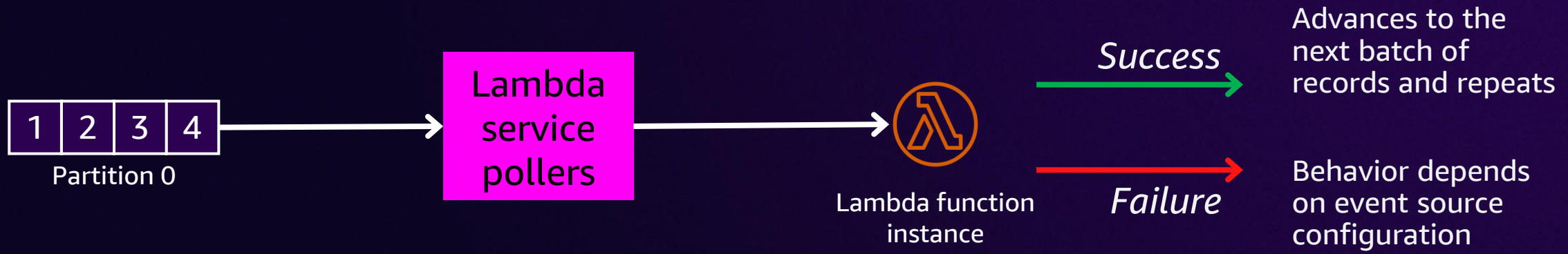


- Evaluates offset lag every minute
- Scales the number of pollers (takes 3 minutes)

Amazon MSK to Lambda: Scaling



Partition processing



Default settings

- Invokes the function with the same batch of records
- Repeat until it succeeds or the records age out of the stream

Failure destination

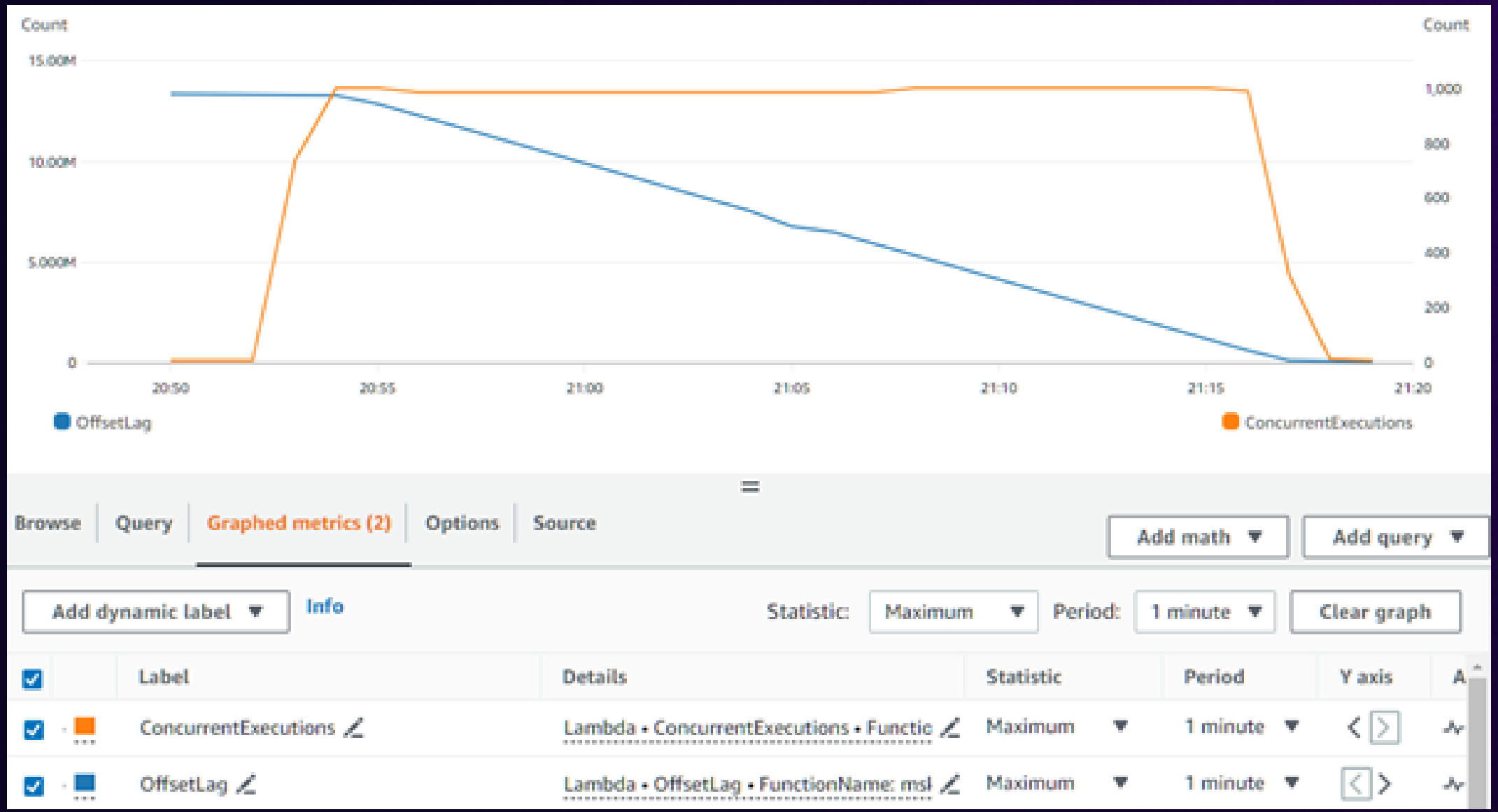
- Send batch to the configured failure destination
  - SQS/SNS: metadata
  - S3: invocation record
- If no failure destination is configured, they are discarded

New Lambda and Kafka features: 2023-2024

- Scaling
  - Faster scale-up and slower scale-down
  - Provisioned Mode
- Availability
  - On-failure destinations for Kafka ESMs
- Configuration
  - Starting position timestamp for Kafka
  - ESM filtering enhancements and filter criteria encryption
  - Cross account support for MSK
  - Provisioned Mode simplified networking

Managing performance

Kafka scaling

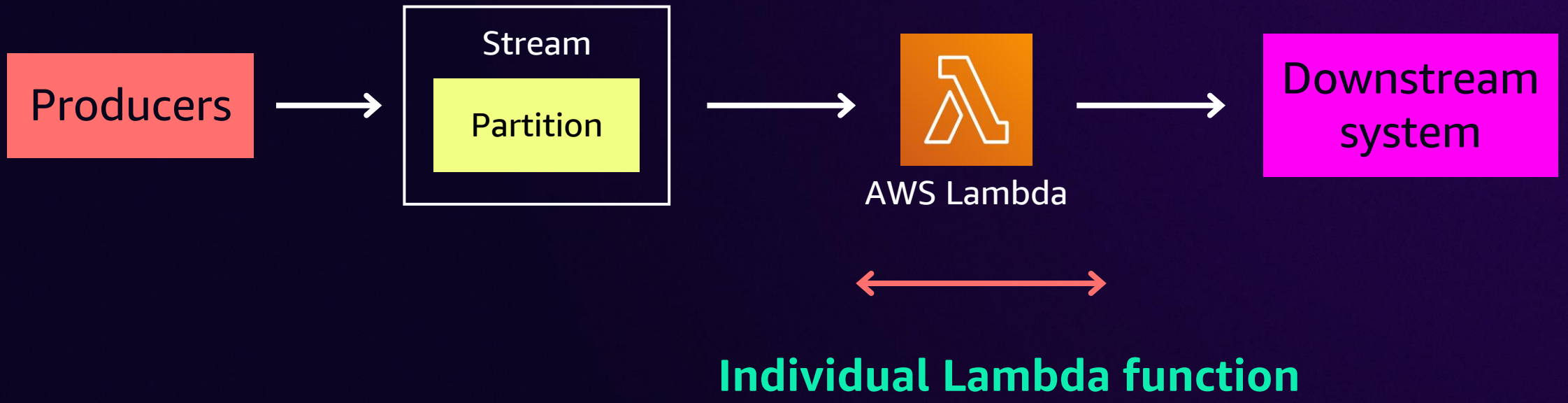


Provisioned Mode

NEW

- Provision event pollers in advance to handle sudden spikes in traffic
- For applications with stringent performance requirements
- Controls to optimize throughput (up to 5 MB/s throughput)
  - minimum event pollers (1-200)
  - maximum event pollers (1-2000)
- Billing unit: Event Poller Unit (EPU) - 20 MB/s of throughput
- No longer required to configure AWS PrivateLink or NAT gateway

Measuring performance



Measuring performance

Measuring performance: Understand the baseline

Stream baseline load

- Average records per second
- Average bytes per second

Lambda performance baseline

- Time to process a single record
- Time to process average batch size

Measuring performance: What's changed?

Increase in record volume per partition?

Increase in the size of the records?

Any function errors, throttles?

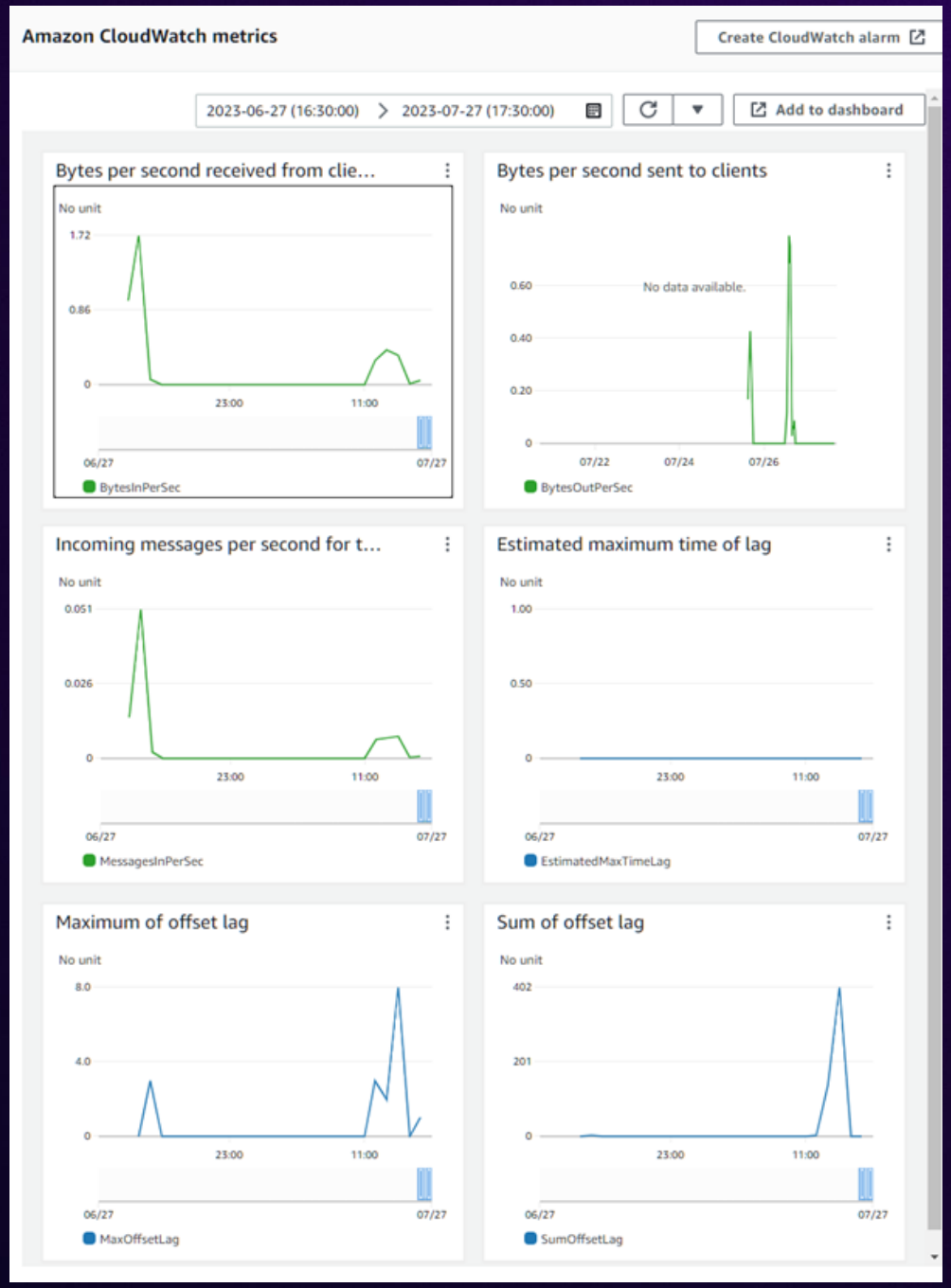
Duration, invocations, retries all affect performance

Kafka metrics/alarms

Amazon CloudWatch metrics

- Basic/default monitoring
- Enhanced monitoring
  - Per broker/topic/partition
- `consumer_lag`
- `consumer_offset`

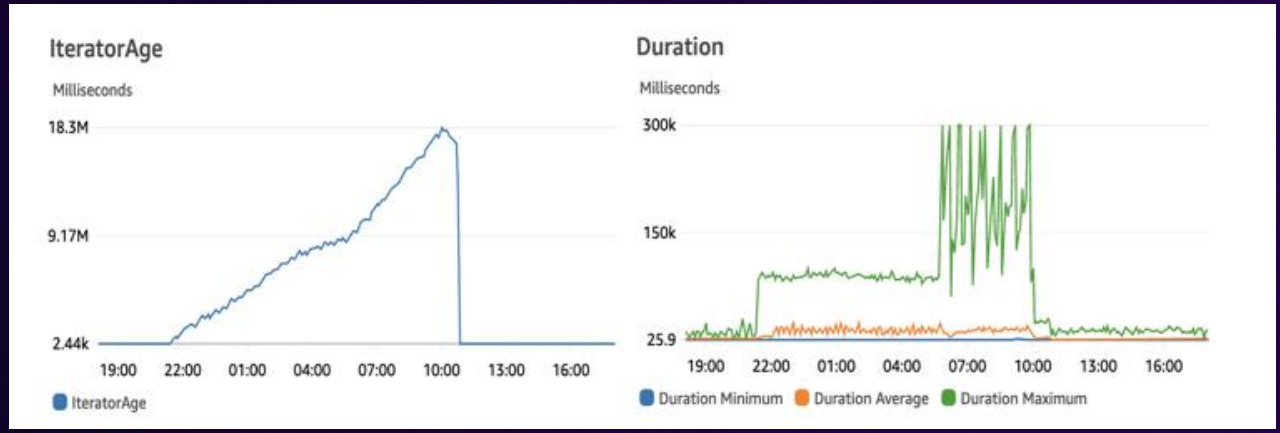
Open monitoring with Prometheus

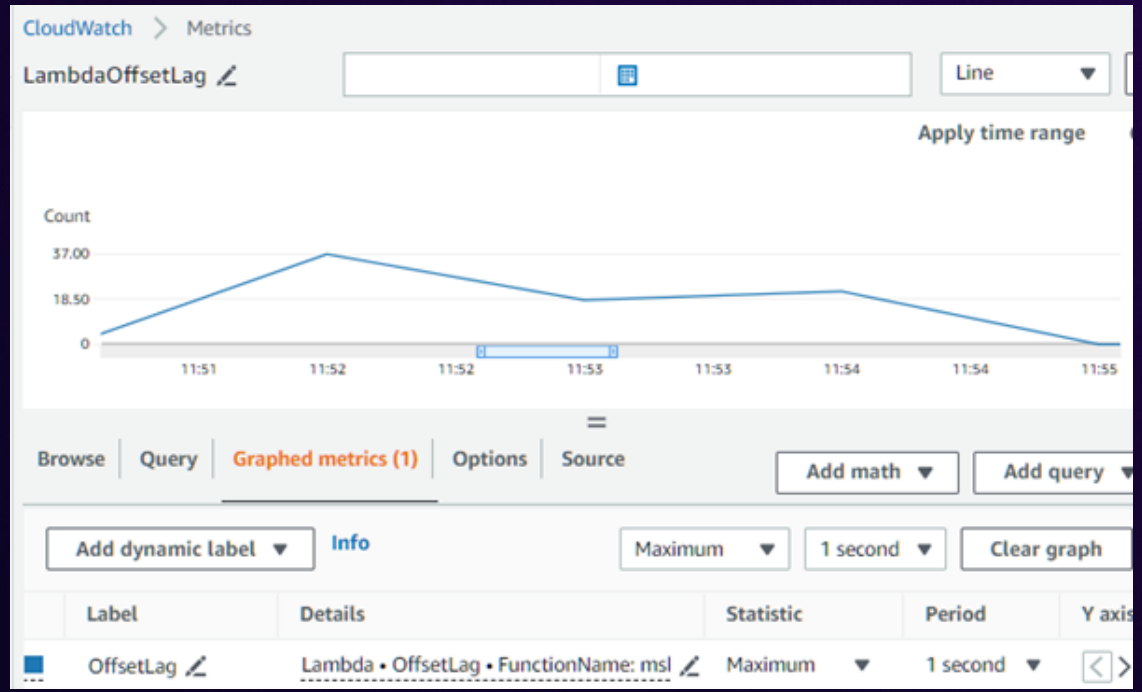


Lambda metrics/alarms

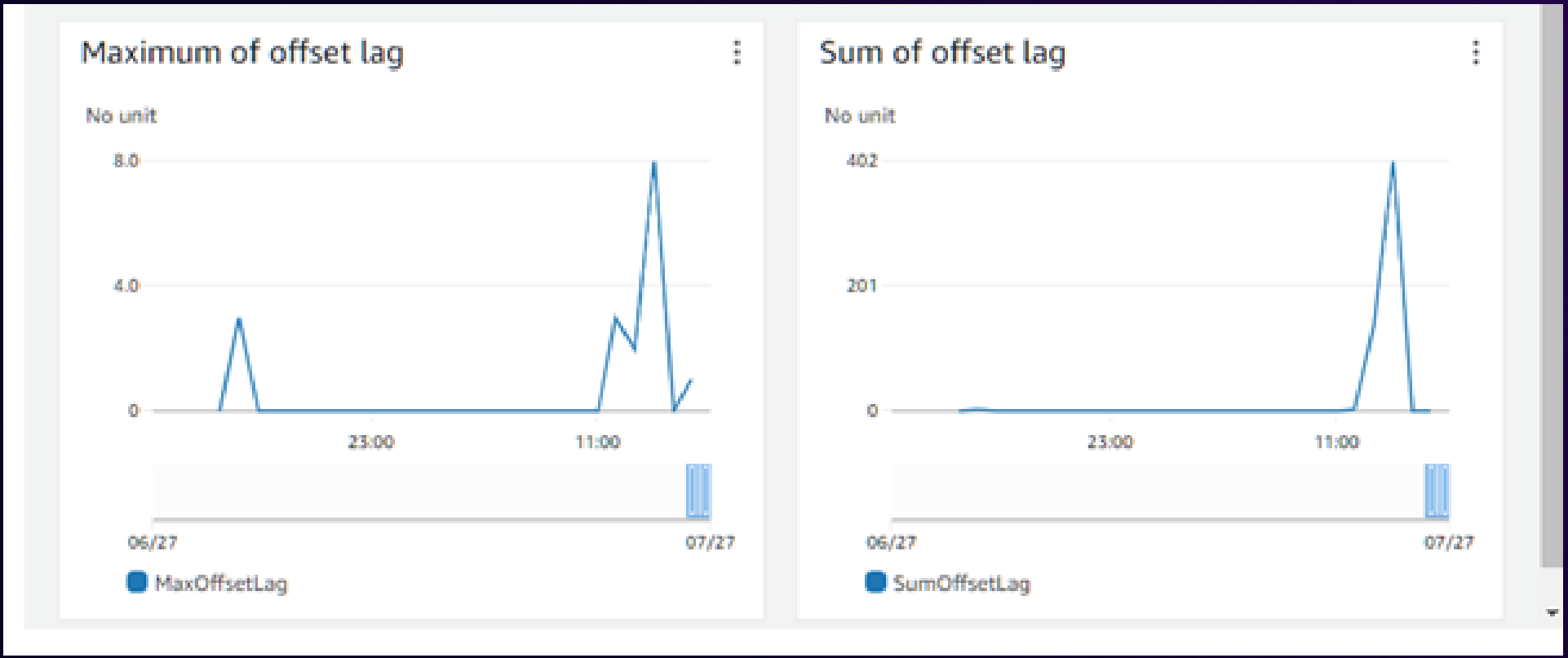
Amazon CloudWatch metrics

- Invocations
- Duration
- Error count
- Throttles
- Concurrency
- OffsetLag (Kafka)





Offset lag growing

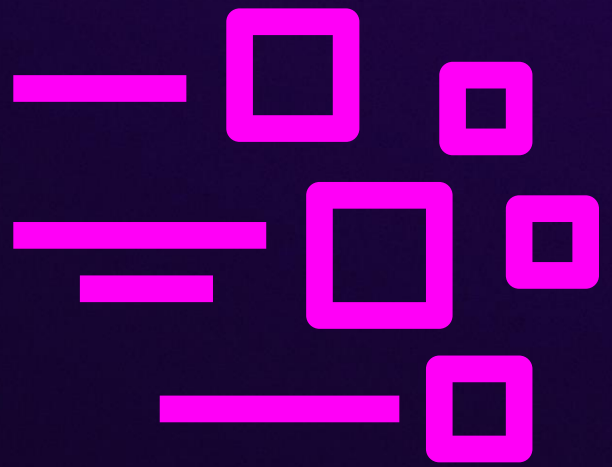


Increasing processing throughput: Lambda

- Use filtering
- Increase the function memory allocation
- Optimize your function code
- Increase the batch size
- Gracefully handle invocation errors
- Manage your concurrency and throttles

Increasing processing throughput: Stream

- Evenly distribute records using partition key
- Increase partitions to increase number of Lambda consumers
- Buffer records at the producer side



Best practices, for everyone

Powertools for AWS Lambda

Python | TypeScript | Java | .NET

- Batch processing
- Config management
- Observability
- Secrets handling
- REST/GraphQL API
- Idempotency
- Input/output validation
- BYO middleware
- Self-documented schemas
- Feature flags
- Caching
- Streaming
- Data extraction

**feature set may vary across languages*

Summary

01 Understanding data streaming

02 Streaming data on AWS

03 Streaming architecture

04 Processing streaming data

05 AWS Lambda and Kafka

06 Managing performance

Resources

s12d.com/svs321-24

Check out these other sessions

API309: Building EDAs with Apache Kafka and Amazon EventBridge

- **Chalk talk:** Wednesday, Dec. 4 at 8:30 AM | Caesars Forum, Level 1, Academy 416

SVS216-R: Serverless data processing with AWS Lambda and Apache Kafka

- **Builders' sessions:**
  - Monday, Dec. 2 at 11:30 AM | Caesars Forum, Level 1, Summit 232
  - Monday, Dec. 2 at 2:30 PM | Caesars Forum, Level 1, Summit 232
  - Wednesday, Dec. 4 at 8:30 AM | Mandalay Bay, Level 2 South, Surf B

SVS406: Scale streaming workloads with AWS Lambda

- **Chalk talk:** Thursday, Dec. 5 at 4:00 PM | MGM Grand, Level 3, Premier 309

SVS401: Best practices for serverless developers

- **Breakout session:** Monday, Dec. 2 at 3:00 PM | Venetian, Level 3, Lido 3002

Continue your AWS serverless learning

Learn at your own pace

Expand your serverless skills with our learning plans on **AWS Skill Builder**

Increase your knowledge

Use our **Ramp-Up Guides** to build your serverless knowledge

Earn an AWS serverless badge

Demonstrate your knowledge by achieving **digital badges**

https://s12d.com/serverless-learning

Thank you!

Please complete the session survey in the mobile app

Julian Wood

@julian_wood

@julianwood.com

linkedin.com/in/julianrwood