



SUMMIT
ONLINE

A N A 0 8

A path to event sourcing with Amazon MSK

James Ousby

Principal Solutions Architect
Amazon Web Services

Let's build a bank

Ousobank

My Accounts

Account

Balance

Savings

0.00

Holiday

912.00

Create Account

Learning objectives

- The what and why of event sourcing and how Amazon Managed Streaming for Apache Kafka (MSK) can help.
- Encourage customers to think differently about how they capture, store and process their data.
- Explore the design and build process behind an event sourcing demo built around MSK.

What is event sourcing?

What is event sourcing?

- What it is not - Persist entities by updating in place

invoice { id: 1, amt: 50, **status: 'issued'** }

What is event sourcing?

- What it is not - Persist entities by updating in place

invoice { id: 1, amt: 50, **status: 'overdue'** }

What is event sourcing?

- What it is not - Persist entities by updating in place

invoice { id: 1, amt: 50, **status: 'paid'** }

What is event sourcing?

- Persist entities by storing a sequence of state changing events

IssuedEvent { id: 1, amt: 50, **status: 'issued'** }



invoice { id: 1, amt: 50, **status: 'issued'** }

transient
summary

What is event sourcing?

- Persist entities by storing a sequence of state changing events

IssuedEvent { id: 1, amt: 50, **status: 'issued'** }

OverdueEvent { id: 1, **status: 'overdue'** }



invoice { id: 1, amt: 50, **status: 'overdue'** }

transient
summary

What is event sourcing?

- Persist entities by storing a sequence of state changing events

IssuedEvent { id: 1, amt: 50, **status: 'issued'** }

OverdueEvent { id: 1, **status: 'overdue'** }

PaidEvent { id: 1, **status: 'paid'** }



invoice { id: 1, amt: 50, **status: 'paid'** }

transient
summary

Event sourcing's close friends

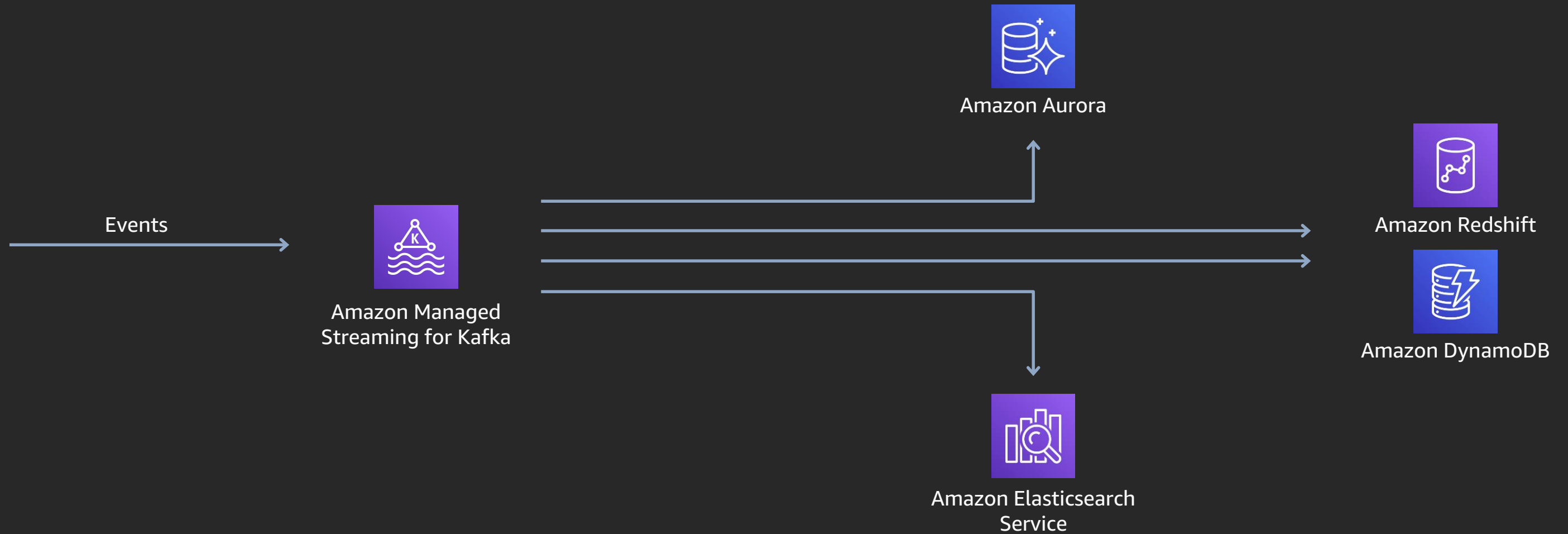
- Domain Driven Design
- CQRS (Command Query Responsibility Segregation)
- Distributed Logs (often Apache Kafka)

Why event sourcing

Future proofing your data architecture

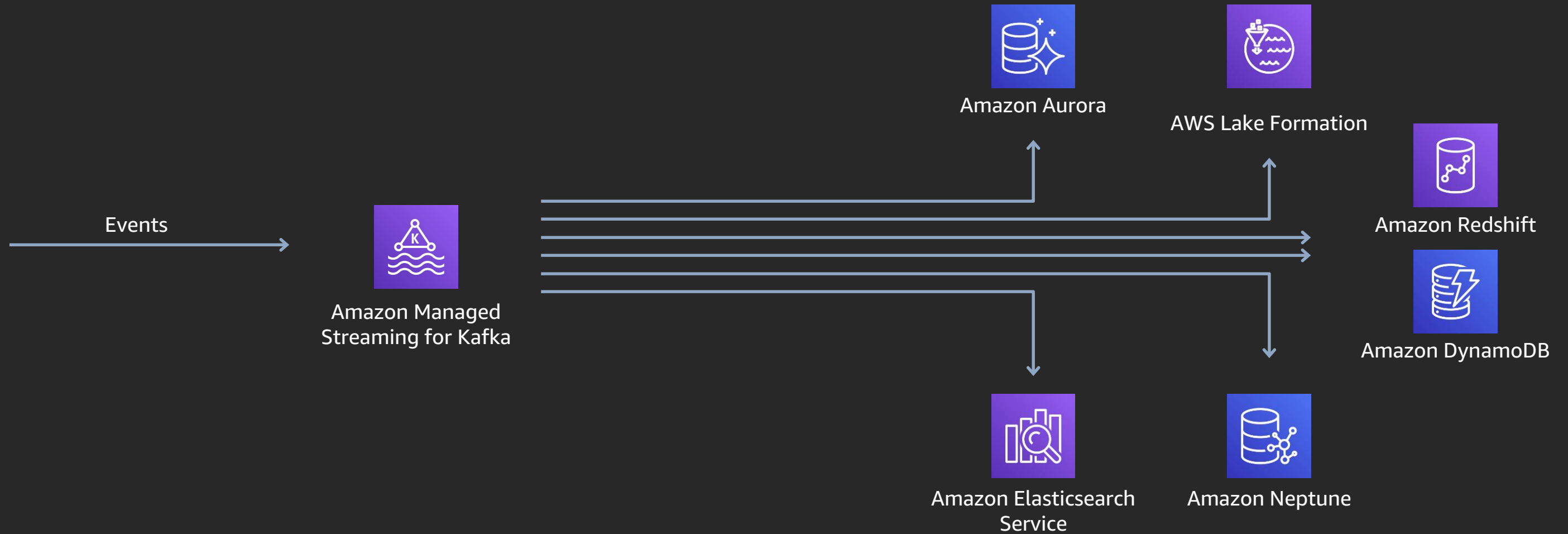
- Maintain complete history (one way and two way doors)
- Its increasingly difficult to anticipate long term data usage patterns at design/build time
- You would like your data architecture to support new patterns of consumption as technologies evolve over time

Future proofing your data architecture



transient – rebuild from event stream

Future proofing your data architecture



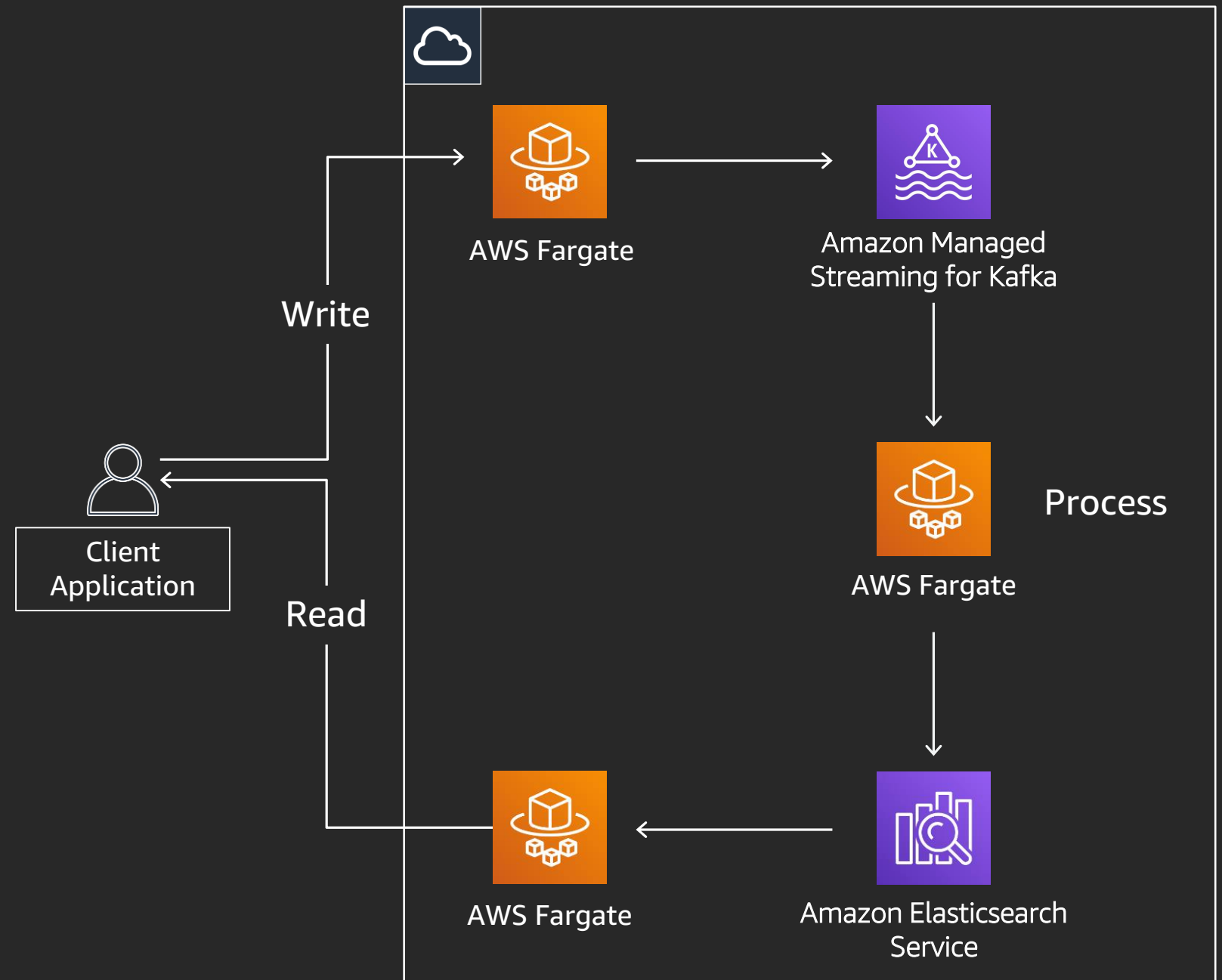
transient – rebuild from event stream

Capturing your raw data is great but...

000000000:	00	00	24	A9	65	88	84	27	C7	11	FE	B3	C7	83	08	00	\$cø?"'34m?3?█
000000010:	08	2A	7B	6E	59	B5	71	E1	E3	9C	0E	73	E7	10	50	00	█=CnY+qgr?JkzP
000000020:	18	E9	25	F7	AA	7D	9C	30	E6	2F	0F	20	00	3A	64	AA	Tdx4C>?0x/0 =dC
000000030:	CA	5E	4F	CA	FF	AE	20	04	07	81	40	00	48	00	0A	28	K^OKaR +*?0 H [K
000000040:	71	21	84	48	06	18	90	0C	31	14	57	9E	7A	CD	63	AA	q!"H+t?419M?zHc
000000050:	E0	9B	96	69	C5	18	AE	F2	E6	07	02	29	01	20	10	70	a>-1E↑Rtx=0)0 ↑p
000000060:	A1	0F	8C	BC	73	F0	78	FA	9E	1D	E1	C2	BF	8C	62	CE	9x??sprxъ?+cBY?b0
000000070:	CE	AC	14	5A	A4	E1	45	44	38	38	85	DB	12	57	3E	F6	0_9CZ=cED88:M†M>u
000000080:	E0	FB	AE	03	04	21	62	8D	F6	F1	1E	37	1C	A2	FF	75	amR*+!b?ucΔ7~0au
000000090:	1C	F1	02	66	0C	92	07	06	15	7C	90	15	6F	7D	FC	BD	~c0F2' +*§1?5o>ь?
0000000A0:	13	1E	2B	0C	14	3C	0C	00	BD	EA	6F	53	B4	98	D7	80	!!Δ+29K2 °ко8??4?
0000000B0:	7A	68	3E	34	69	20	D2	FA	F0	91	FC	75	C6	00	01	18	zh>4i Тър'ьuЖ Et
0000000C0:	C0	00	3B	9A	C5	E2	7D	BF	E1	FF	87	02	00	E0	84	E5	A ;?Ev>Y6я÷0 a"e
0000000D0:	E3	C9	8B	3B	F2	E4	41	14	40	04	DB	DE	2B	8A	F8	5D	гH<;тдA9E2+MN+?mJ
0000000E0:	CA	41	29	87	3D	29	78	50	15	38	7E	F1	5B	B5	BB	40	Ka>÷=>xP58~c [4>0

Event sourcing + CQRS benefits

- Data architectures that can scale elastically as demand increases
- Data architectures that can prioritize read/write performance
- Compute components can be scaled independently



Event sourcing challenges

Its not a new concept. Why isn't this the default way to think about data?

- Platform and tooling maturity
- Lack of framework support
- More complex / more moving parts

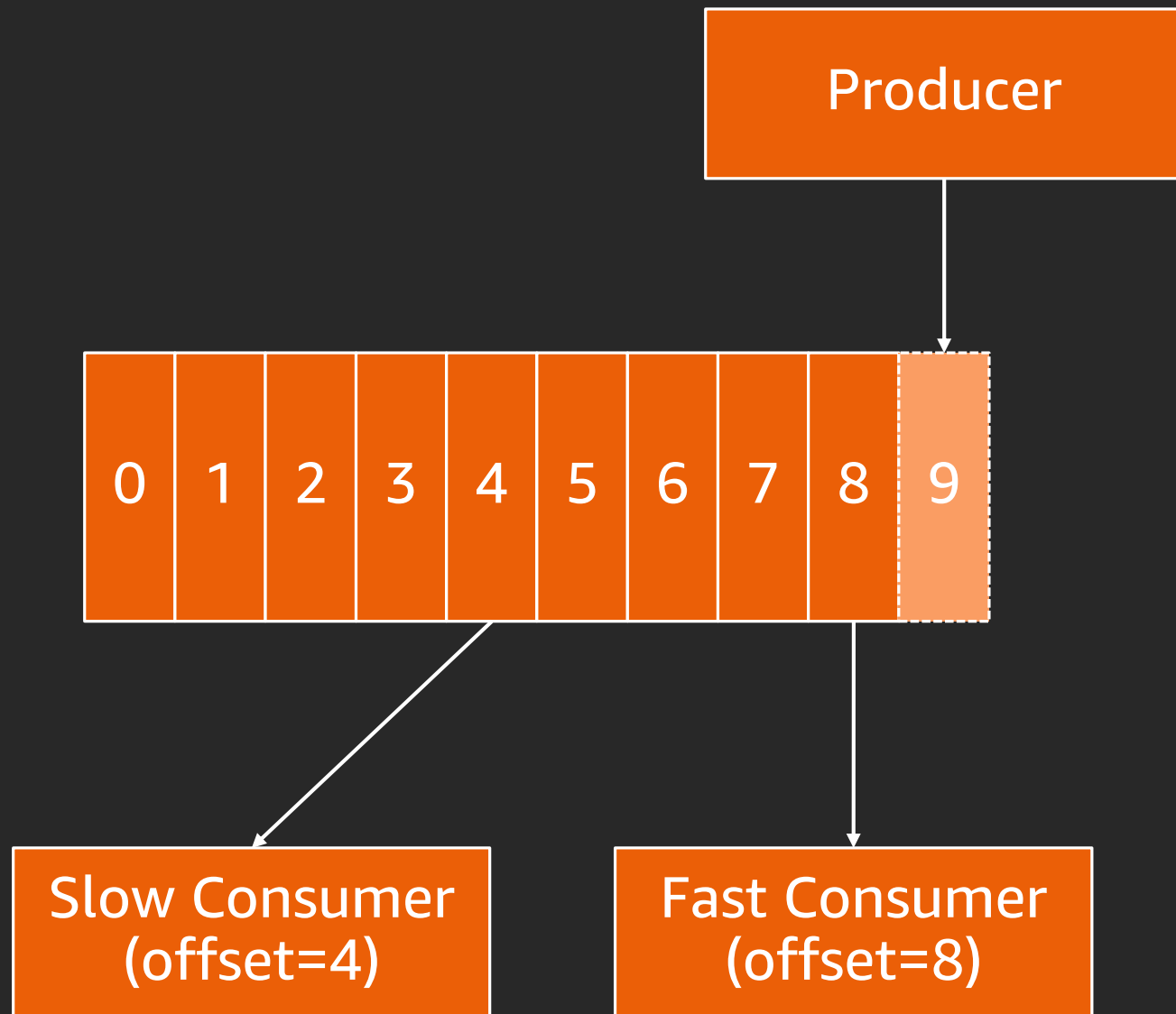
Amazon Managed Streaming for Apache Kafka

Apache Kafka

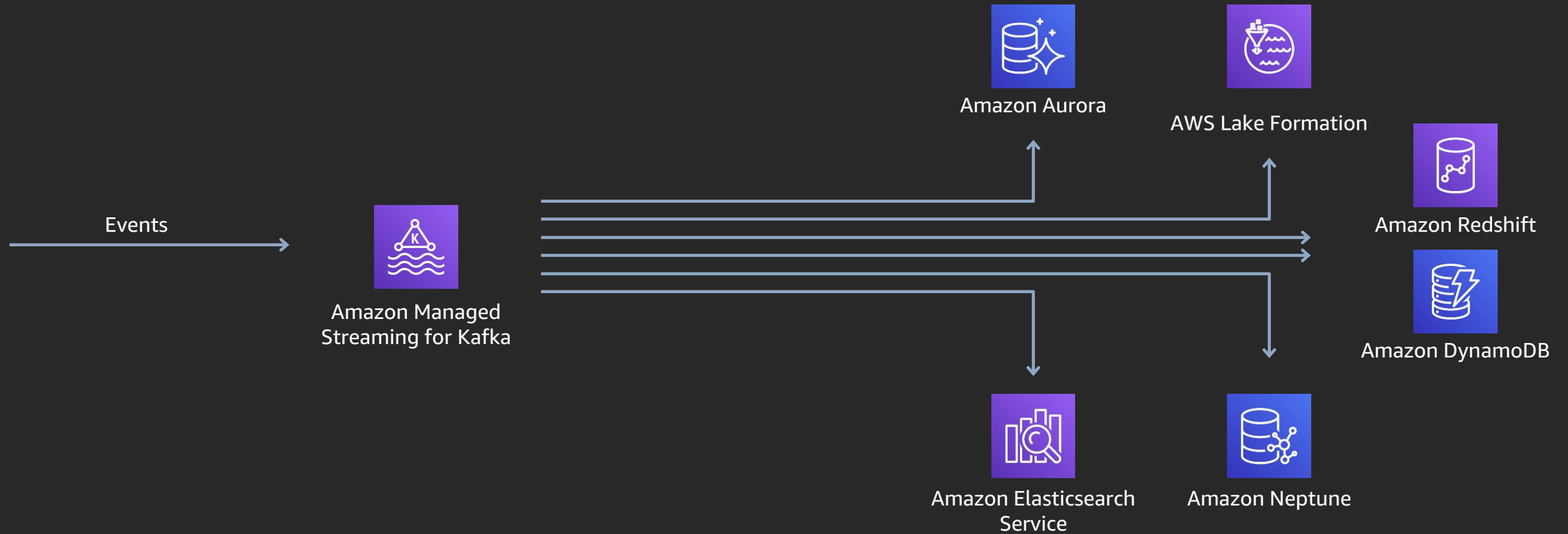


Why is Apache Kafka a good fit for event sourcing?

- Immutable append only log
- Guaranteed ordering



Multiple consumers all at different log offsets



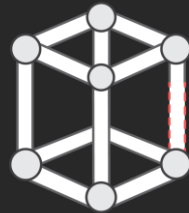
transient – rebuild from event stream

Challenges operating Apache Kafka



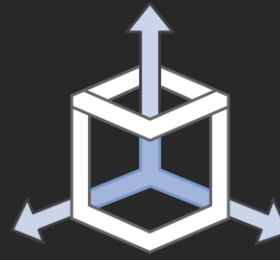
Setup

Difficult to
setup



Availability

Hard to
achieve



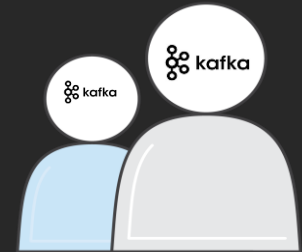
Scale

Hard to
scale



Monitoring

Hard to capture
metrics



Support

Labor
intensive

Amazon Managed Streaming for Apache Kafka

Fully managed and highly available Apache Kafka service



Fully compatible

Migrate and run your existing Kafka applications on AWS without code changes



Fully managed

Manages provisioning, configuration, and maintenance of Kafka clusters



Highly available

Multi-AZ replication within an AWS Region

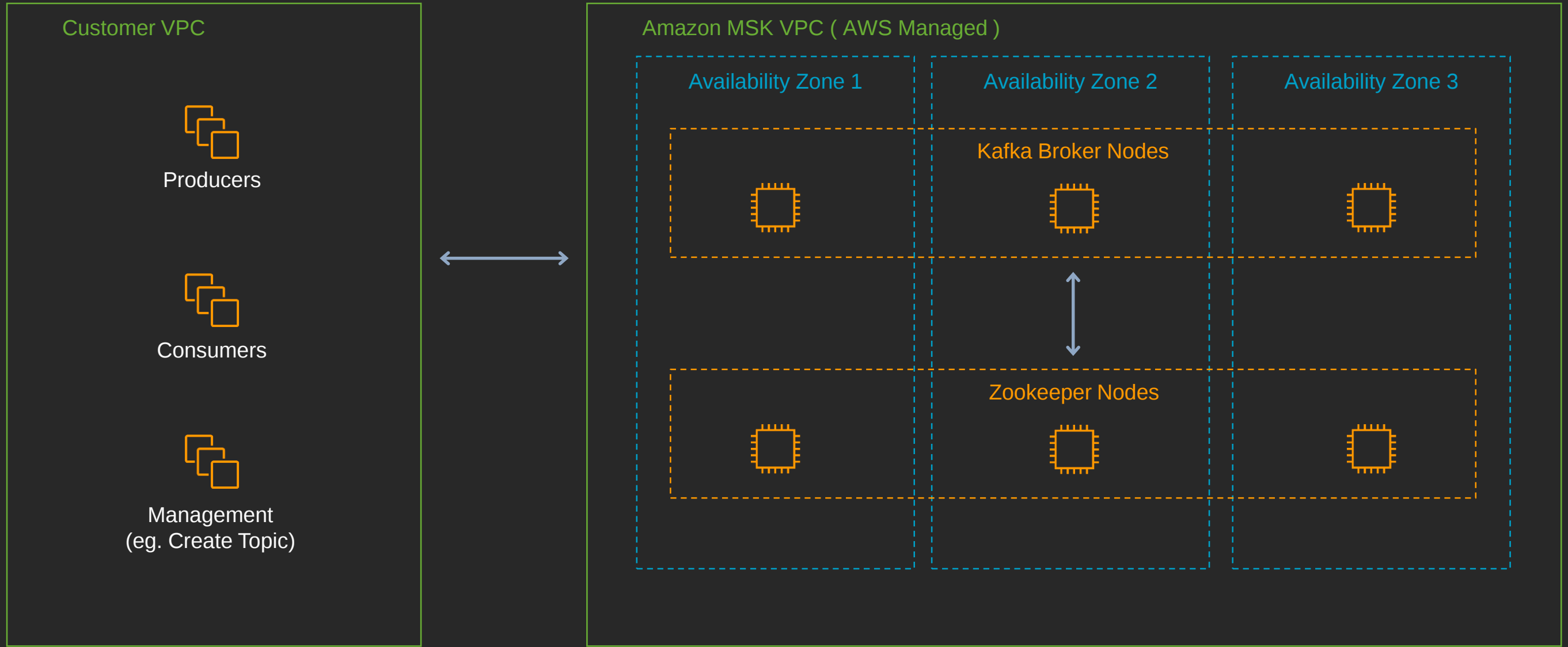
Health monitoring



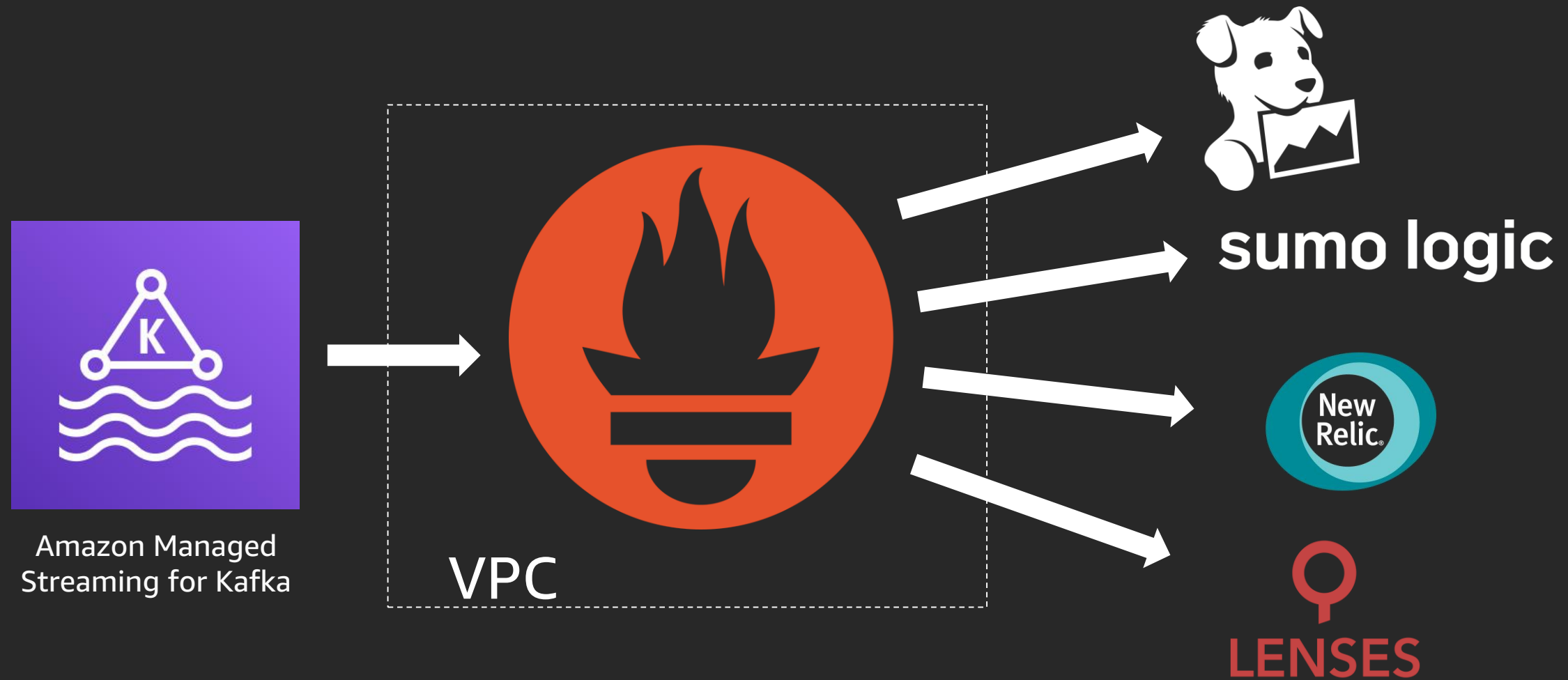
Highly Secure

Multiple levels of security for your Kafka clusters

MSK deployment architecture



Open monitoring with Prometheus



Getting started with Amazon MSK is easy

- Fully compatible with Apache Kafka 1.1.1, 2.2.1 and 2.3.1
- AWS Management Console and AWS API for provisioning
 - Clusters are automatically set up
 - Provision Kafka brokers and storage
 - Create and tear down clusters on demand
- Deeply integrated with AWS services

Demo

Our banking app

Ousobank	
My Accounts	
Account	Balance
Savings	0.00
Holiday	912.00
Create Account	

Design Process

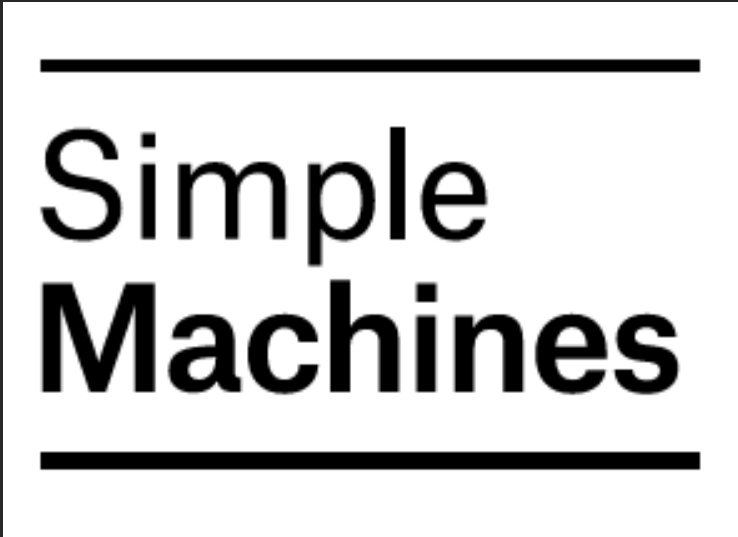
Terminology:

- Command
- Event
- Aggregate
- Projection /
Materialized View

1. Model your domain as commands, events, aggregates and views.
2. Design the command handlers
3. Design the event processor(s) that create your projection
4. Design the read side

Simple Machines

“Simple Machines is a leading Australian technology consultancy at the intersection of data architecture, data engineering and enterprise software delivery. The company specialises in engineering real-time data driven platforms and applications.”

The logo for Simple Machines is contained within a white square. It features the word "Simple" in a large, black, sans-serif font, positioned above the word "Machines" which is in a larger, bold, black, sans-serif font. Two thick, black horizontal lines are positioned above and below the text, framing it.

**Simple
Machines**

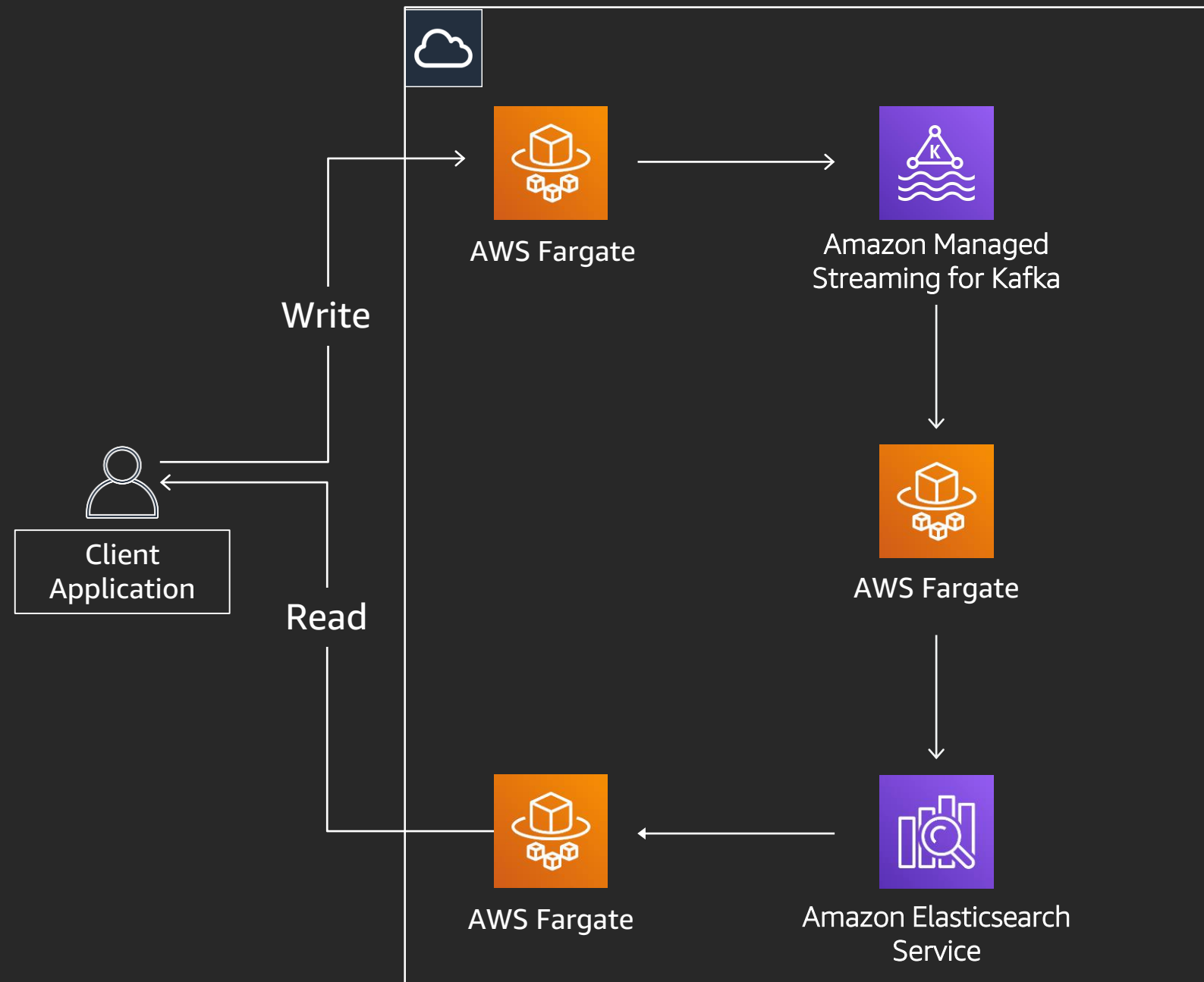
Simple Machines - experimenting

Simple Sourcing - event source data abstraction built to use Kafka as primary data store

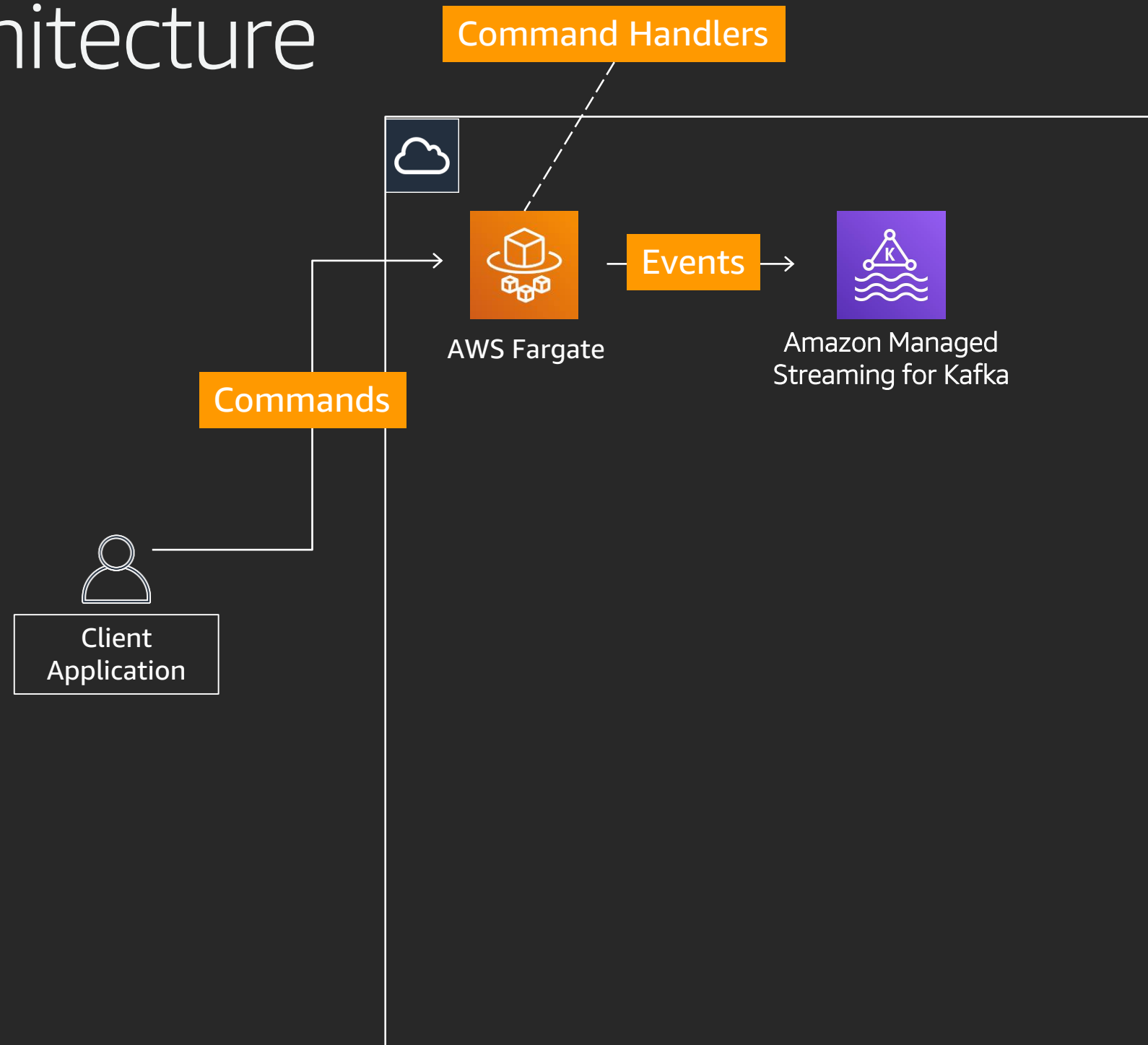
Simple Saga - Saga (co-ordination) layer built using Simple Sourcing to provide a robust execution engine with compensation semantics

<https://simplesource.io/>

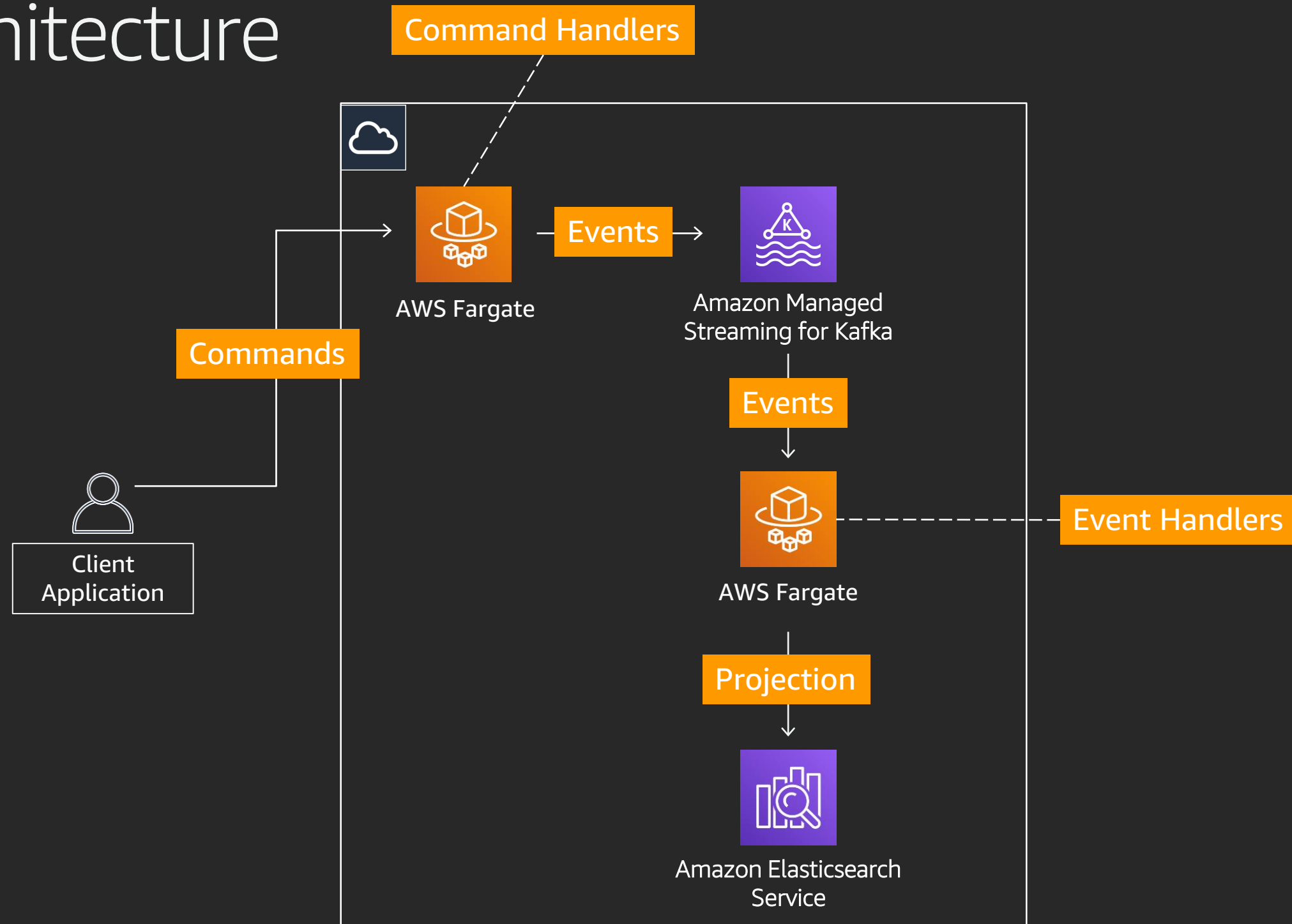
Demo architecture



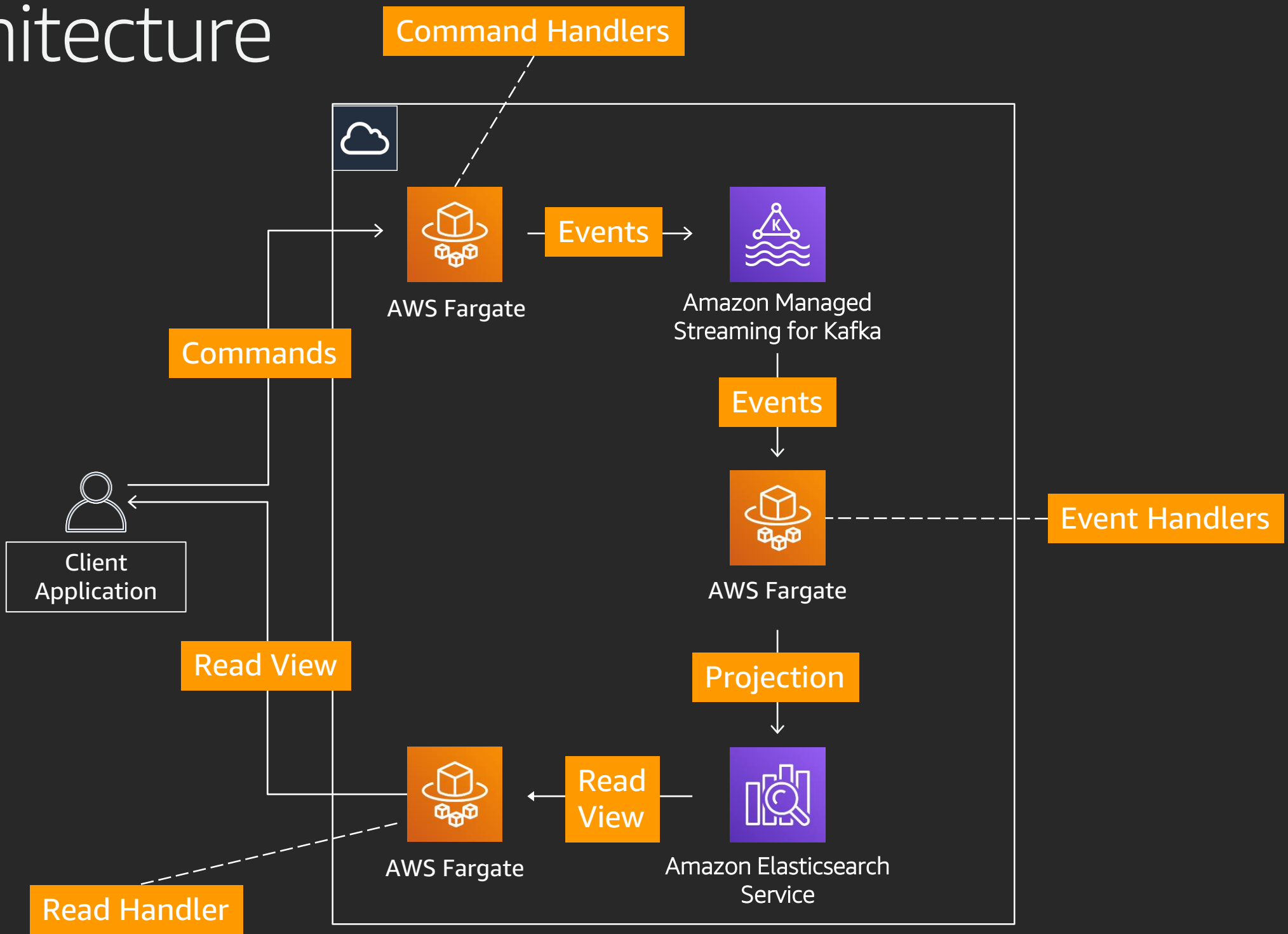
Demo architecture



Demo architecture



Demo architecture



< time for some code />

Closing

Learning objectives

- The what and why of event sourcing and how Amazon Managed Streaming for Apache Kafka (MSK) can help.
- Encourage customers to think differently about how they capture, store and process their data.
- Explore the design and build process behind an event sourcing demo built around MSK.

Thank you!

James Ousby

joousby@amazon.com