ADC-C2

Modern Data Streaming Architecture on AWS

Lydia Ray (she/her)

Sr Analytics Solution Architect
Amazon Web Services

Veljko Kokanovic

Lead Architect
LetsGetChecked

In every minute of the day . . .

Instagram users share

66K

photos

Tinder users swipe

1.1M

times

Email users send

231.4M

messages

Source: https://www.domo.com/data-never-sleeps#

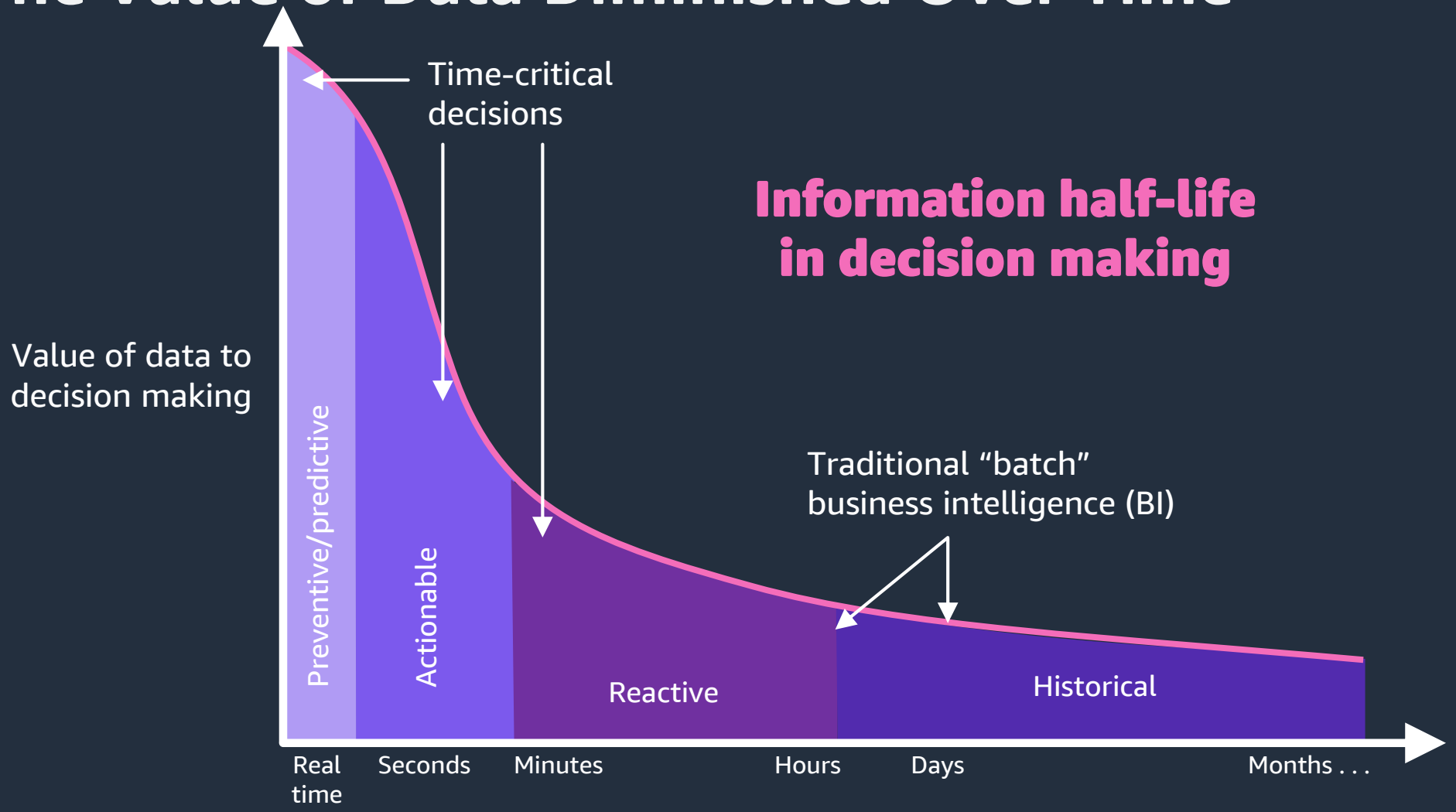
Agenda

- Why Data Streaming?
- Modern Data Streaming Architecture on AWS
- Streaming Architecture Patterns
- Data Streaming: The Catalyst for Transforming

  Microservice Architecture in Healthcare

Why Data Streaming?

The Value of Data Diminished Over Time



Source: Perishable Insights – Stop Wasting Money On Unactionable Analytics, Mike Gualtieri, Forrester (bit.ly/2NHJ840)

Common Data Streaming Use Cases

Anomaly and fraud detection

Nourishing marketing campaigns

Tailoring customer experience in real time

Real-time personalization

Empowering IoT analytics

Supporting healthcare and emergency services

Challenges of Data Streaming

Difficult to set up

Tricky to scale

Hard to achieve high availability

Integration requires development

Error prone and complex to manage

Expensive to maintain

Modern Data Streaming Architecture

5 Logical Layers Composed of Purpose-Built Components

Source

Devices or applications that produce real-time data at high velocity

Stream ingestion

Data from tens of thousands of data sources can be collected and ingested in real time

Stream storage

Data is stored in the order received for a set time and can be replayed indefinitely during that time

Stream processing

Records are read in the order they're produced, allowing for real-time analytics or streaming ETL

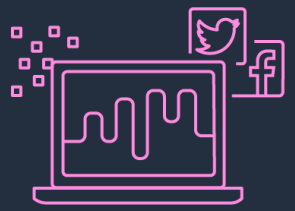
Destination

Data lake
Data warehouse
Database

OpenSearch Event driven Applications

Data Sources

Mobile apps

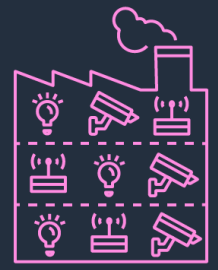


Web clickstream/social

Application logs

IoT sensors

Connected products



Smart buildings

Stream ingestion

DATA FROM TENS OF THOUSANDS OF DATA SOURCES CAN BE COLLECTED AND INGESTED IN REAL TIME

AWS Toolkits/Libraries

AWS SDK

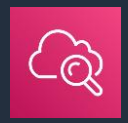
Amazon Kinesis Producer Library

Amazon Kinesis Agent

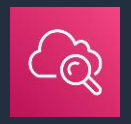
AWS Mobile SDK
IOS, Android, Fire

AWS Service Integrations

AWS IoT



Amazon CloudWatch logs



Amazon CloudWatch events

Amazon DynamoDB

AWS DMS*

Amazon MSK Connect

Third-Party Offerings

LOG4J

Apache Flume

Fluentd

*AWS DMS includes 8 on-premises databases, 1 Microsoft Azure database, 5 Amazon RDS/Amazon Aurora database types, and Amazon S3.

Stream Storage

DATA IS STORED IN THE ORDER IT WAS RECEIVED FOR A SET DURATION OF TIME AND CAN BE REPLAYED INDEFINITELY DURING THIS TIME

Amazon Kinesis Data Stream

Collect and store data streams for analytics

Amazon Managed Service for Apache Kafka

Collect and store data in Kafka topics for analytics

Which Streaming Storage to Use?

	Kinesis Data Streams Provisioned	Kinesis Data Streams On-Demand	Amazon MSK Provisioned	Amazon MSK Serverless
Infrastructure provisioning	Number of Shards to be calculated as per ingestion & consumption pattern.	Shards are automatically added/removed as per ingestion traffic & upper limit.	Broker instance class & cluster size to be calculated as per ingestion & consumption pattern.	No need to specify broker instance class or cluster size. Managed by the service.
Throughput	A single shard can : • Ingest 1 MB/sec or 1k rec/sec. • Retrieve 10 MB or 10k rec/call with 5 TPS.	The stream by default can : • Ingest 4 MB/sec or 4K records/sec. • Retrieve at least 2x the ingestion quota. • Can scale to 1 GB/s write and 2GB/s read capacity through support ticket.	Throughput depends on choice of broker instance class and spread of partitions.	A cluster by default can : • Ingest 200 MB/s • Retrieve 400 MB/s
Scalability	Requires API driven addition/removal of shards in seconds.	Accommodates up to 2x of previous peak write throughput observed in the last 30 days. Can scale to 1 GB/s write and 2GB/s read capacity through support ticket.	Provisioned cluster scales in minutes. Supports scaling out broker nodes and scale up of compute and storage.	Up to 5 MB/s of write capacity and 10 MB/s of read capacity per partition is available instantly.

Which Streaming Storage to Use? (continued...)

	Kinesis Data Streams Provisioned	Kinesis Data Streams On-Demand	Amazon MSK Provisioned	Amazon MSK Serverless
Open-sourced?	No	No	Yes (AWS managed service for Apache Kafka)	Yes (AWS managed service for Apache Kafka)
Maximum Data retention	365 days	365 days	• Configurable as server property. • Depends on size of Amazon EBS volumes per broker. NEW With Tiered storage you can scale to virtually unlimited storage at a low cost.	• Unlimited retention • Max number of partitions in the cluster 2400. • Maximum storage per partition 250 GiB.
Latency	70ms (with Enhanced Fan Out)	70ms (with Enhanced Fan Out)	Low, depends on cluster provisioning and consumer performance.	Low

Stream Processing

RECORDS ARE READ IN THE ORDER THEY ARE PRODUCED, ENABLING REAL-TIME ANALYTICS OR STREAMING ETL

Amazon Kinesis

Kinesis Data Firehose

Amazon Kinesis Client Library

AWS services

Amazon Managed Service for Apache Flink

AWS Lambda

Amazon EMR

Third Party

Apache Spark

Apache STORM

DATADOG

mongoDB

anodot

databricks

Qubole

VOLTDB

SingleStore

splunk>

Which Stream Processing Technology to Use?

AWS Lambda

Simple programming interface and scaling

- Serverless functions
- Six languages
- Event-based, stateless processing
- Custom coding required for the use case
- Continuous and simple scaling mechanism

Kinesis Data Firehose

Zero administration configuration based stream processing

- Serverless applications
- Supports 20+ data sources & 15+ outputs
- Configuration driven
- Supports advanced processing with Lambda function

Amazon Managed Service for Apache Flink

Easy and powerful stateful stream processing

- Serverless applications
- Apache Flink
- Stateful processing with automatic backups
- Stream operators make building app easy

Amazon EMR

Flexibility and choice for your needs

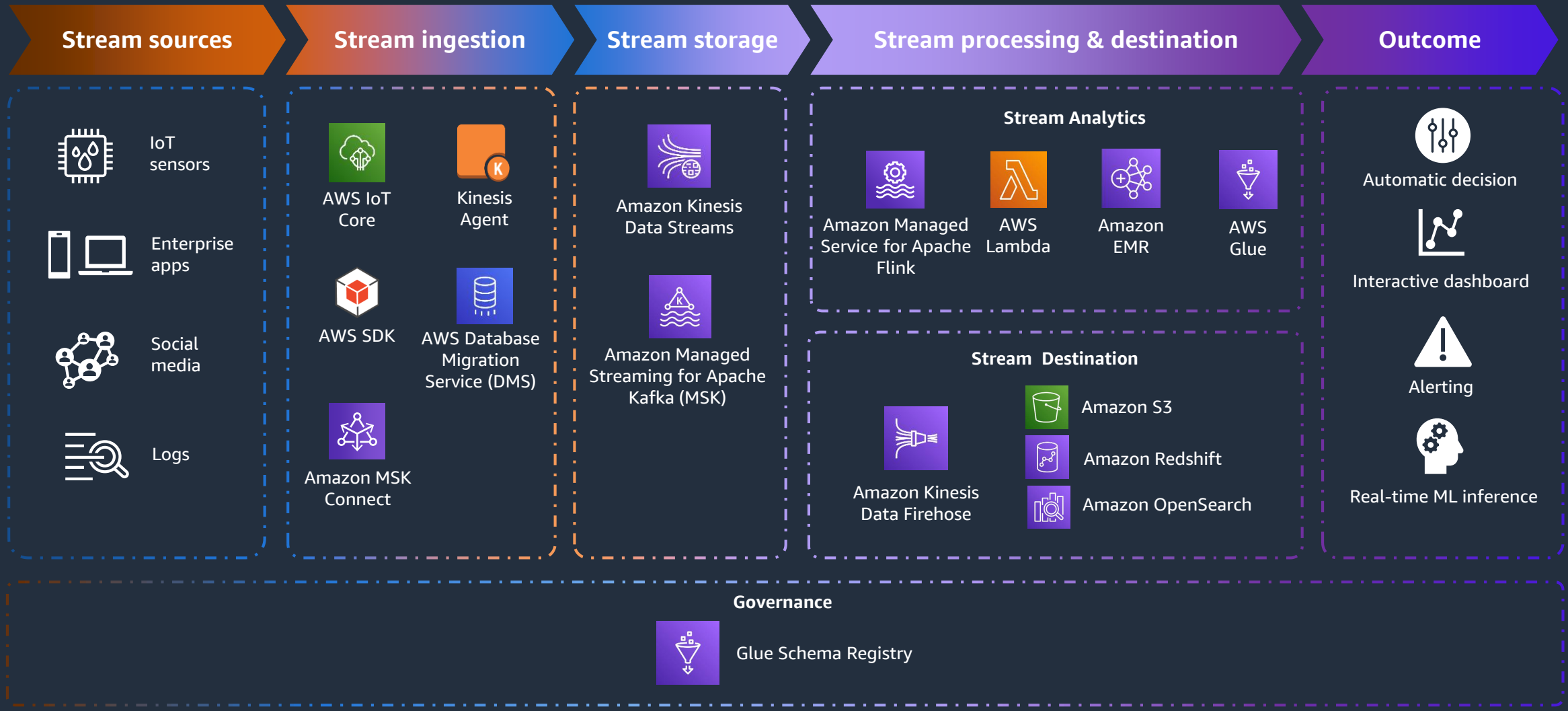
- Choose your instances
- Use your favorite open-source framework
- Fine-grained control over cluster, debugging tools, and more
- Deep open-source tool integrations with AWS

AWS Glue

Serverless data integration

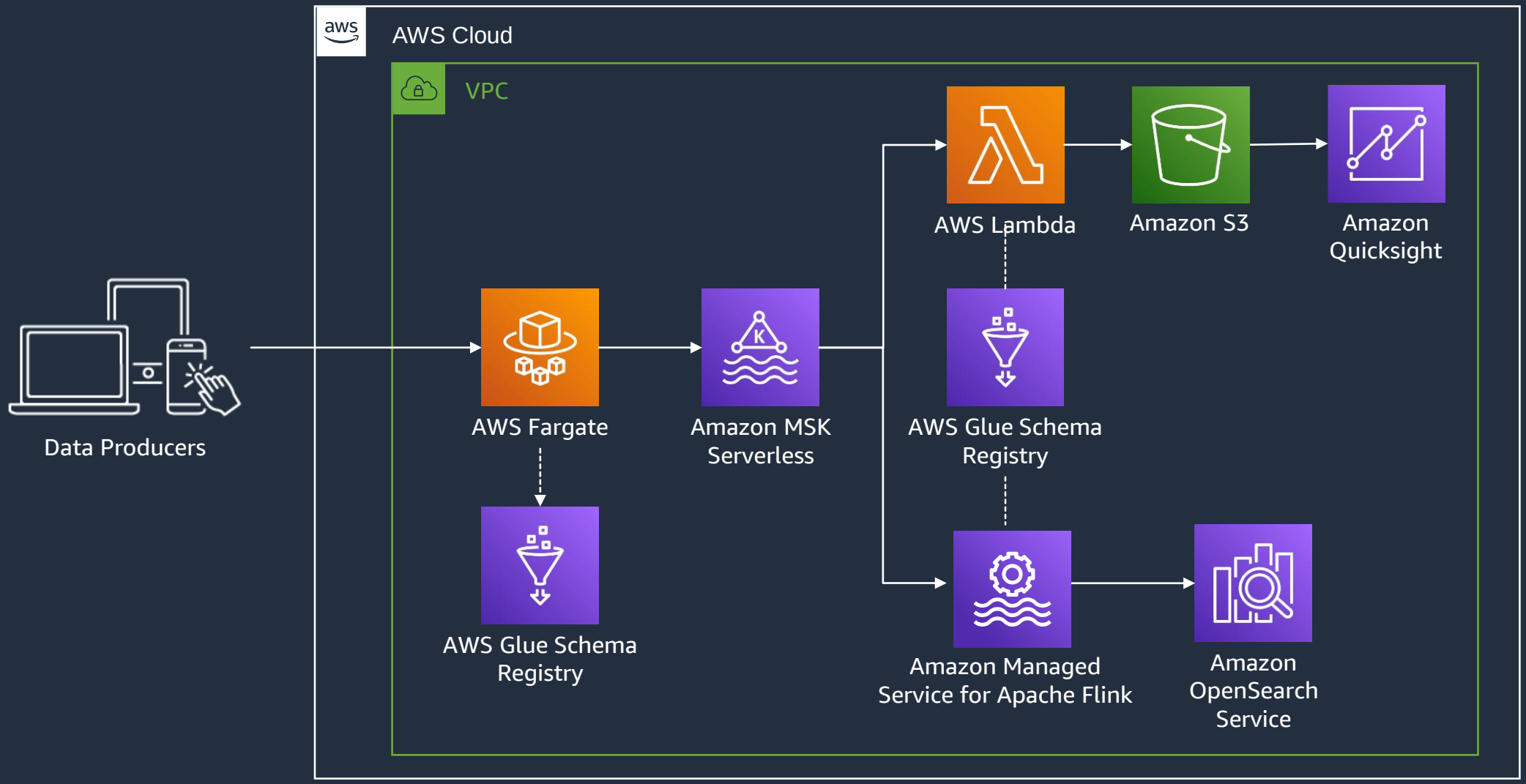
- Choose if you are already using AWS Glue or Apache Spark
- You need to process data in batch, streaming, and event modes
- You want to build your streaming jobs visually
- Near real-time use cases – your SLA is 1 second or more

Modern Data Streaming Architecture

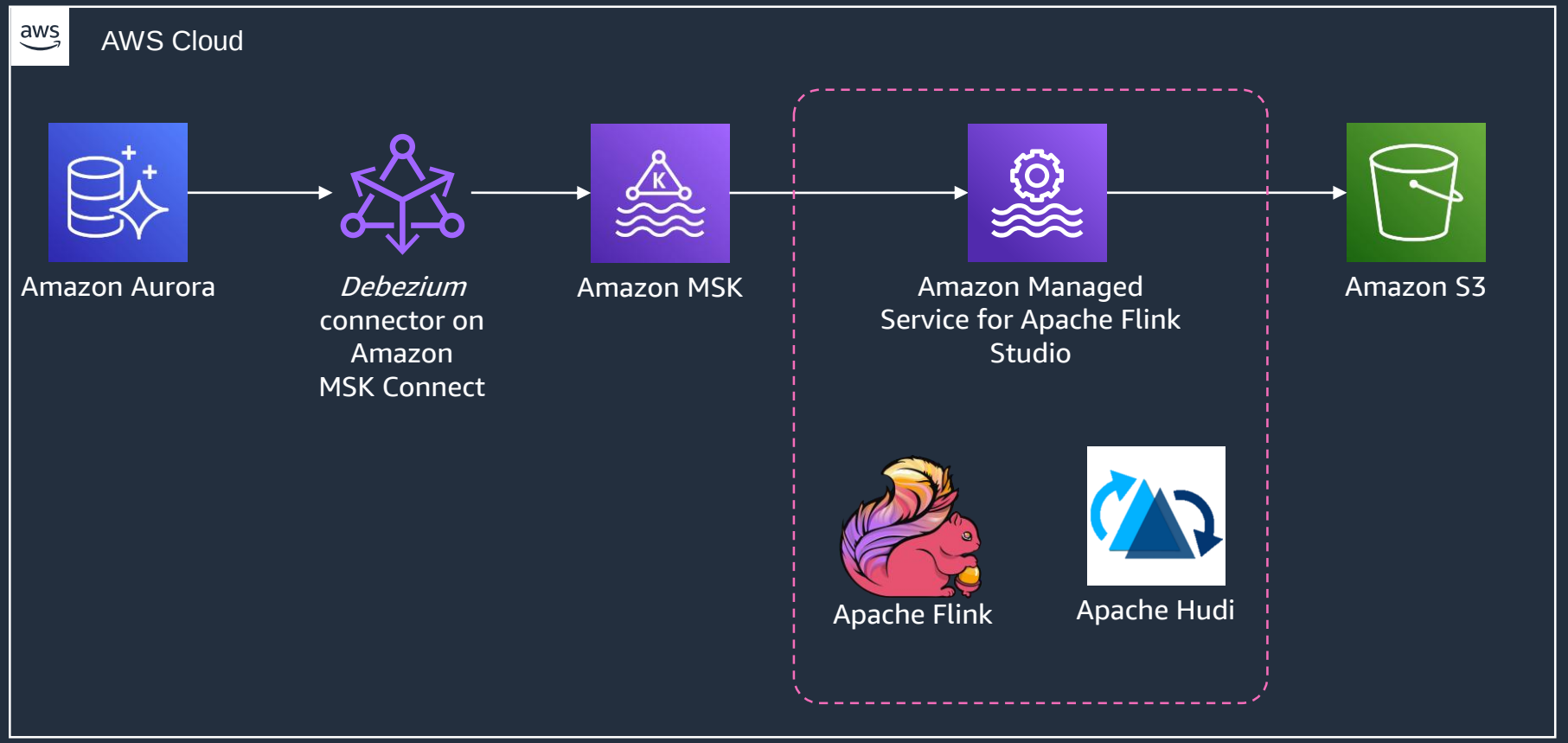


Streaming Architecture Patterns

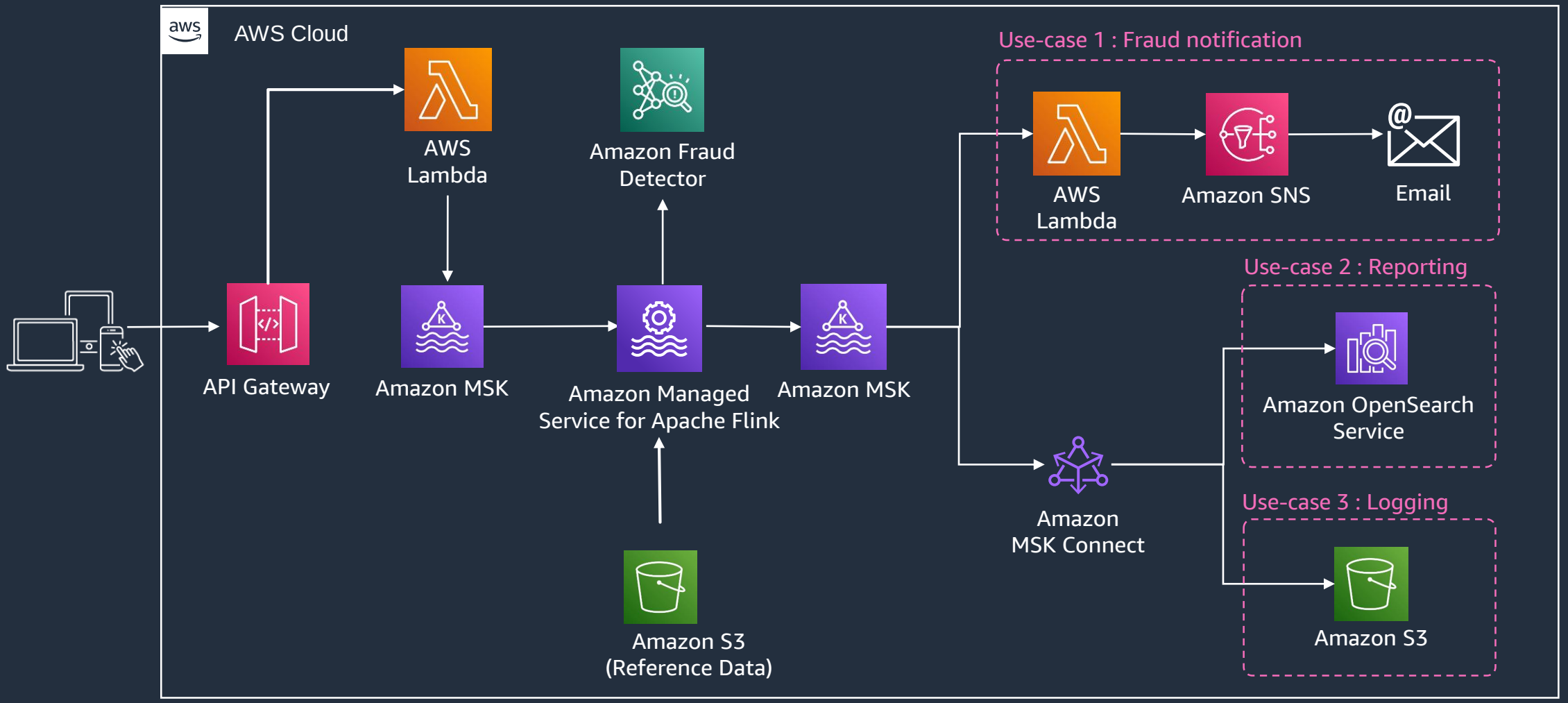
Clickstream Analytics



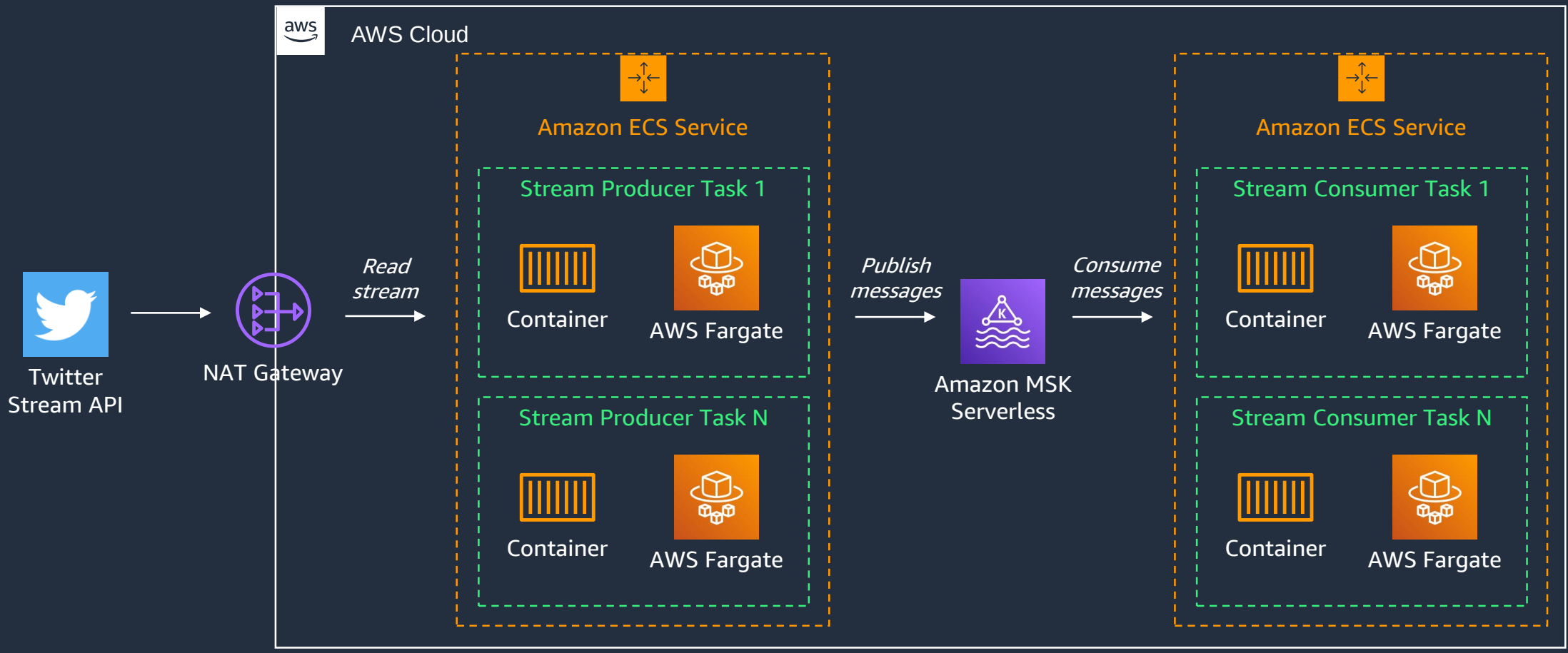
Change Data Capture



Real-Time Fraud Detection System



Microservice Architecture



Data Streaming: The Catalyst for Transforming Microservice Architecture in Healthcare

About The Speaker

Veljko [v EH l - k oh] **Kokanović**

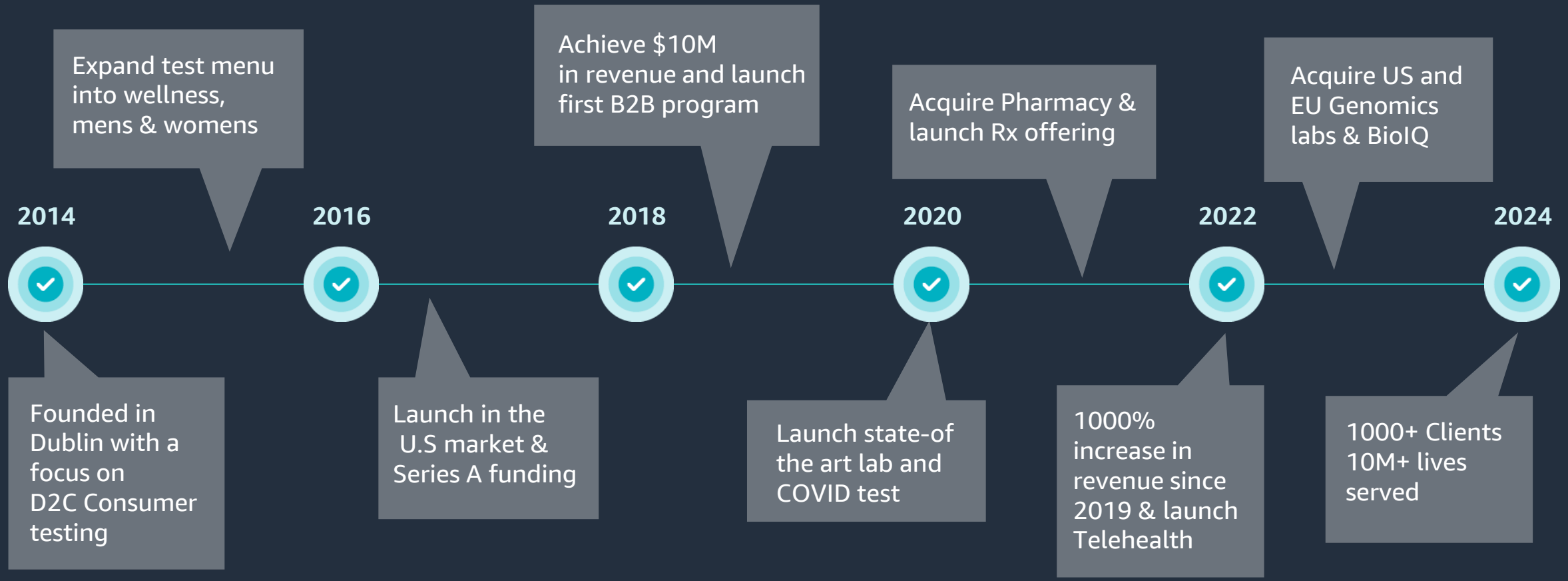
- Lead Architect at LetsGetChecked
- Past decade I spent building or migrating software platforms to Microservices, DDD, EDA and Cloud
- HealthCare, Telecoms, Online Retail, Finance

Pioneering Tomorrow's Healthcare Today

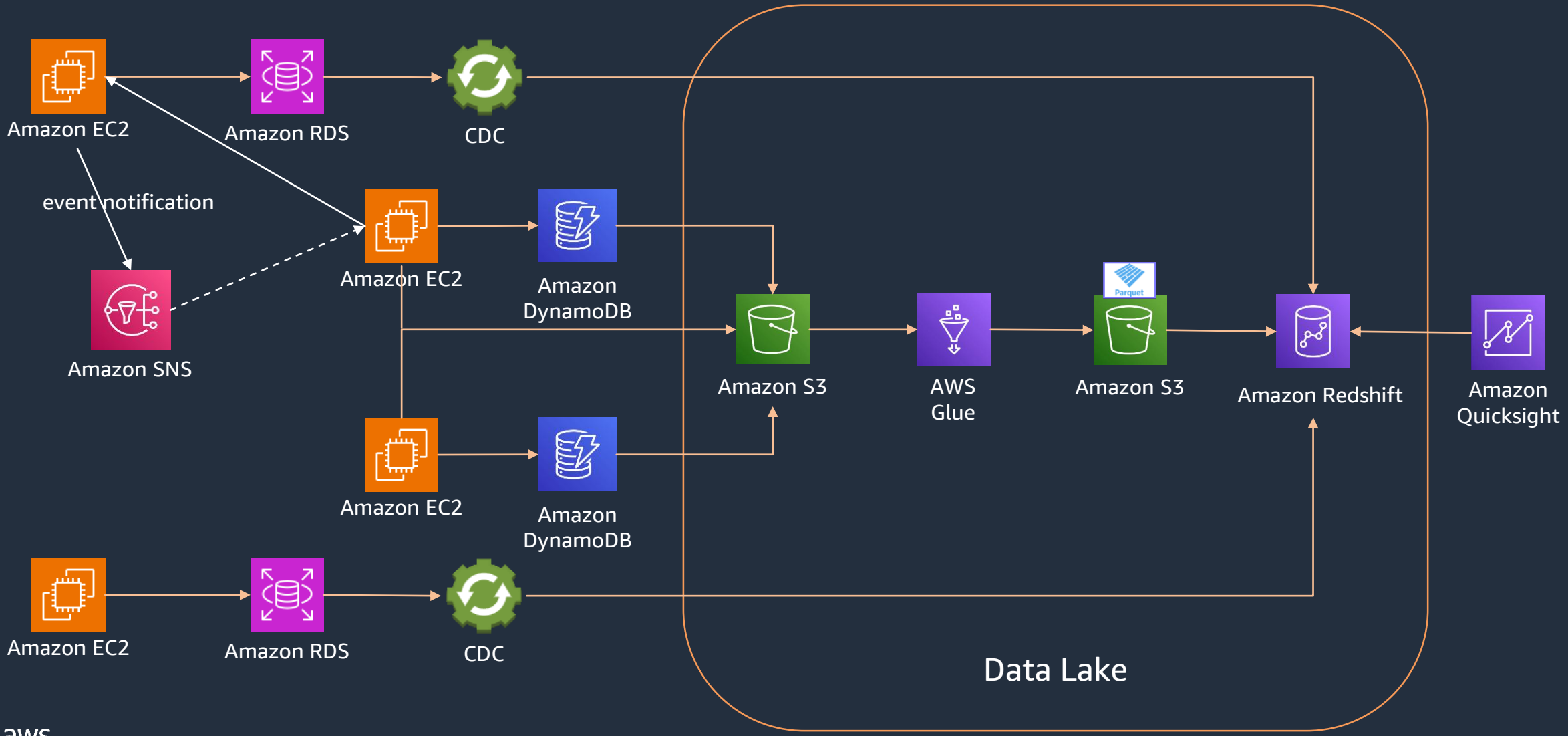
At LetsGetChecked, our purpose is to empower people to live longer, happier lives.

- 40+ at-home health tests
- Genetic sequencing
- Telehealth
- Pharmacy Services

How Our Business Model Evolved?



Monolith Days and First Microservices



Amazon Managed Service for Apache Kafka (MSK) at The Heart of It

Adopting Amazon MSK

LetsGet Checked

Technical Considerations

Compliance (HIPAA, HITRUST, GDPR)
- Data protection
- Retention periods
- Removing data
  - Crypto shredding
  - Compacted topics

Sizing and Capacity Planning
- Storage size
- Broker size
- Partition count
  - Throughput
  - Processing time

Handling large messages
- Increasing message limit
- Compression
- Chunking

Operational Challenges

Monitoring
- *MessagesInPerSec (b)*
- *KafkaDataLogsDiskUsed*
- *ZooKeeperRequestLatencyMsMean*
- *GlobalTopicCount*
- *EstimatedTimeLag (t)(p)*
- *OffsetLag (t)(p)*

Scaling
- Storage autoscaling
- Broker scaling: CpuUser + CpuSystem P95 > 0.6

Maintenance
- RF (t) >= AZ
- minISR = RF – 1

Organizational & Cultural

Skill Gap

Collaboration
- Early involvement of data engineering
- Contract first
- Design collaboration (GitHub)
- Context map (DDD)

Data Governance

Data Serialization System (AVRO)
- Schema evolution
- Compact and efficient
- Language neutrality
- Documentation
- Tooling support
- Automated contract generation

Schema Management & Evolution
- Schema registry
  - Subject name strategy
  - Compatibility mode
- Versioning strategy

Streaming Data for Microservices

Latency

- Static initialization
- Lambda runtime
- Keep it alive
- Provisioned concurrency

There are Two Hard Things in Distributed Systems

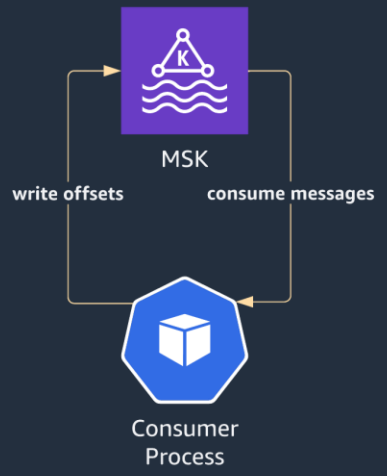
LetsGet Checked

2. Exactly-once delivery

1. Guaranteed order of messages

2. Exactly-once delivery

Stream Processing in Microservice Architecture



Data Integration

Use Cases
- ms → ms
- cqrs
- ms → data gw, bff

Exactly Once Delivery
- Idempotent producers
- Kafka transactions

Ordered delivery
- Optimistic concurrency
- Message key = event source id

Error handling
- Exception filters
- Message filters
- Ignoring messages
- Consumer idempotency

Business Process Orch

Use Cases
- Saga
- Choreography
- Policy

Message Specific Topics
- Derived from integration event topics
- Shorter retention

Idempotency
- Saga reentry
- Domain idempotency

Dead lettering
- K-Table + compacted topic
- External queuing system

General

Consumer rebalance
- Static consumer membership
- groupInstanceId = pod.name

Offset Management

Producer/Consumer Library

Transforming Data Engineering and Analytics

Retrospective and Forward-Look

LetsGet Checked

- Improve data consistency
- Support ordered delivery of events

Single Writer Principle

- Reduce direct coupling between services
- Improve service autonomy
- Rebuild system state

Stateful Events
event carried state transfer

- Simplify dependencies across the platform
- Improve scalability
- Streamline data integration scenarios
- Enforce delivery guarantees
- Support compliance and security

MSK + Schema Registry

- Improve system interoperability
- Backward data compatibility
- Simplify contract management

AVRO

Improve Data Observability

Streaming Applications

Kafka Sinks/Connectors

aws

Thank you!

Lydia Ray

https://www.linkedin.com/in/lydia-ray/

Veljko Kokanovic

https://www.linkedin.com/in/veljko-kokanovic/

Please complete the session survey