

AWS re:Invent

DECEMBER 2 - 6, 2024 | LAS VEGAS, NV

Event-driven architectures at Scale: Manage millions of events

Eric Johnson

Principal Developer Advocate
AWS

Brian Zambrano

Senior Specialist Solutions Architect,
Serverless
AWS

- What is scale?
- What is EDA?
- Event schema enforcement and evolution
- Event schema management and discoverability
- Observability
- Wrap

What is scale?

Scale in traffic



We've got you covered

SOME NUMBERS

- During Prime Day 2024, Amazon DynamoDB peaked at **146 million requests per second**
- AWS Lambda handled over **1.3 trillion invocations during Prime Day 2024**
- AWS Lambda processes **tens of trillions of requests each month**
- Amazon Simple Queue Service (Amazon SQS) processes over **100 million messages per second at peak**
- Amazon Managed Streaming for APACHE Kafka (Amazon MSK) ingests **trillions of events per day** for real-time stream processing

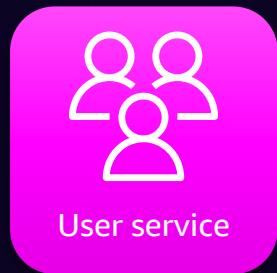
This, is not that

Scale in architecture and the teams that manage it



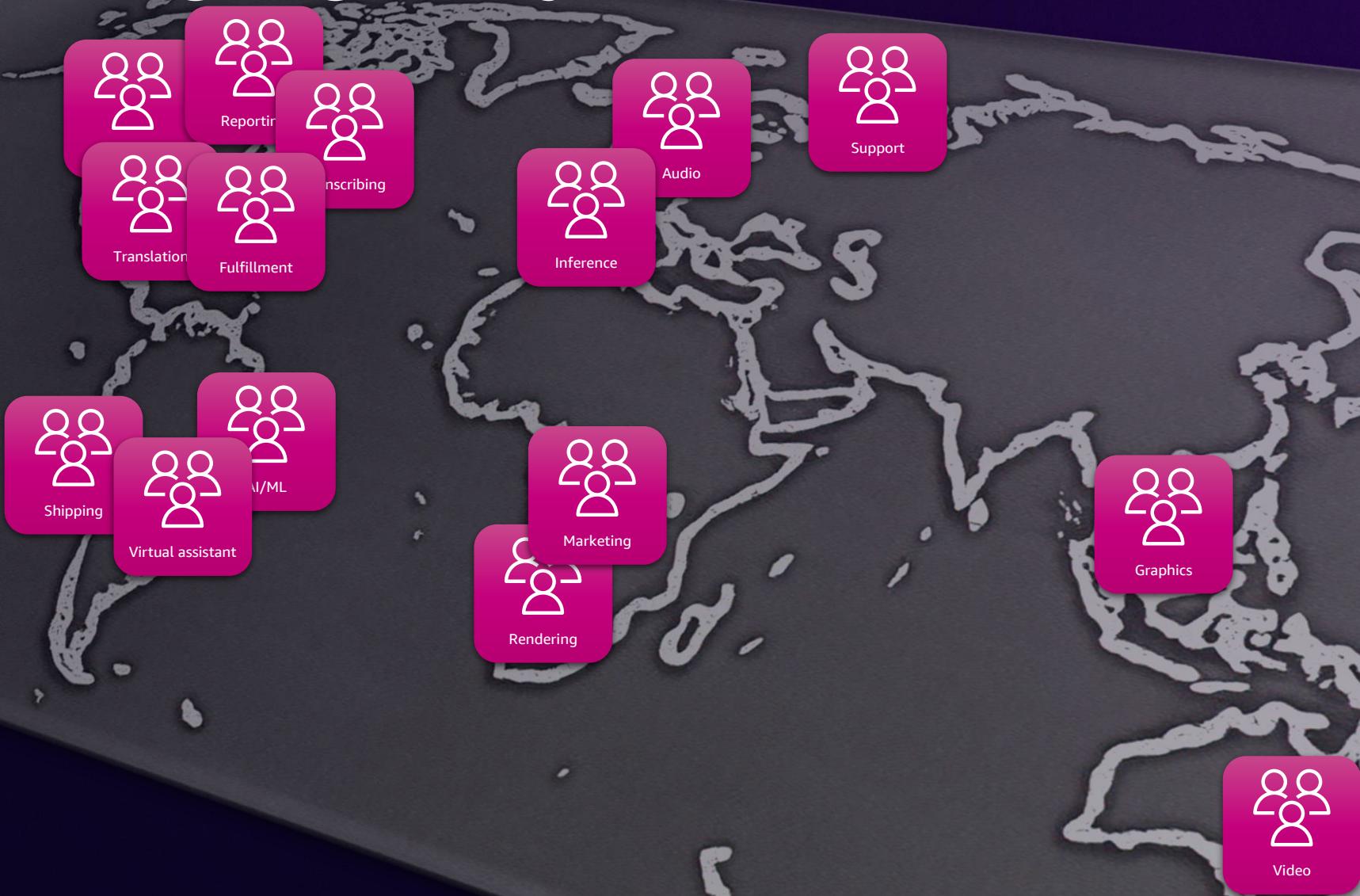
Successful EDAs do not generally look like this

Scale in architecture and the teams that manage it



They look like this

Scale in geography



It's all about the event

In event-driven architectures, the event and health of the event are critical to the success of the application

This is what this talk is all about

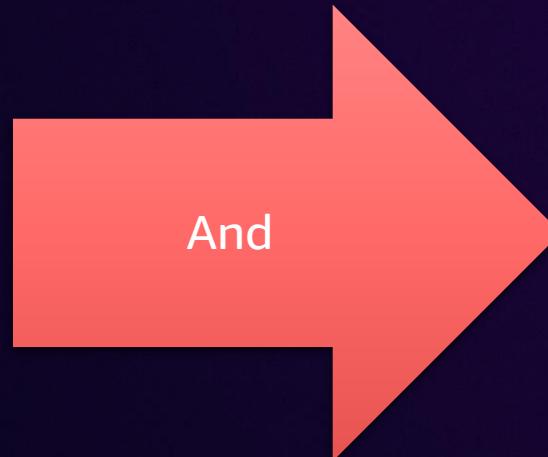


What is EDA?

Defining event-driven architecture



Something happens



And



We react

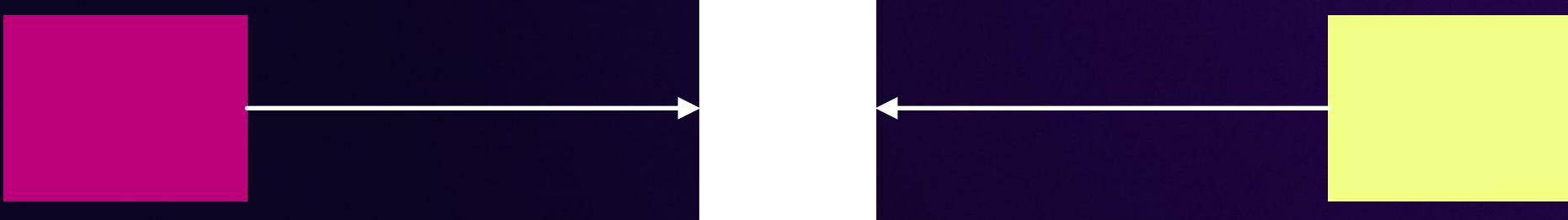
What is an event?

- Events are signals that a system's state has changed
- Events occur in the past (such as, ChannelCreated)
- Events cannot be changed (immutable)
- Event schemas are the contract between the producers and consumers

```
{  
  "source": "pizza.service.com",  
  "detail-type": "OrderCreated",  
  "detail": {  
    "metadata": {  
      "idempotency-key": "837uy4erje"  
    },  
    "data": {  
      "channel-id": "983u4ejrhewio9039oi4kerj",  
      "created-at": "2021-11-26T16:05:09-04:00",  
      "name": "noPineappleOnPizza"  
      "description": "All about real pizza",  
      "Tags": ["Food", "Proper Pizza"],  
      "OrderType": ["Pickup"],  
      "city": "Melbourne"  
      "region": "AU",  
    }  
  }  
}
```

Schema management

EDA Components



Producer

Event producers are systems that detect a change in state or notice updates and publish those facts. Application code, a database or a time-based trigger can serve as event producers, among other things.

Event broker

An event broker is a meeting place between the producers and consumers. The broker is how events are exchanged.

Consumer

Event consumers are systems that listen for events and use them for their purposes. It's possible, and common, for a consumer to receive an event, perform some work and publish its event in response.

Event broker categories



Event broker

An event broker is a meeting place between the producers and consumers. The broker is how events are exchanged.

Pull-based

Queues

Amazon SQS
Amazon MQ

Streams

Amazon Kinesis
Apache Kafka

Amazon Managed Streaming for Apache Kafka (MSK)

Push-based

Routers

Amazon EventBridge
Amazon SNS

Event brokers



Event broker

An event broker is a meeting place between the producers and consumers. The broker is how events are exchanged.

Push-based

EventBridge



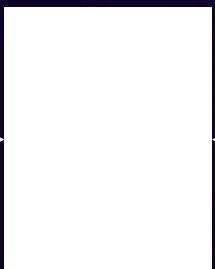
I will focus on this, with callouts for other event brokers like Kafka.

Loose coupling

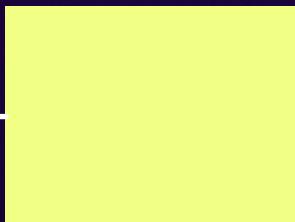
```
{  
  "eventType": "OrderCreated",  
  "eventId": "c4d7e9c1-2e1d",  
  "timestamp": "2023-05-01T14:25:43.123Z",  
  "orderId": "ORD-12345",  
  "customerDetails": {  
    "email": "customer@example.com",  
    "name": "John Doe",  
    "phone": "+1 555 123 4567",  
    "address": "123 Main St, Anytown USA"  
  }  
}
```



Producer

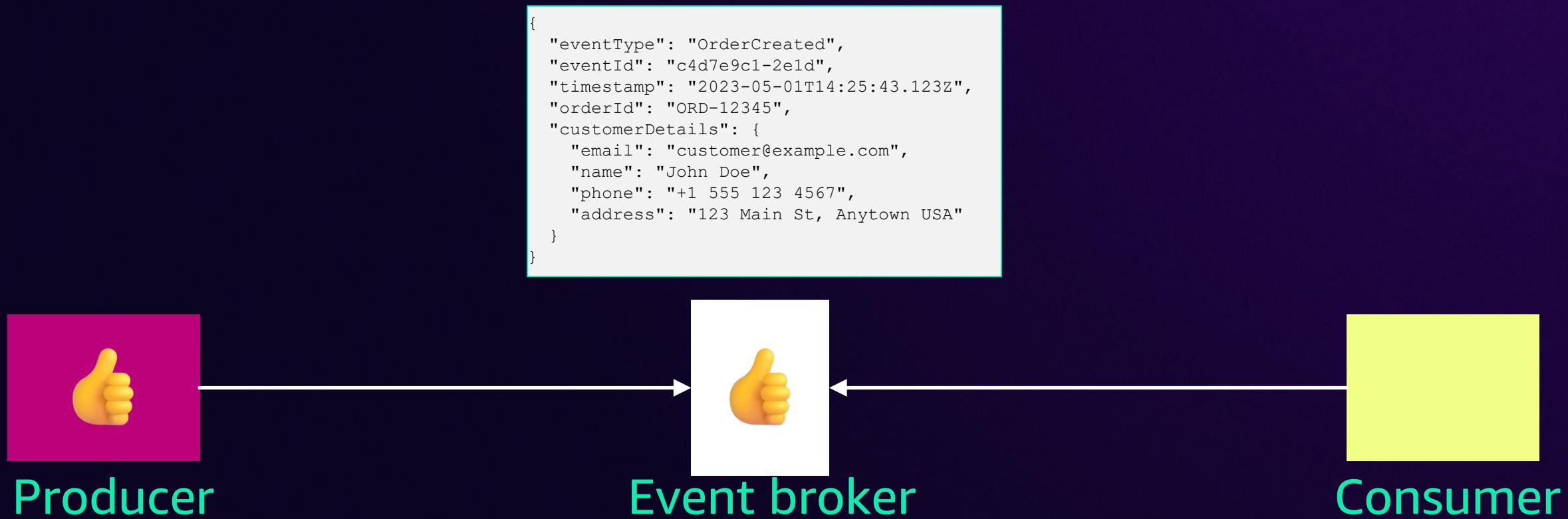


Event broker



Consumer

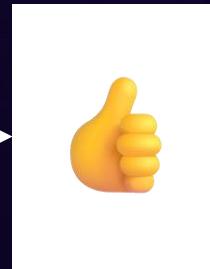
Loose coupling



Loose coupling



Producer



Event broker



Consumer

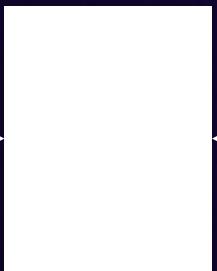
```
{  
  "eventType": "OrderCreated",  
  "eventId": "c4d7e9c1-2e1d",  
  "timestamp": "2023-05-01T14:25:43.123Z",  
  "orderId": "ORD-12345",  
  "customerDetails": {  
    "email": "customer@example.com",  
    "name": "John Doe",  
    "phone": "+1 555 123 4567",  
    "address": "123 Main St, Anytown USA"  
  }  
}
```

Loose coupling++

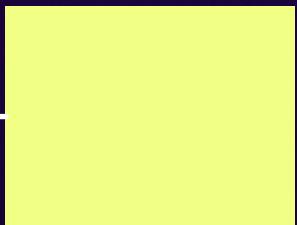
```
{  
  "eventType": "OrderCreated",  
  "eventId": "c4d7e9c1-2e1d",  
  "timestamp": "2023-05-01T14:25:43.123Z",  
  "orderId": "ORD-12345",  
  "customerDetails": {  
    "email": "customer@example.com",  
    "name": "John Doe",  
    "phone": "+1 555 123 4567",  
    "address": "123 Main St, Anytown USA"  
  }  
}
```



Producer



Event broker



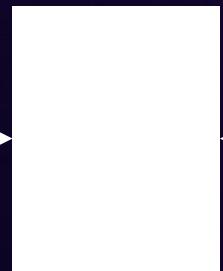
Consumer

Loose coupling++

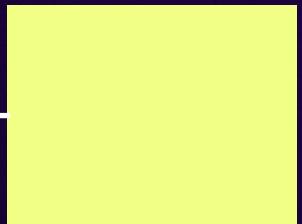
```
{  
  "eventType": "OrderCreated",  
  "eventId": "c4d7e9c1-2e1d",  
  "timestamp": "2023-05-01T14:25:43.123Z",  
  "order": {  
    "id": "ORD-12345",  
    "timestamp": "2023-05-01T...",  
  },  
  "customerDetails": {  
    "email": "customer@example.com",  
    "name": "John Doe",  
    "phone": null,  
    "address": "123 Main St, Anytown USA"  
  }  
}
```



Producer

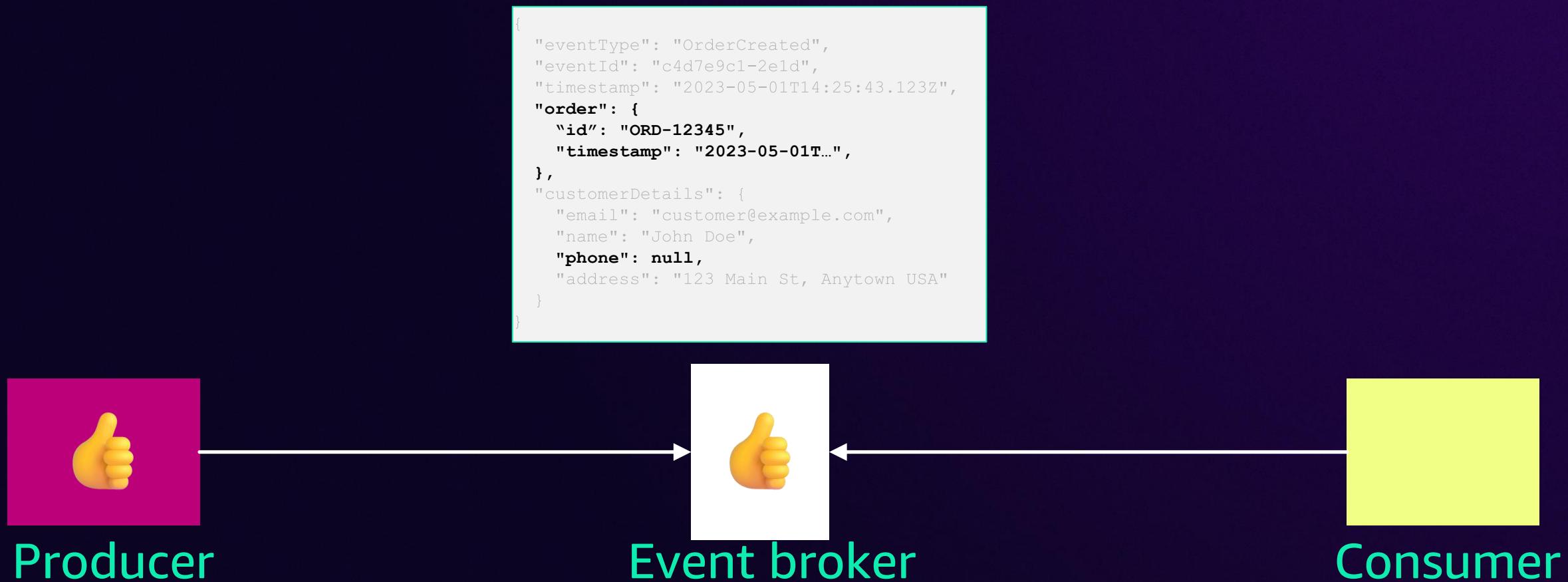


Event broker



Consumer

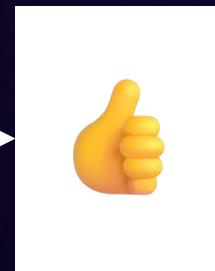
Loose coupling++



Loose coupling++



Producer



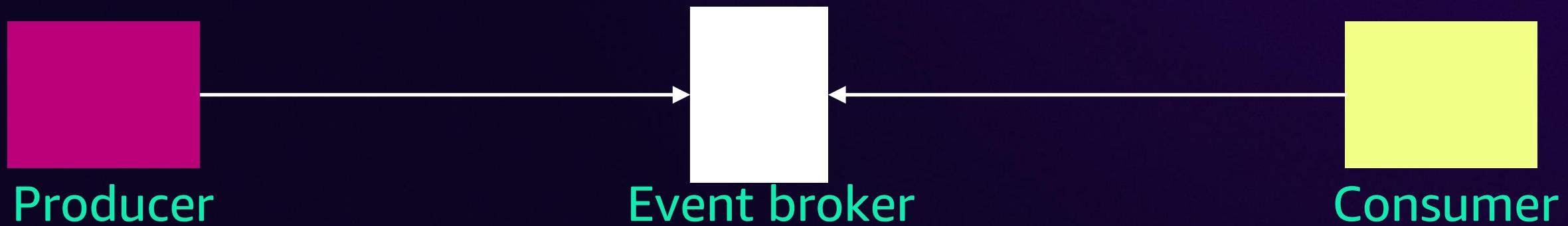
Event broker



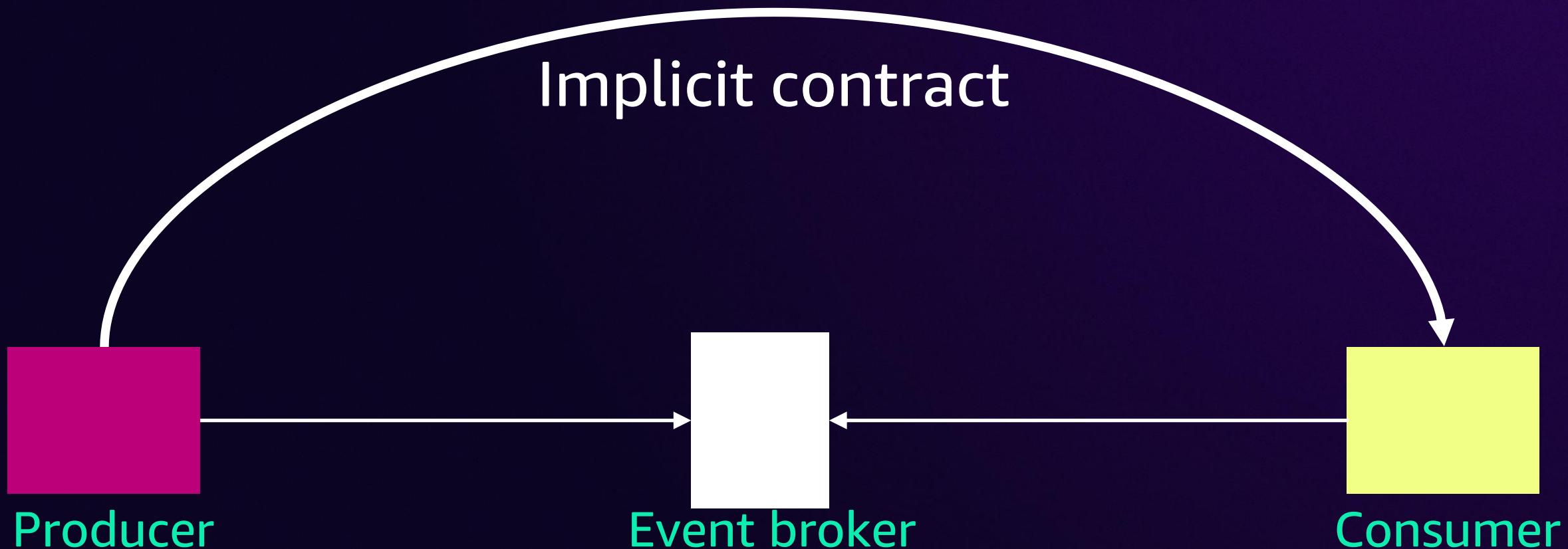
Consumer



Loose coupling != No coupling



Loose coupling != No coupling

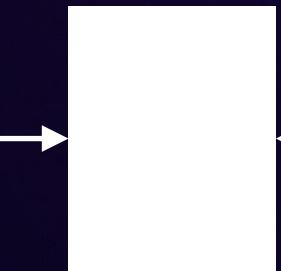


Loose coupling != No coupling

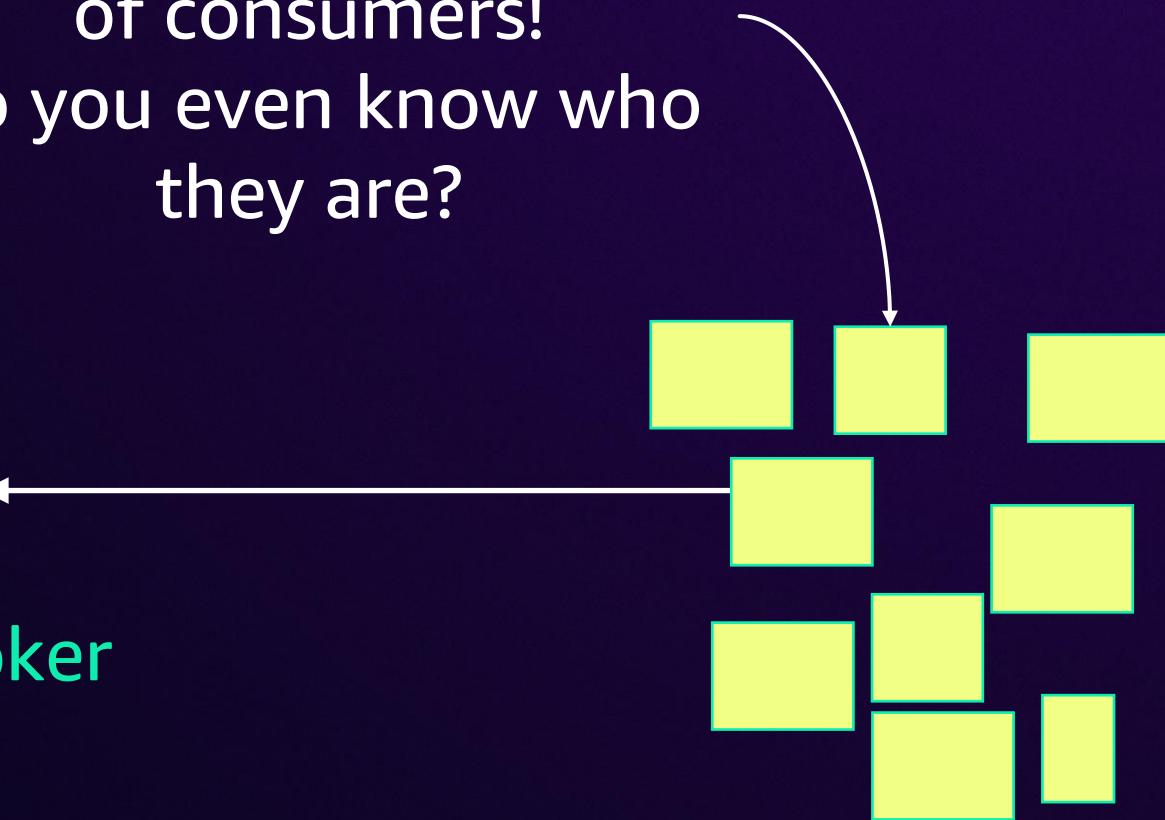
At scale, there may be
hundreds or thousands
of consumers!
Do you even know who
they are?



Producer

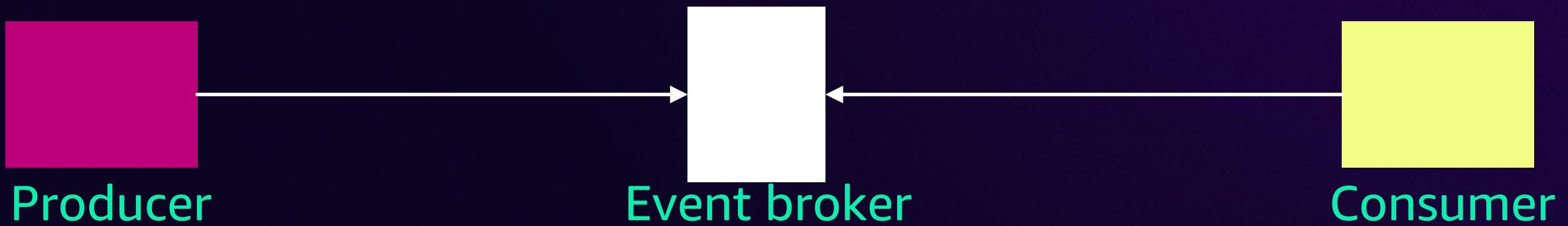


Event broker



Consumers

Options

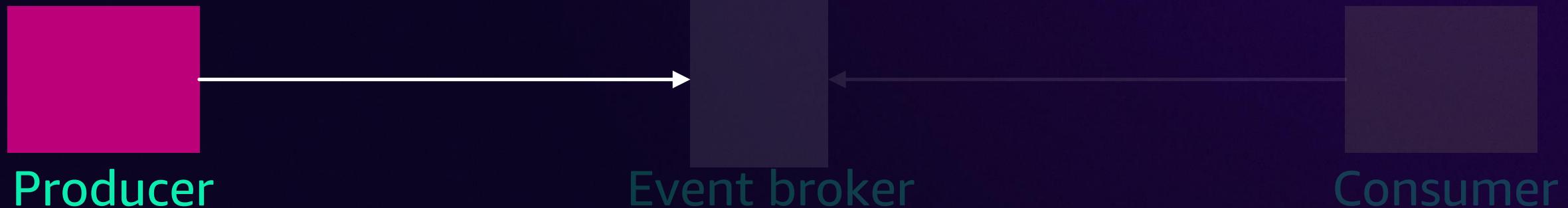


Options

Promises from the producers



[This Photo](#) by Unknown Author is licensed under [CC BY-SA-NC](#)

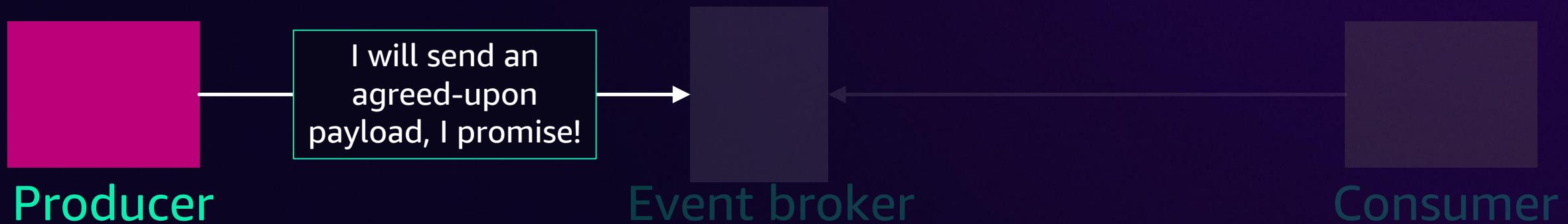


Options

Promises from the producers



[This Photo](#) by Unknown Author is licensed under [CC BY-SA-NC](#)



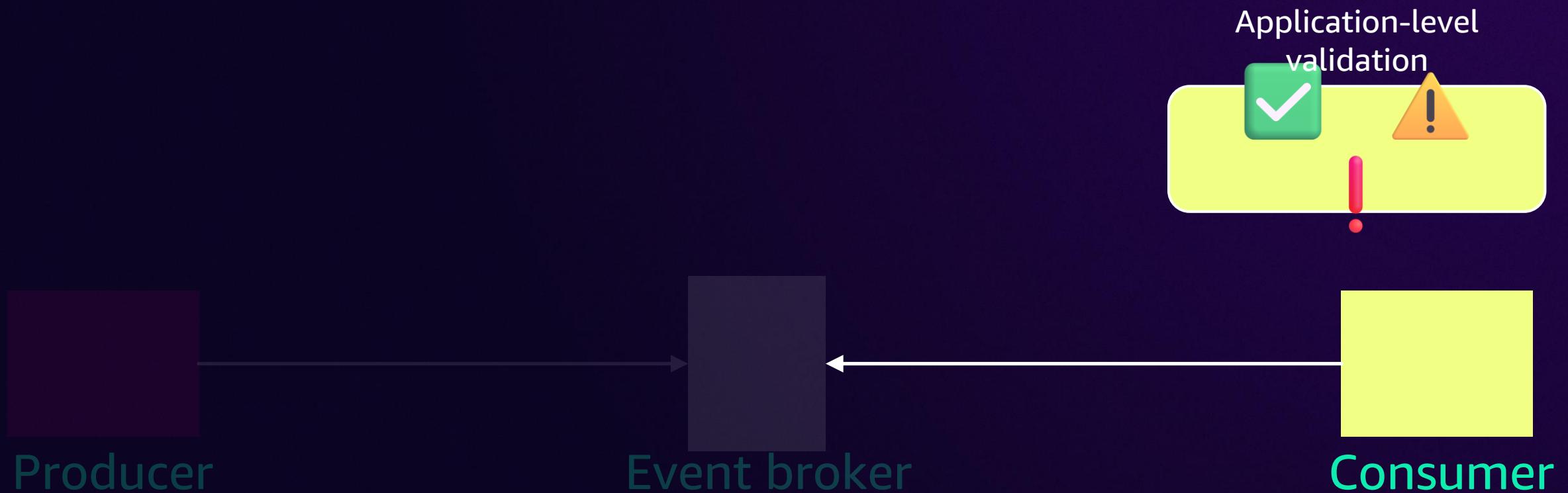
Options



Enforcement



Options

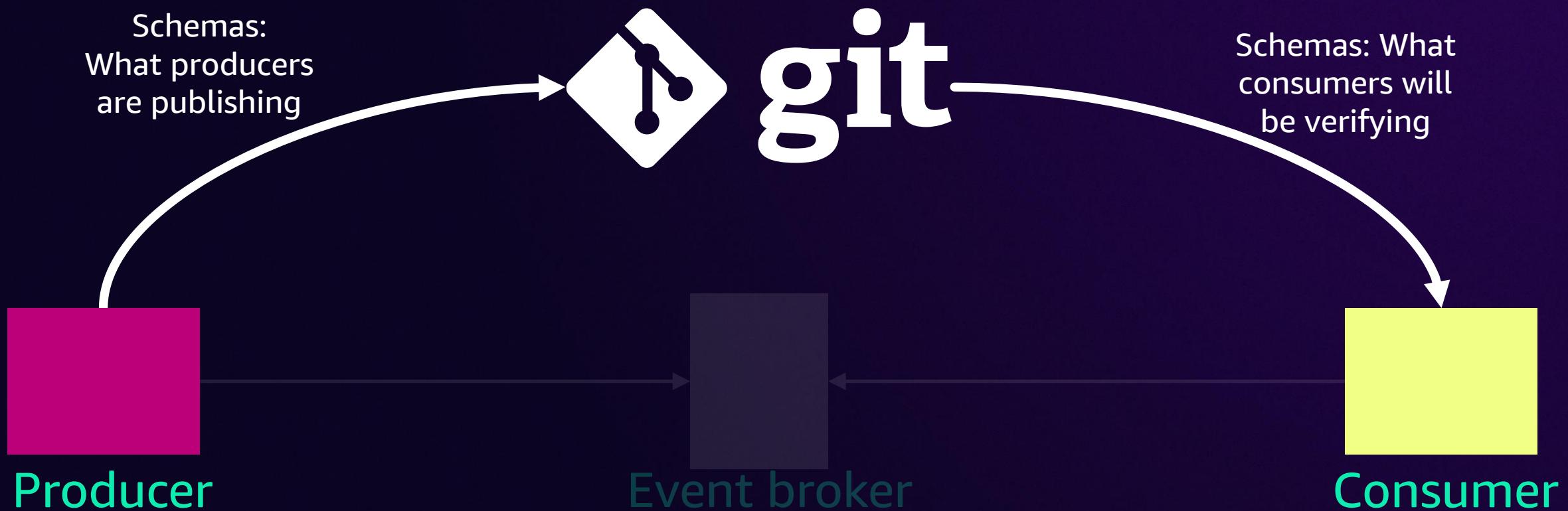


JSON Schema

- Defines the structure and format of JSON data, allowing you to describe expected properties, types, and constraints
- Helps to maintain data integrity and improve reliability by preventing issues like missing or additional fields or invalid types in API payloads

```
{  
  "$schema": "http://json-schema.org/draft-  
04/schema#",  
  "title": "PetStoreModel",  
  "type" : "object",  
  "required" : [ "price", "type" ],  
  "properties" : {  
    "id" : {  
      "type" : "integer"  
    },  
    "type" : {  
      "type" : "string",  
      "enum" : [ "dog", "cat", "fish" ]  
    },  
    "price" : {  
      "type" : "number",  
      "minimum" : 25.0,  
      "maximum" : 500.0  
    }  
  }  
}
```

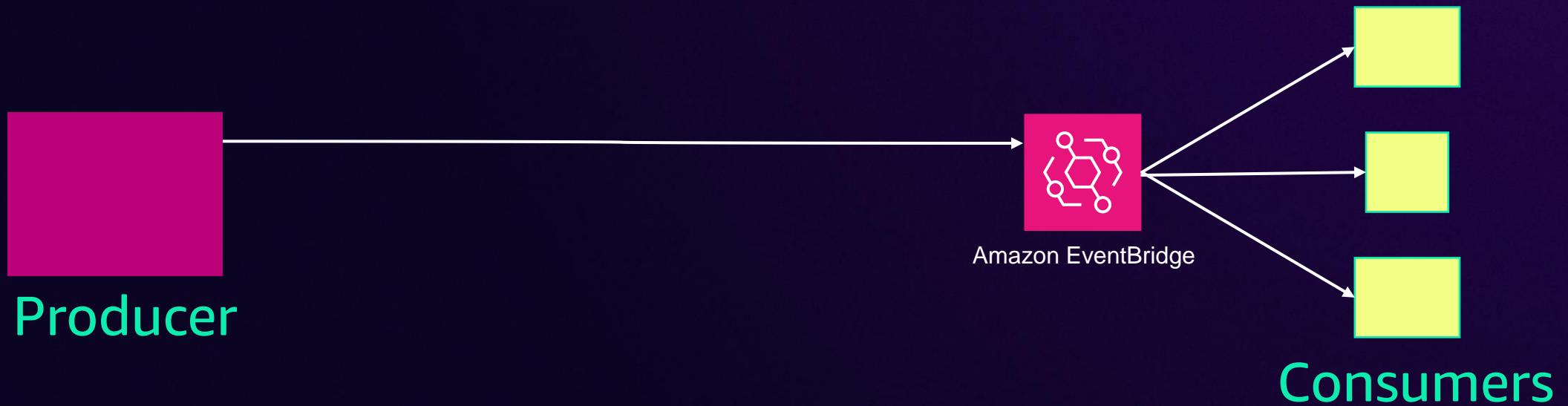
Validation at the edges



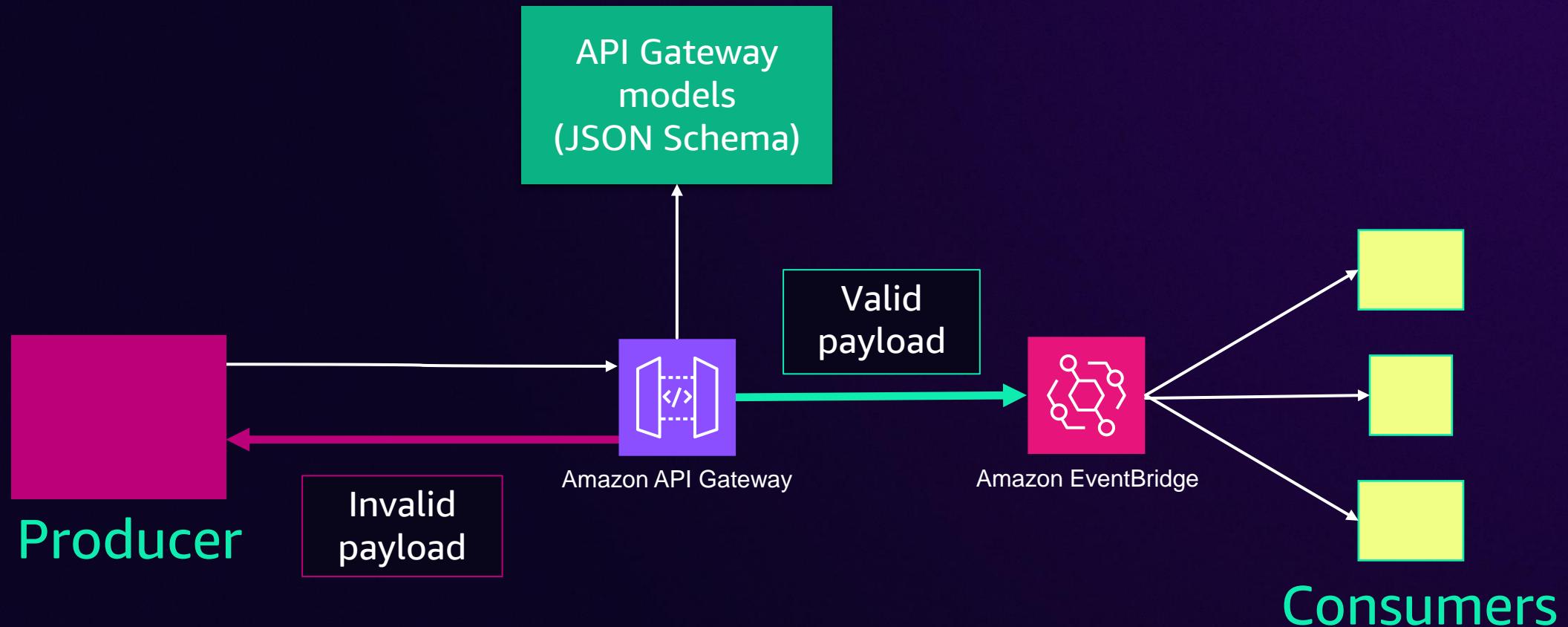
Let's look at enforcement



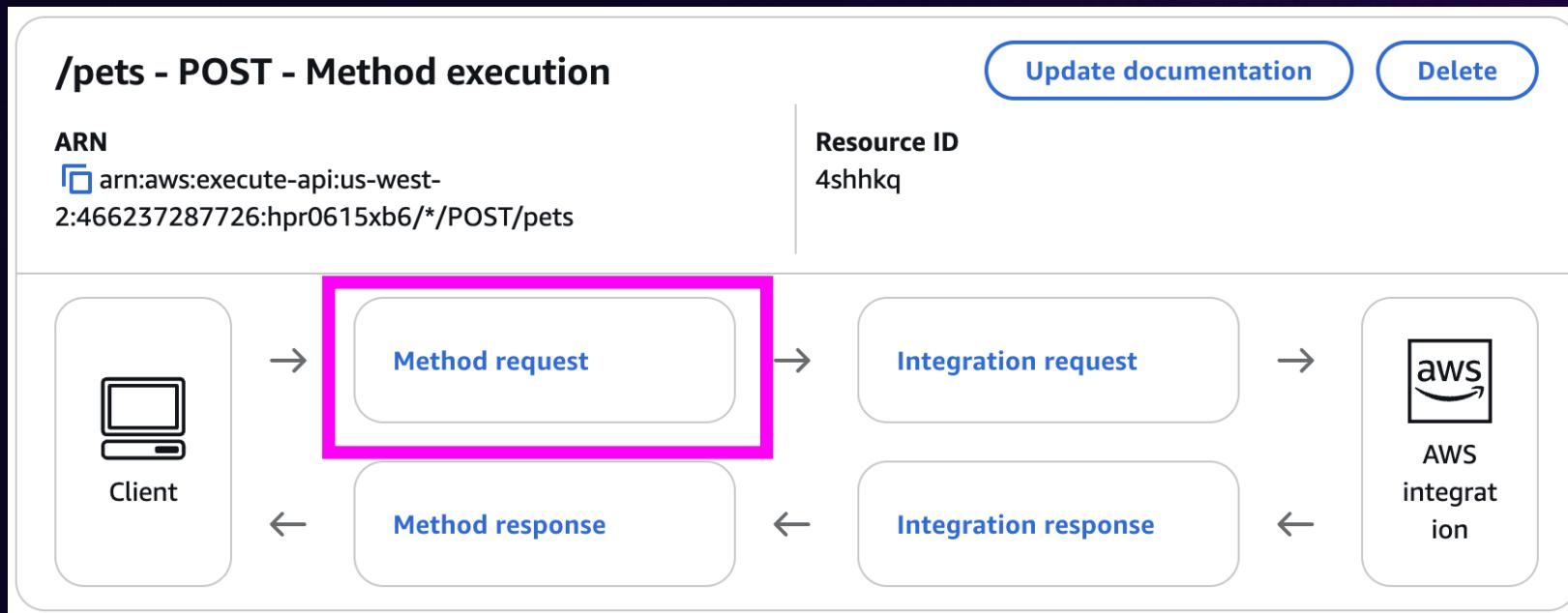
Schema enforcement with API Gateway



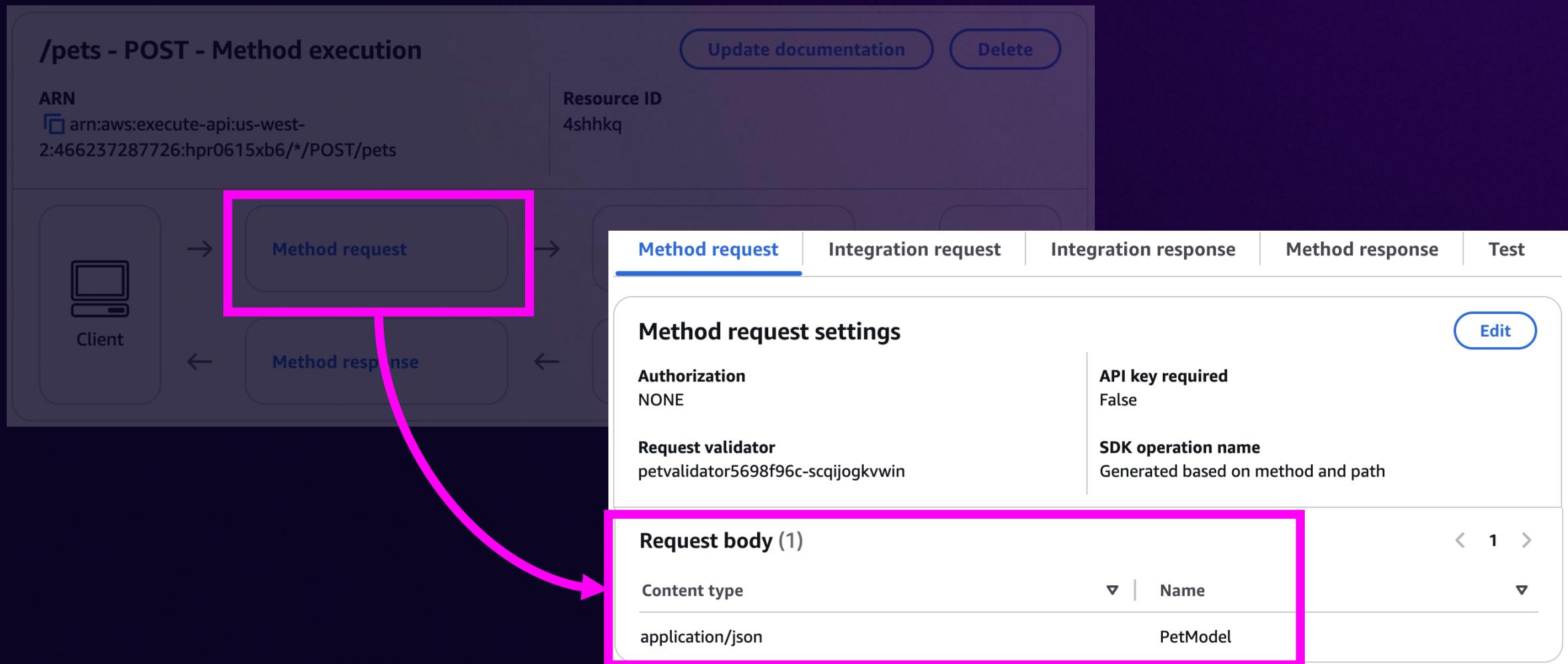
Schema enforcement with API Gateway



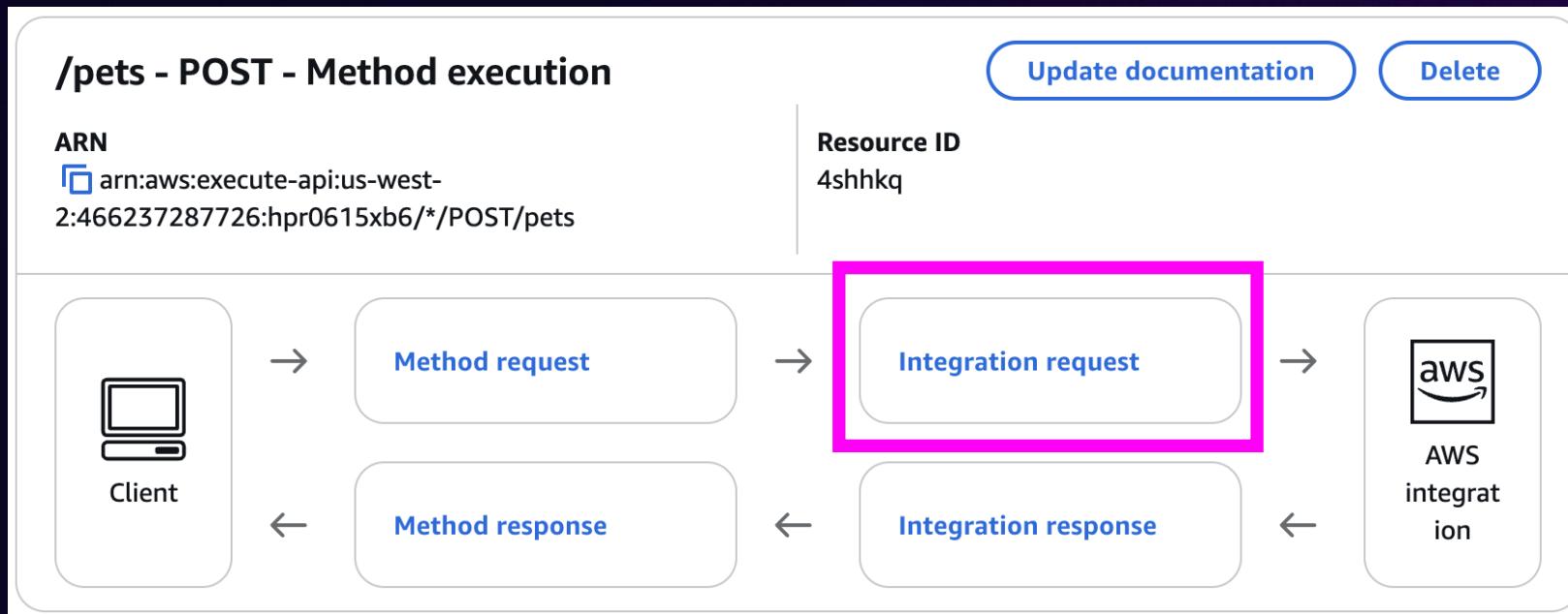
API Gateway method request



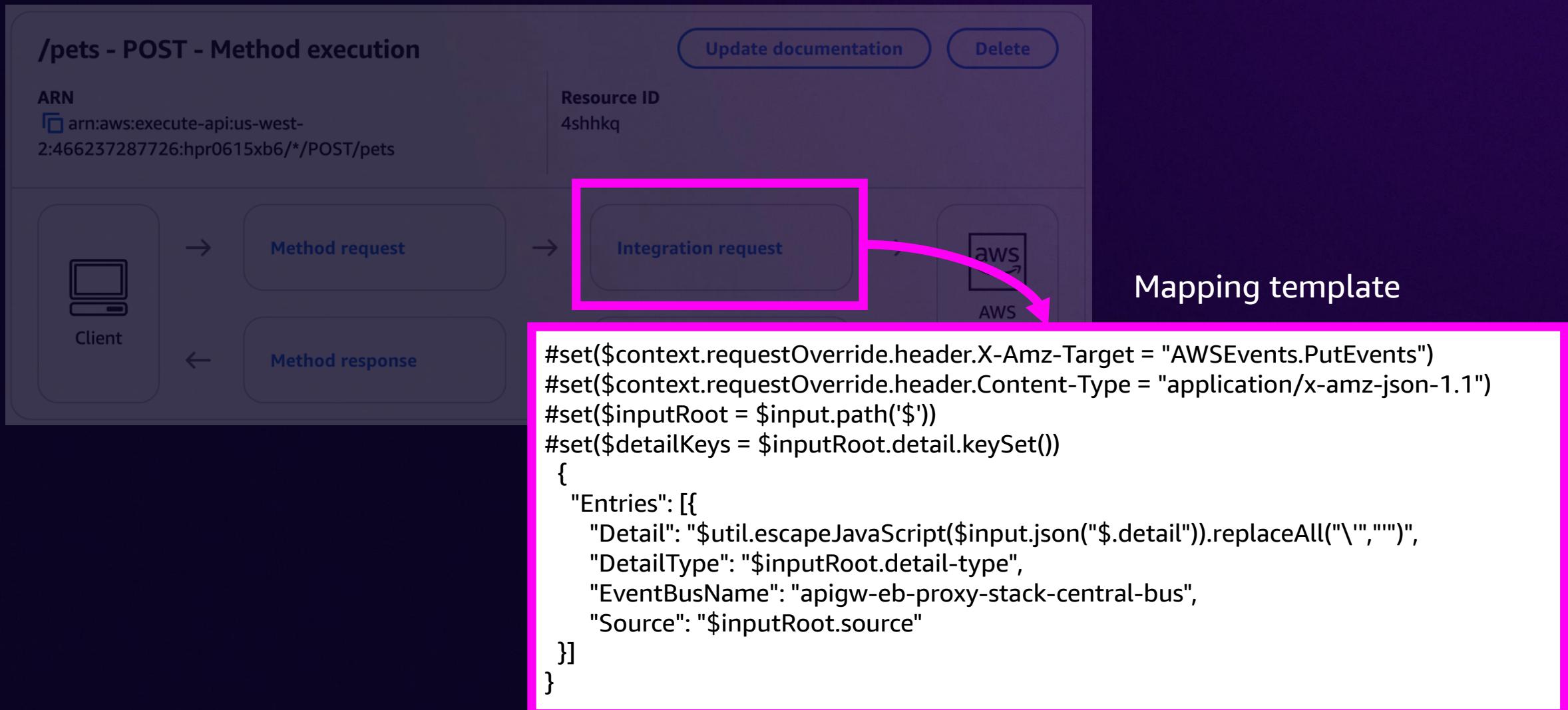
API Gateway method execution



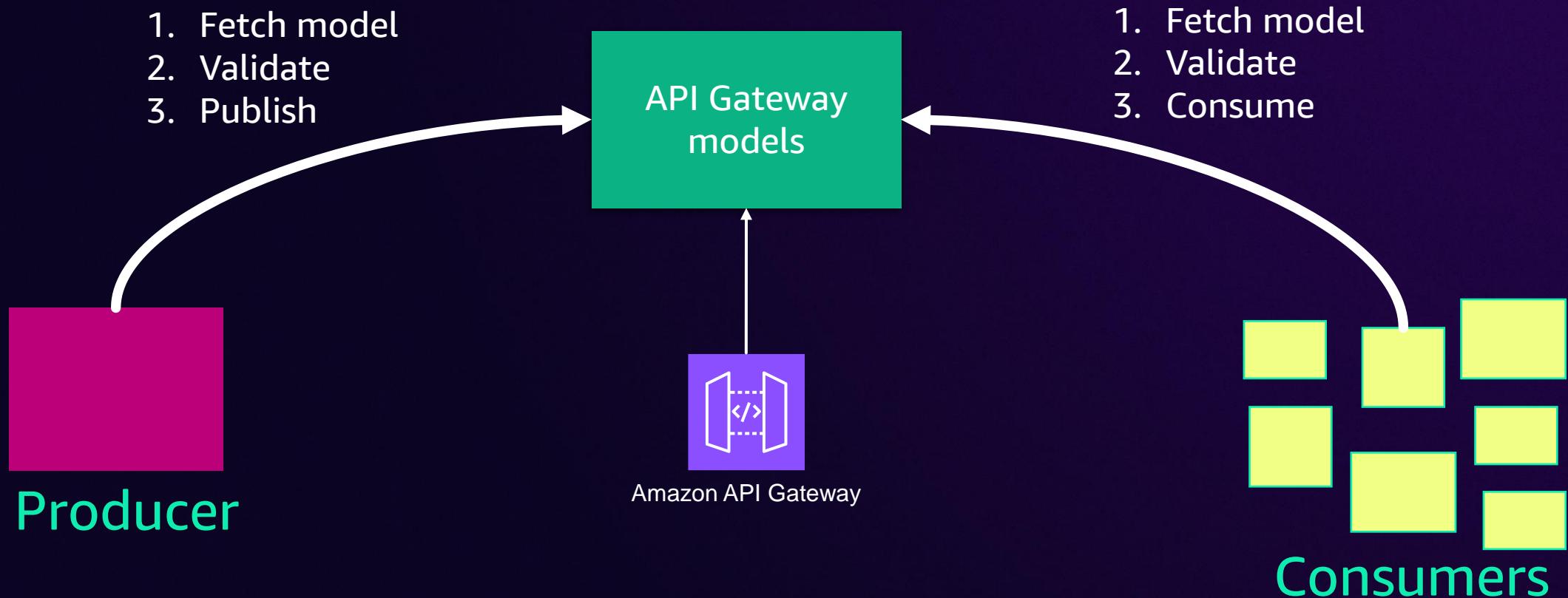
API Gateway integration request



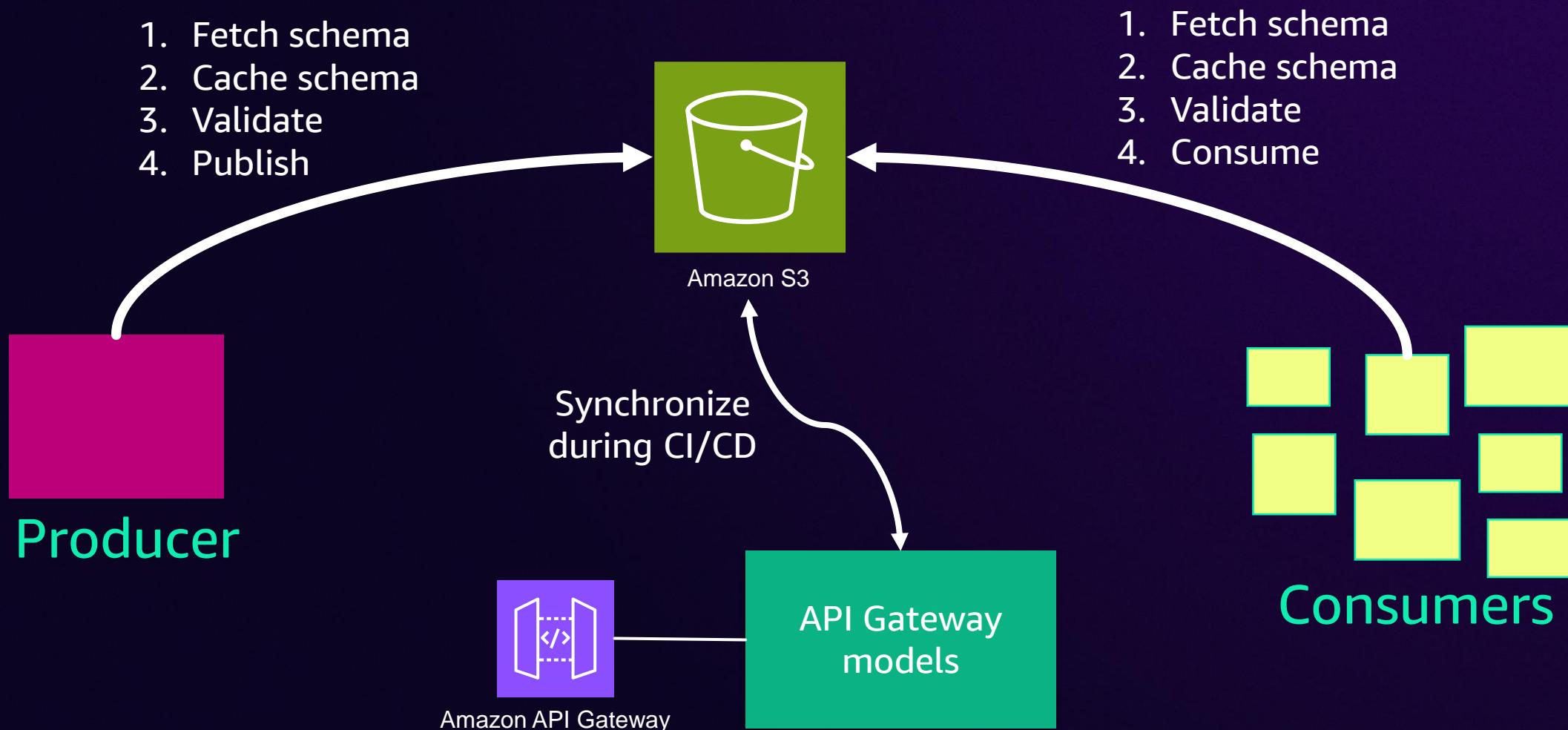
API Gateway integration request



JSON Schema at the edges



JSON Schema at the edges



JSON Schema at the edges

```
import boto3

from jsonschema import validate

api_gateway = boto3.client("apigateway")

def validate_payload(api_id, model_name, json_payload):
    # Fetch json schema / model from APIGW (or from S3, etc.)
    model = api_gateway.get_model(
        restApiId=api_id,
        modelName=model_name
    )
    schema = json.loads(model["schema"])
    # Validate the payload using jsonschema library
    validate(instance=json_payload, schema=schema)
```

Schema evolution

Schema evolution

```
{  
  "id": 1234,  
  "type": "dog",  
  "adoptionFee": 25  
}
```

Schema evolution

```
{  
    "detail": {  
        "id": 1234,  
        "type": "dog",  
        "adoptionFee": 25  
    }  
}
```

Schema evolution

```
{  
    "metadata": { },  
    "detail": {  
        "id": 1234,  
        "type": "dog",  
        "adoptionFee": 25  
    }  
}
```

Schema evolution

```
{  
  "metadata": {  
    "correlationId": "05059C53",  
    "idempotencyKey": "E430C4DC",  
    "version": 1,  
    "schema": "s3://bucket/Pets-1.schema.json",  
    "ttl": null,  
    "nextVersion": null  
  },  
  "detail": {  
    ...  
  }  
}
```

Schema evolution

```
{  
  "metadata": {  
    "correlationId": "05059C53",  
    "idempotencyKey": "E430C4DC",  
    "version": 1,  
    "schema": "s3://bucket/Pets-1.schema.json",  
    "ttl": null,  
    "nextVersion": null  
  },  
  "detail": {  
    ...  
  }  
}
```

This version

Schema evolution

```
{  
  "metadata": {  
    "correlationId": "05059C53",  
    "idempotencyKey": "E430C4DC",  
    "version": 1,  
    "schema": "s3://bucket/Pets-1.schema.json",  
    "ttl": null,  
    "nextVersion": null  
  },  
  "detail": {  
    ...  
  }  
}
```

Downloadable
schema

Schema evolution

```
{  
  "metadata": {  
    "correlationId": "05059C53",  
    "idempotencyKey": "E430C4DC",  
    "version": 1,  
    "schema": "s3://bucket/Pets-1.schema.json",  
    "ttl": null,  
    "nextVersion": null  
  "detail": {  
    ...  
  }  
}
```

This version is good
indefinitely

Schema evolution

```
{  
  "metadata": {  
    "correlationId": "05059C53",  
    "idempotencyKey": "E430C4DC",  
    "version": 1,  
    "schema": "s3://bucket/Pets-1.schema.json",  
    "ttl": "2025-04-01:00:00:00Z",  
    "nextVersion": {  
      "version": 2,  
      "schema": "s3://bucket/Pets-2.schema.json"  
    }  
  },  
  "detail": { ... }  
}
```



Schema evolution

```
{  
  "metadata": {  
    "correlationId": "05059C53",  
    "idempotencyKey": "E430C4DC",  
    "version": 1,  
    "schema": "s3://bucket/Pets-1.schema.json",  
    "ttl": "2025-04-01:00:00:00Z",  
    "nextVersion": {  
      "version": 2,  
      "schema": "s3://bucket/Pets-2.schema.json"  
    }  
  },  
  "detail": { ... }  
}
```

Expiration of
version 1



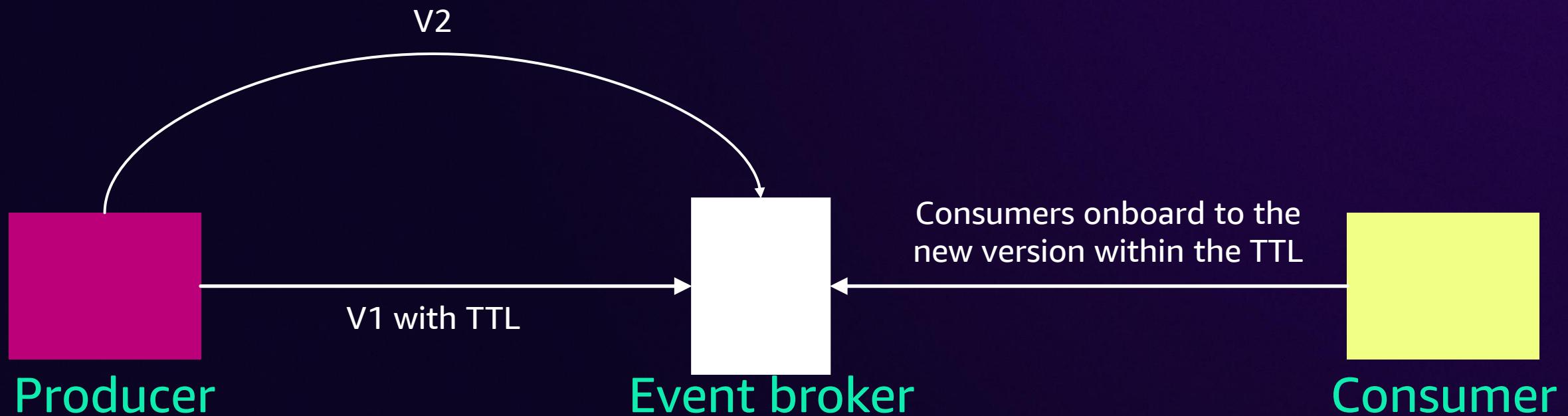
Schema evolution

```
{  
  "metadata": {  
    "correlationId": "05059C53",  
    "idempotencyKey": "E430C4DC",  
    "version": 1,  
    "schema": "s3://bucket/Pets-1.schema.json",  
    "ttl": "2025-04-01:00:00:00Z",  
    "nextVersion": {  
      "version": 2,  
      "schema": "s3://bucket/Pets-2.schema.json"  
    }  
  },  
  "detail": { ... }  
}
```

Next active version after ttl



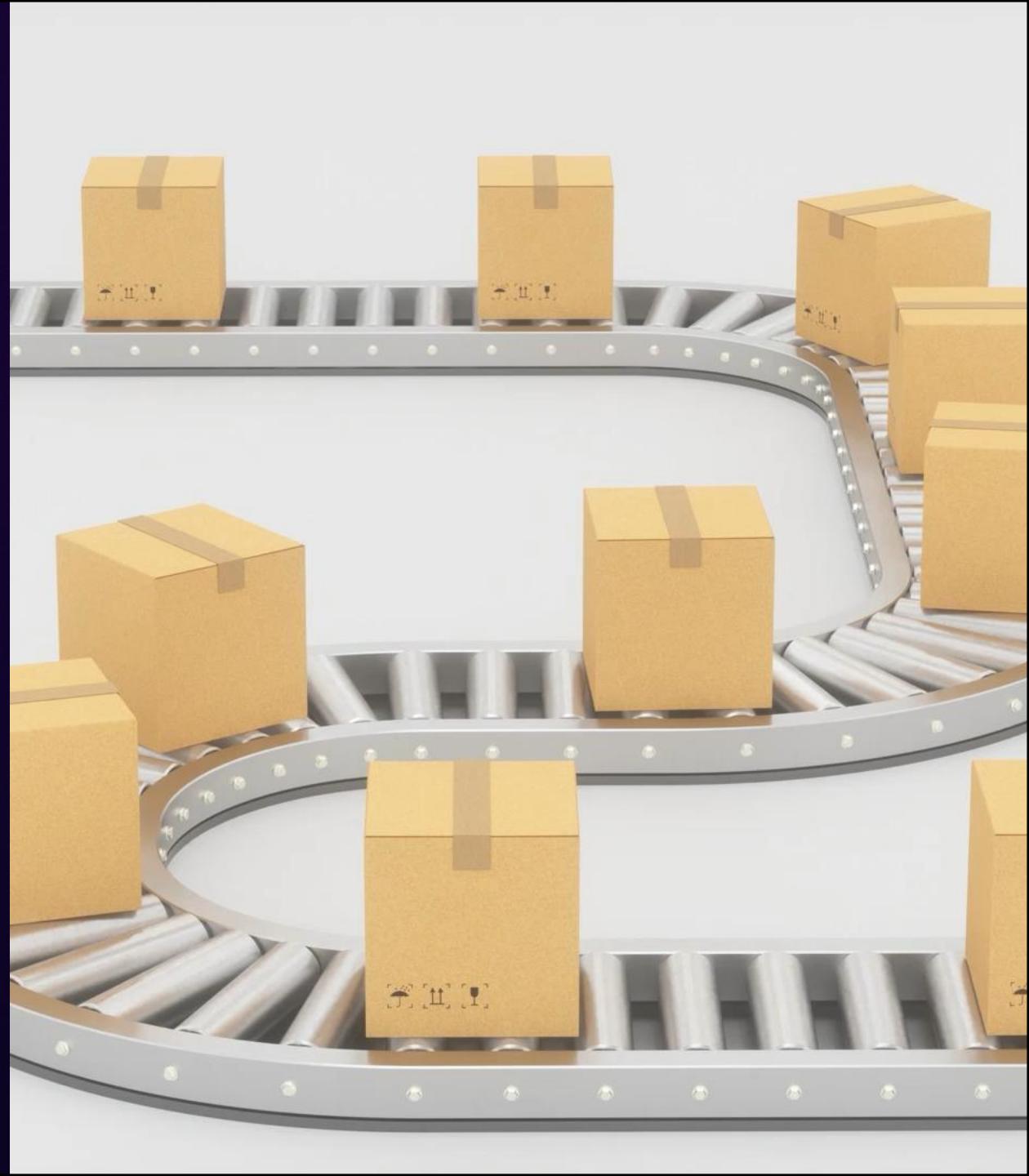
Double publishing events



Event schema management and discovery

Infrastructure governance

- Provide deployable AWS resources using proper structure
- Control event structure from an infrastructure standpoint
- Most effective when you have control end-to-end.





AWS Service Catalog

Portfolios Info

Local Imported

Local portfolios (1)



Actions ▾

Create portfolio

< 1 > |

Name	▲	Created time	▼	Portfolio ID	▼	ARN	▼	Owner	▼	Description	▼	Current vs. budget	▼	Forecast vs. budget	▼
Bus Rules		Thu, Oct 3, 2024, 8:12:58 PM PDT		port-tcd7n2ydi2lz4				ericdj@amazon.com		Sample portfolio		-		-	

Products (2)



Create product

Add product to portfolio

Remove

< 1 > |

Product name	▲	Product ID	▼	Product type	▼	Created time	▼	Description	▼	Constraints	▼
Reporting Rule		prod-k6z4ainehbzac		CLOUDFORMATIONTEMPLATE		Thu, Oct 3, 2024, 8:37:46 PM PDT		AWS EventBridge rule to capture reporting events		0	
Transcription Rule		prod-kpga3kq4lebgi		CLOUDFORMATIONTEMPLATE		Thu, Oct 3, 2024, 8:41:46 PM PDT		AWS EventBridge rule to capture transcription rules		0	





AWS Service Catalog

Products (384)

Info

Add to portfolio

Serverless

X

18 matches

< 1 > |

Product name	Source	Description
.NET CI/CD for Serverless Applications	AWS Quick Starts	Deploys a CI/CD pipeline for .NET serverless applications.
Amazon SageMaker for Tableau	AWS Quick Starts	This Quick Start sets up an Amazon Web Services (AWS) architecture that allows you to integrate Amazon SageMaker machine learning (ML) models in Tableau's calculated fields. The serverless application it deploys is based on Tableau's analytics extension framework. With it, you can connect SageMaker ML models to Tableau workbooks in both Tableau Desktop and Tableau Server. This Quick Start uses AWS CloudFormation templates to deploy a REST API managed by Amazon API Gateway and Lambda functions to connect Tableau and SageMaker. With Amazon Cognito, it also provides a user authentication flow based on AWS best practices.
Application Layer	High Reliability Architectures	The High Reliability Serverless App Layer deploys the back-end API for a photo-sharing website to an s3 bucket and cloudfront distribution.
AWS Marketplace Serverless SaaS Integration	AWS Quick Starts	Deploys AWS Marketplace Serverless Software as a Service (SaaS) Integration on the AWS Cloud for registered sellers integrating new SaaS listings.





Backstage

Bună ziua, guest!

Backstage Service Catalog

NYC 07:56 UTC 11:56 STO 13:56 TYO 20:56

SERVICES WEBSITES LIBRARIES DOCUMENTATION OTHER

CREATE COMPONENT ? SUPPORT

Services

PERSONAL

- Owned 3
- Starred 0

SPOTIFY

- All 6

Owned (3)

NAME	OWNER	LIFECYCLE	DESCRIPTION	ACTIONS
playback-order	guest	production	Playback Order	
searcher	guest	production	Searcher	
shuffle-api	guest	production	Shuffle API	





Backstage

Bună ziua, guest!

Backstage Service Catalog

NYC 07:56 UTC 11:56 STO 13:56 TYO 20:56

SERVICES WEBSITES LIBRARIES DOCUMENTATION OTHER

CREATE COMPONENT ? SUPPORT

Services

PERSONAL

- Owned 3
- Starred 0

SPOTIFY

- All 6

Owned (3)

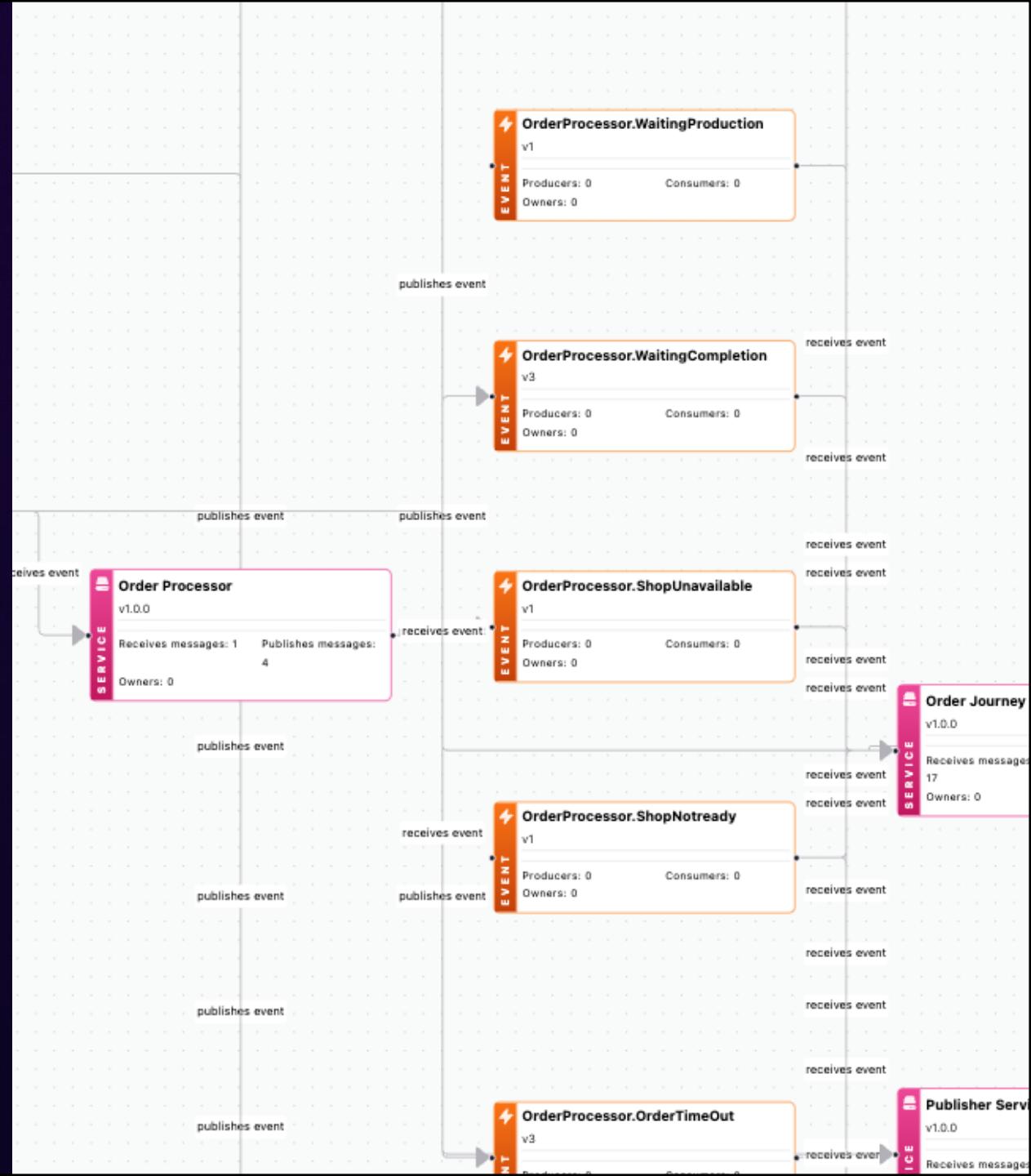
NAME	OWNER	LIFECYCLE	DESCRIPTION	ACTIONS
playback-order	guest	production	Playback Order	
searcher	guest	production	Searcher	
shuffle-api	guest	production	Shuffle API	

David Anderson
Architect, Globalization Partners
Author: *Creating the Flywheel Effect*



Event schema discovery

- Documentation
- Versioning
- Mapping





Amazon EventBridge schema registry

Schemas Info

A schema defines the structure and content of events that are passed on an event bus in Amazon EventBridge. You can browse or search for the schemas of all AWS services on EventBridge. You can automatically generate schemas for events on an event bus, create or upload custom schemas, and organize your custom schemas in custom registries.

All schemas AWS event schema registry **Discovered schema registry** lambda-testevent-schemas serverlesspresso.events

Create registry **Create schema**

Search discovered event schemas

Search schema titles and contents.

< 1 2 > |

aws.partner-mongodb.com-stitch.trigger-629137d5d50929fc8dc6285e@MongoDBDatabaseTriggerForSePc.myTable Discovered schema registry 1 version Last updated May 27, 2022, 01:58 PM PDT	awsserverlessda.serverlesspresso@ConfigService.ConfigChanged Discovered schema registry 22 versions Last updated Sep 21, 2021, 06:48 AM PDT	awsserverlessda.serverlesspresso@OrderJourney.AllEventsStored Discovered schema registry 36 versions Last updated Sep 27, 2021, 03:07 AM PDT	awsserverlessda.serverlesspresso@OrderManager.MakeOrder Discovered schema registry 27 versions Last updated Sep 28, 2021, 05:16 AM PDT
awsserverlessda.serverlesspresso@OrderManager.OrderCancelled Discovered schema registry 63 versions Last updated Sep 27, 2021, 03:07 AM PDT	awsserverlessda.serverlesspresso@OrderManager.OrderCompleted Discovered schema registry 71 versions Last updated Sep 24, 2021, 09:31 AM PDT	awsserverlessda.serverlesspresso@OrderManager.WaitingCompletion Discovered schema registry 99 versions Last updated Sep 27, 2021, 03:07 AM PDT	awsserverlessda.serverlesspresso@OrderProcessor.OrderTimeOut Discovered schema registry 3 versions Last updated Sep 22, 2021, 06:49 AM PDT
awsserverlessda.serverlesspresso@OrderProcessor.ShopNotready Discovered schema registry 1 version Last updated Aug 26, 2021, 05:04 AM PDT	awsserverlessda.serverlesspresso@OrderProcessor.ShopUnavailable Discovered schema registry 1 version Last updated Sep 15, 2021, 11:16 AM PDT	awsserverlessda.serverlesspresso@OrderProcessor.WaitingCompletion Discovered schema registry 3 versions Last updated Aug 27, 2021, 04:12 AM PDT	awsserverlessda.serverlesspresso@OrderProcessor.WaitingProduction Discovered schema registry 1 version Last updated Aug 23, 2021, 04:52 AM PDT





Amazon EventBridge schema registry

awsserverlessda.serverlesspresso@OrderManager.OrderCompleted

Schema details	
Schema name awsserverlessda.serverlesspresso@OrderMan ager.OrderCompleted	Last modified Sep 24, 2021, 09:31 AM PDT
Description -	Schema ARN arn:aws:schemas:us-west-2:468083054740:schema/discovered-schemas/awsserverlessda. serverlesspresso@OrderManager.OrderCompleted
	Schema registry discovered-schemas
	Number of versions 71
	Schema type OpenAPI 3.0

Version 71 Created on Sep 24, 2021, 09:31 AM PDT

```
1 {
2   "openapi": "3.0.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "OrderManager.OrderCompleted"
6   },
7   "paths": {},
8   "components": {
9     "schemas": {
10       "AWSEvent": {
11         "type": "object",
12         "required": ["detail-type", "resources", "detail", "id", "source", "time", "region", "version", "account"],
13         "x-amazon-events-detail-type": "OrderManager.OrderCompleted",
14         "x-amazon-events-source": "awsserverlessda.serverlesspresso",
15         "properties": {
16           "detail": {
17             "$ref": "#/components/schemas/OrderManager.OrderCompleted"
18         },
19         "account": {
20           "type": "string"
21         }
22       }
23     }
24   }
25 }
```





Amazon EventBridge schema registry

awsserverlessda.serverlesspresso@OrderManager.OrderCompleted

Delete all versions

Schema details

Schema name	Last modified	Schema ARN
awsserverlessda.serverlesspresso@OrderMan	Sep 24, 2021, 09:31 AM PDT	<input type="checkbox"/> arn:aws:schemas:us-west-2:468083054740:schema/discovered-schemas/awsserverlessda.
ager.OrderCompleted		serverlesspresso@OrderManager.OrderCompleted

Description

-

Download code bindings

You can download code bindings for the following languages. Code bindings can be used directly in your code to help develop applications that use events in EventBridge.

It may take several minutes to generate code bindings. Complex events can take up to 5 minutes to generate and download. You can close this pop-up. The download will continue once the code bindings have been generated, unless you close the EventBridge console.

Java 8+ Python 3.6+
 TypeScript 3+ Go 1+

Close **Download**

Version 71 Created on Sep 24, 2021, 09:31 AM PDT

Actions ▾ **Download code bindings**

```
1 {  
2   "openapi": "3.0.0",  
3   "info": {  
4     "version": "1.0.0",  
5     "title": "OrderM  
6   },  
7   "paths": {},  
8   "components": {  
9     "schemas": {  
10       "AWSEvent": {  
11         "type": "obj  
12         "required": [ "detail-type", "Resources", "detail", "ID", "source", "time", "region", "version", "account"],  
13         "x-amazon-events-detail-type": "OrderManager.OrderCompleted",  
14         "x-amazon-events-source": "awsserverlessda.serverlesspresso",  
15         "properties": {  
16           "detail": {  
17             "$ref": "#/components/schemas/OrderManager.OrderCompleted"  
18           },  
19           "account": "  
20         }  
21       }  
22     }  
23   }  
24 }
```

Copy





AsyncAPI

Studio BETA

Information

Servers

KAFKA-SECURE scram-co...

KAFKA-SECURE mtls-con...

Channels

LIGHTINGMEASURED sm...

LIGHTTURNON smartyligh...

LIGHTTURNOFF smartylig...

LIGHTSDIM smartylighting...

Operations

RECEIVE receiveLightMea...

SEND turnOn

SEND turnOff

SEND dimLight

Messages

lightMeasured

turnOnOff

dimLight

Schemas

lightMeasuredPayload

turnOnOffPayload

dimLightPayload

From localStorage

```
1  asyncapi: 3.0.0
2  info:
3    title: Streetlights Kafka API
4    version: 1.0.0
5    description: |-  
        The Smartylighting Streetlights API allows you to remotely manage the city  
        lights.  
        !!! Check out its awesome features:  

6
7        * Turn a specific streetlight on/off 🌈
8        * Dim a specific streetlight 😊
9        * Receive real-time information about environmental lighting conditions ✅
10
11    license:  
        name: Apache 2.0
12        url: https://www.apache.org/licenses/LICENSE-2.0
13    defaultContentType: application/json
14    servers:  
        scram-connections:  
            host: test.mykafkacluster.org:18092
15            protocol: kafka-secure
16            description: Test broker secured with scramSha256
17            security:  
                - $ref: '#/components/securitySchemes/scramScram'
18            tags:  
                - name: env:test-scram
19                description: >-
20                    This environment is meant for running internal tests through
21                    scramSha256
22                - name: kind:remote
23                description: This server is a remote server. Not exposed by the application
24                - name: visibility:private
25                description: This resource is private and only available to certain users
26            mtls-connections:  
                host: test.mykafkacluster.org:28092
27                protocol: kafka-secure
28                description: Test broker secured with X509
29                security:  
                    - $ref: '#/components/securitySchemes/certs'
30
31
32
33
34
35
36
37
38
```

VALID AUTOSAVE: ON YAML

0 0 0 0 Filter diagnostics...

Type Line Message

No issues.

APACHE 2.0 APPLICATION/JSON

The Smartylighting Streetlights API allows you to remotely manage the city lights.

Check out its awesome features:

- Turn a specific streetlight on/off 🌈
- Dim a specific streetlight 😊
- Receive real-time information about environmental lighting conditions ✅

Servers

kafka-secure://test.mykafkacluster.org:18092 KAFKA-SECURE SCRAM-CONNECTIONS

Test broker secured with scramSha256

Security:

ScramSha256

Provide your username and password for SASL/SCRAM authentication

SECURITY.PROTOCOL: SASL_SSL

SASL.MECHANISM: SCRAM-SHA-256

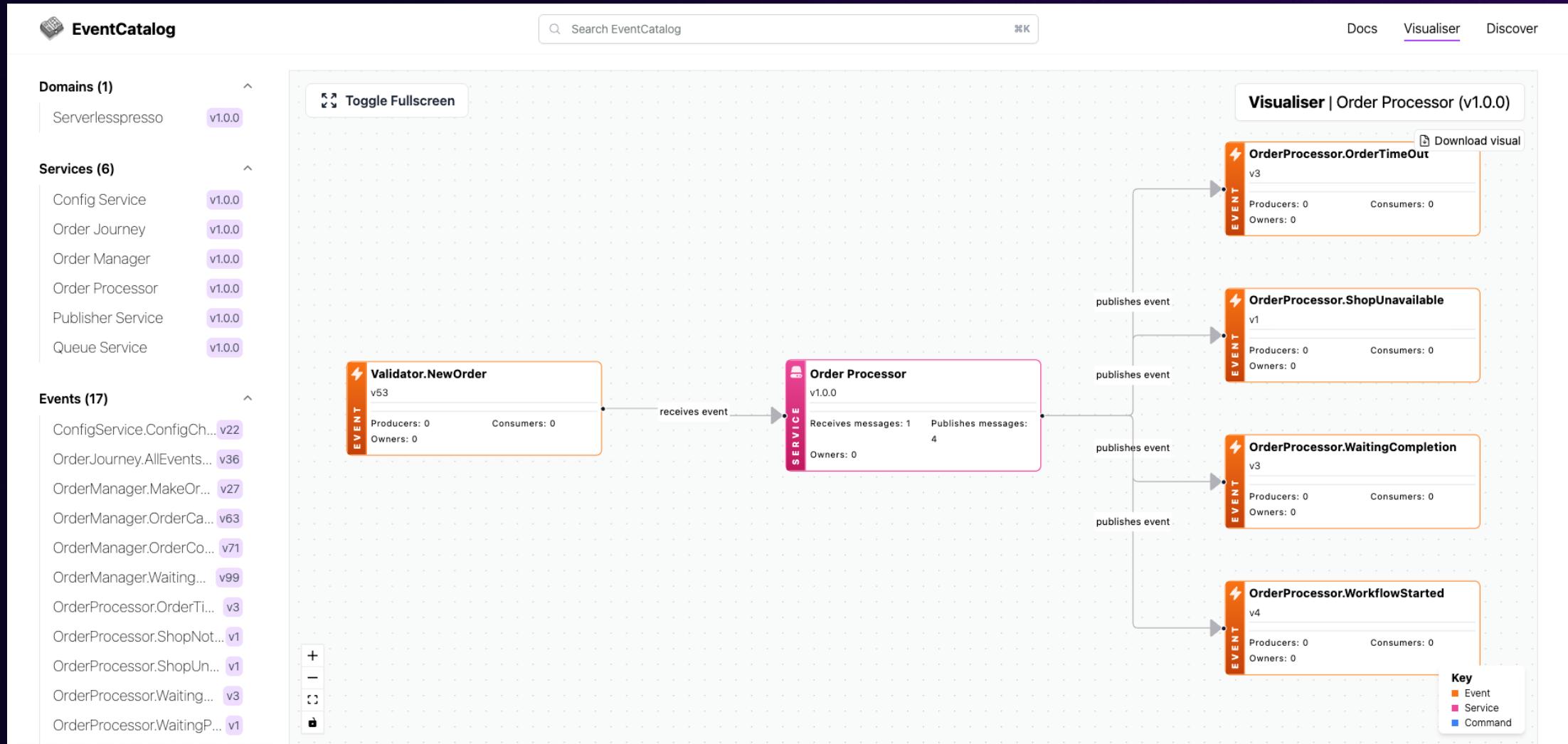
#env:test-scram #kind:remote #visibility:private

kafka-secure://test.mykafkacluster.org:28092 KAFKA-SECURE MTLS-CONNECTIONS



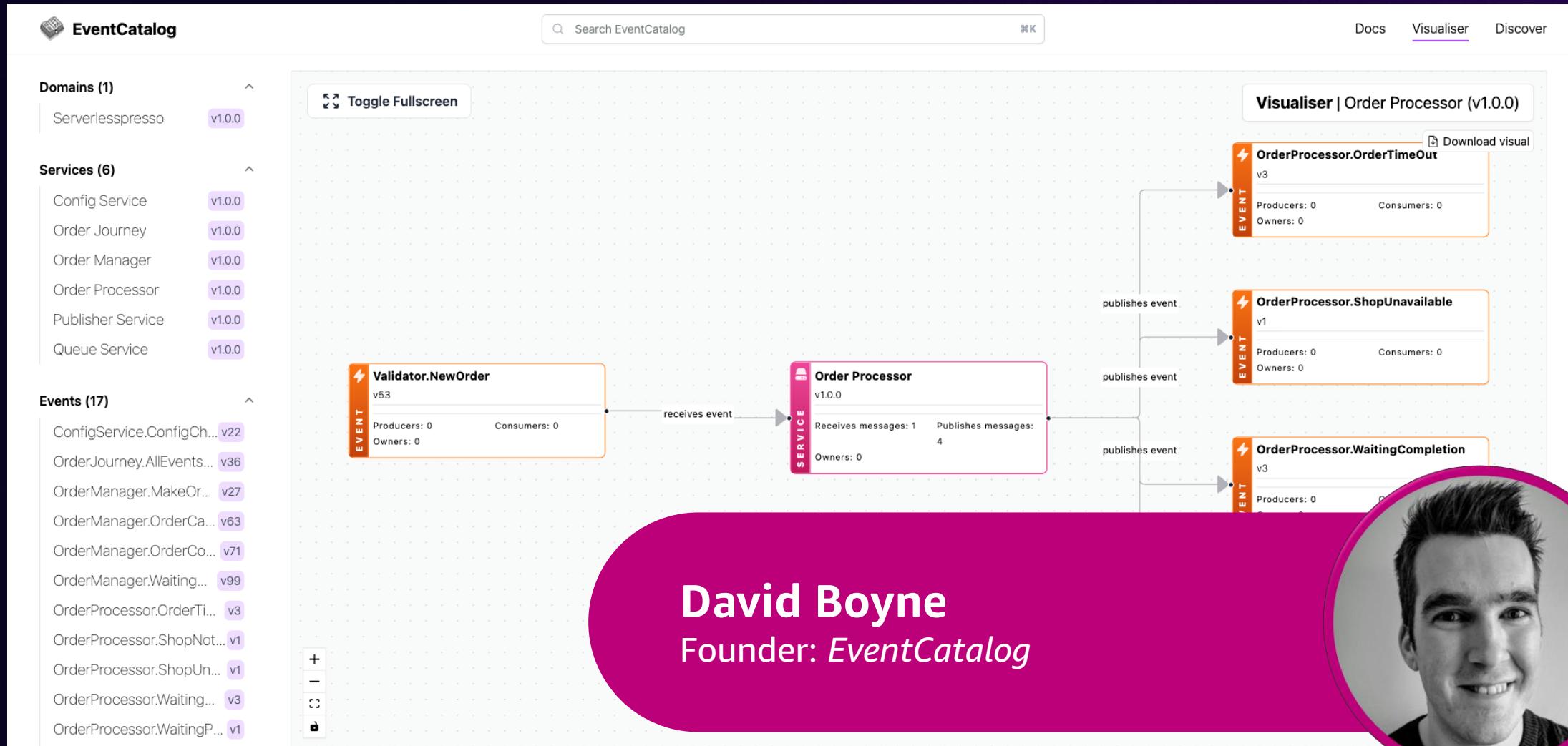


EventCatalog



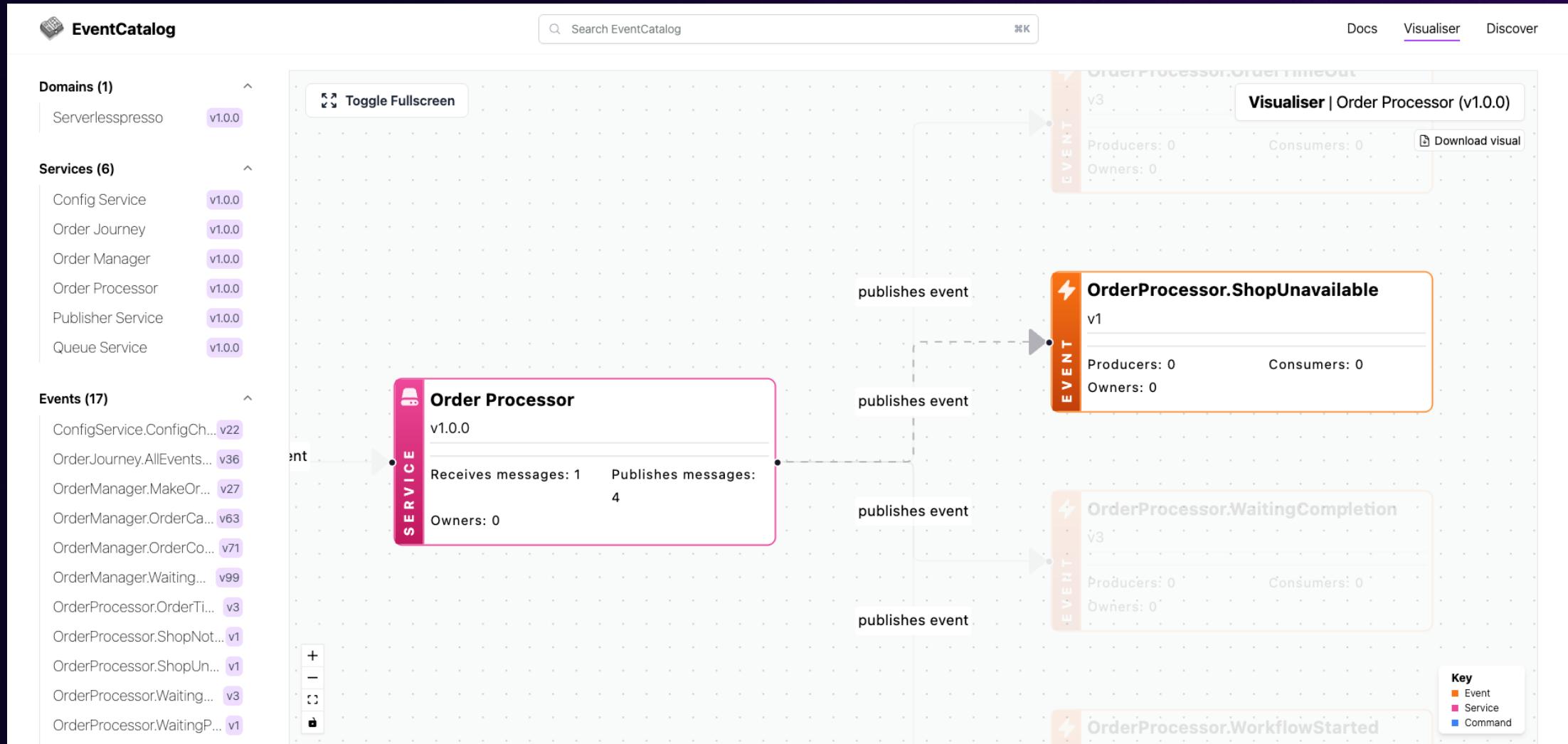


EventCatalog





EventCatalog





EventCatalog

EventCatalog

Architecture diagram

Messages

- ConfigService.ConfigChanged (v22)**
 - Overview
 - How is this event used?
 - Schemas
 - JSON Schema
 - OpenAPI Schema
- OrderJourney.AllEvents (v36)**
 - Overview
 - How is this event used?
 - Schemas
 - JSON Schema
 - OpenAPI Schema
- OrderManager.MakingOrder (v27)**
 - Overview
 - How is this event used?
 - Schemas
 - JSON Schema
 - OpenAPI Schema
- OrderManager.OrderPlaced (v63)**
 - Overview
 - How is this event used?
 - Schemas
 - JSON Schema
 - OpenAPI Schema
- OrderManager.OrderShipped (v71)**
 - Overview
 - How is this event used?
 - Schemas
 - JSON Schema
 - OpenAPI Schema

Search EventCatalog

Event

Source: awsserverlessda.serverlesspresso

DetailType: ConfigService.ConfigChanged

Schema name: awsserverlessda.serverlesspresso@ConfigService.ConfigChanged

ConfigService.ConfigChanged (v22)

Overview

Documentation for the Amazon EventBridge event ConfigService.ConfigChanged.

Open schema registry
Open the discovered-schemas registry in the AWS console

View schema in the AWS console
Open the schema for ConfigService.ConfigChanged directly in the AWS console

Download code bindings
Download Java, Python, TypeScript and Go code bindings for ConfigService.ConfigChanged

Download schema
Download the schema for this event

How is this event used?

Download visual

Docs Visualiser Discover

Event Producers (1)

Config Service (service)
v1.0.0

Event Consumers (1)

Order Journey (service)
v1.0.0

Event Owners (0)

This event does not have any documented owners.

Versions (1)

v22 (latest)
View changelogs

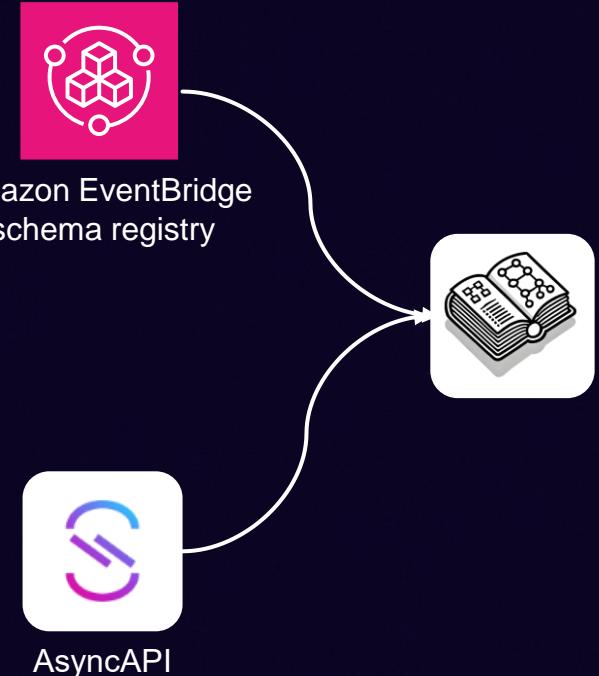
Download Schema

View in Visualiser

Changelog



Combining



- **AsyncAPI generator**
- **Amazon EventBridge generator**

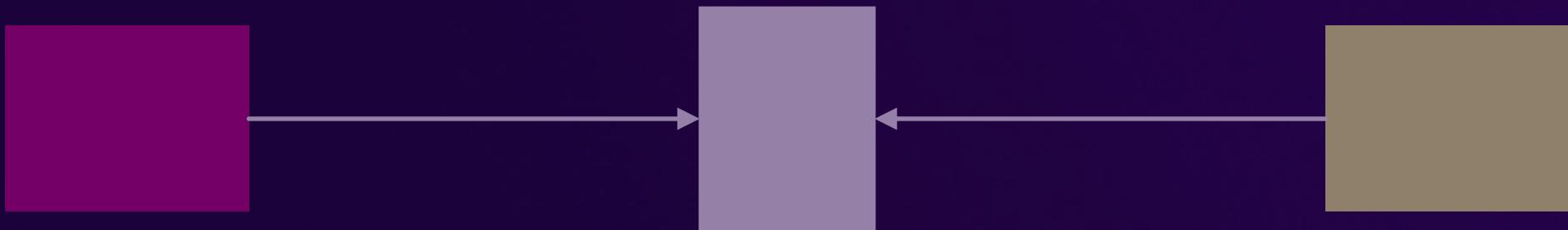
```
Processing domain: Serverlesspresso (v1.0.0)
  - Domain (v1.0.0) already exists, skipped creation...
Processing service: Order Processor (v1.0.0)
Processing Order Journey
Processing event: OrderJourney.AllEventsStored (v36)
  - Event (OrderJourney.AllEventsStored v36) created
  - Schema added to event (v36)
  - Schema added to event (v36)
Processing event: ConfigService.ConfigChanged (v22)
  - Event (ConfigService.ConfigChanged v22) created
  - Schema added to event (v22)
  - Schema added to event (v22)
Processing event: OrderJourney.AllEventsStored (v36)
  - Event (OrderJourney.AllEventsStored v36) created
  - Schema added to event (v36)
  - Schema added to event (v36)
Processing event: OrderManager.MakeOrder (v27)
  - Event (OrderManager.MakeOrder v27) created
  - Schema added to event (v27)
  - Schema added to event (v27)
Processing event: OrderManager.OrderCancelled (v63)
  - Event (OrderManager.OrderCancelled v63) created
  - Schema added to event (v63)
  - Schema added to event (v63)
Processing event: OrderManager.OrderCompleted (v71)
  - Event (OrderManager.OrderCompleted v71) created
  - Schema added to event (v71)
  - Schema added to event (v71)
Processing event: OrderManager.WaitingCompletion (v99)
  - Event (OrderManager.WaitingCompletion v99) created
  - Schema added to event (v99)
  - Schema added to event (v99)
Processing event: OrderProcessor.OrderTimeOut (v3)
  - Event (OrderProcessor.OrderTimeOut v3) created
  - Schema added to event (v3)
  - Schema added to event (v3)
Processing event: OrderProcessor.ShopNotready (v1)
  - Event (OrderProcessor.ShopNotready v1) created
  - Schema added to event (v1)
  - Schema added to event (v1)
Processing event: OrderProcessor.ShopUnavailable (v1)
  - Event (OrderProcessor.ShopUnavailable v1) created
```

Observability



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking about observability in EDAs



Health of the
producer

Health of the
broker

Health of the
consumer

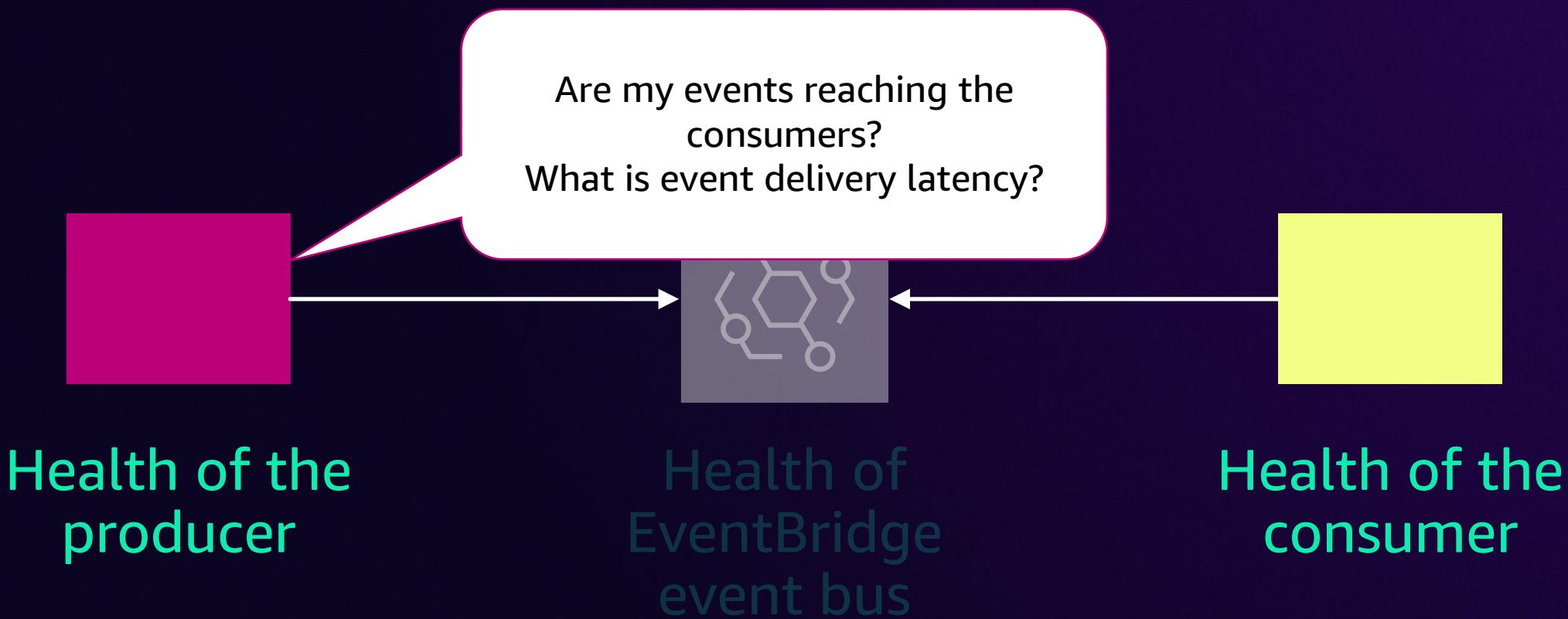
Health of the system(s)

Shared responsibility model

AWS managed



Thinking about observability in EDAs

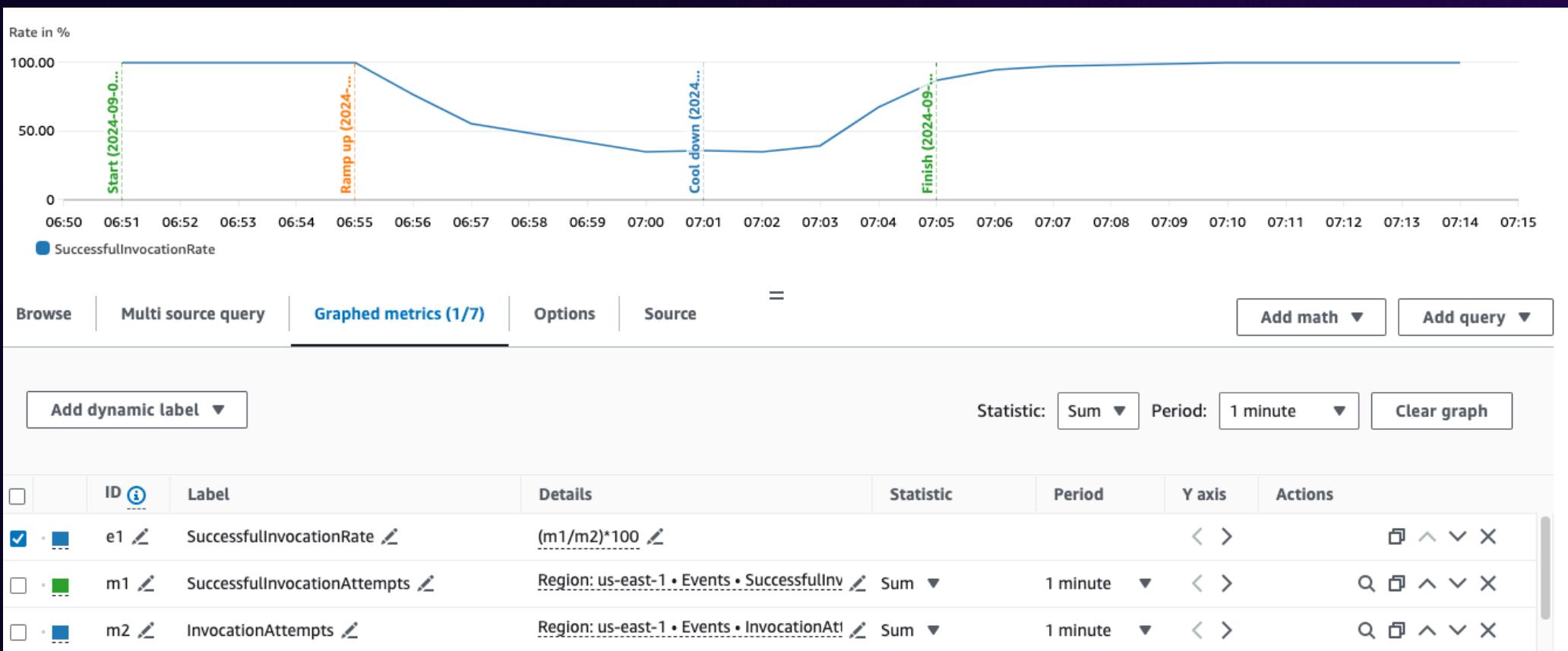


Monitoring EventBridge

Metric	Dimension	Units
InvocationAttempts	EventBusName, None, RuleName	Count
SuccessfulInvocationAttempts	EventBusName, None, RuleName	Count
RetryInvocationAttempts	EventBusName, None, RuleName	Count
MatchedEvents	EventBusName, None, RuleName	Count

Amazon CloudWatch metric math

`SuccessfulInvocationRate = SuccessfulInvocationAttempts/InvocationAttempts`



Monitoring delivery failures

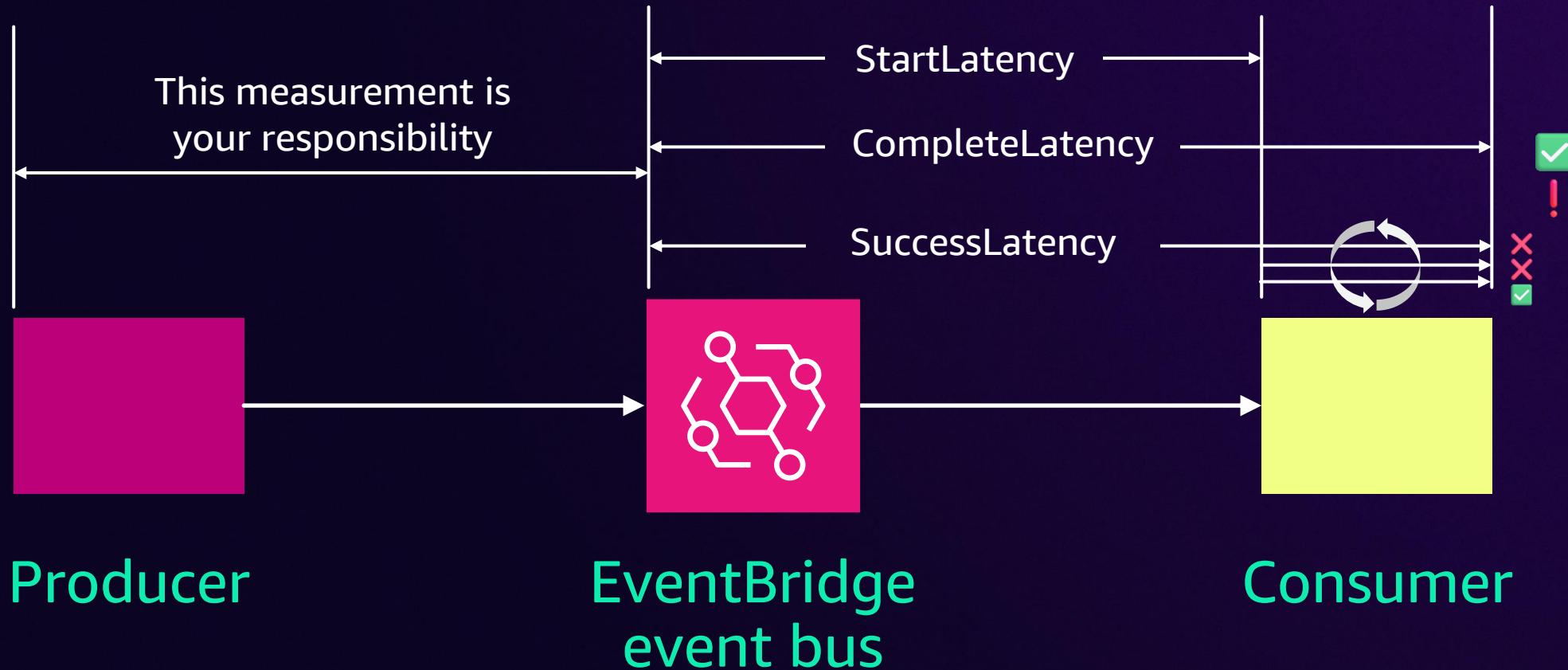
Metric	Dimension	Units
FailedInvocations	RuleName	Count
InvocationsSentToDlq	RuleName	Count
InvocationsFailedToBeSentToDlq	RuleName	Count

EventBridge will retry delivery for up to 185 times over 24 hours by default

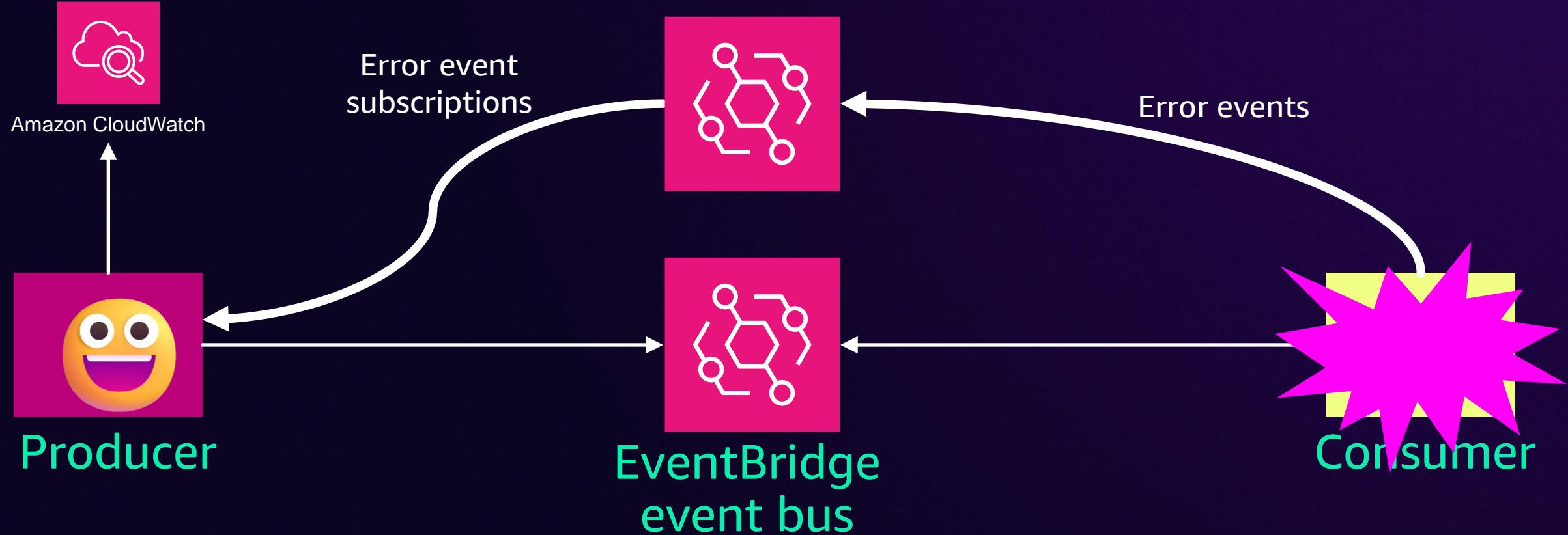
Detecting delivery delays

Metric	Dimension	Units
IngestiontoInvocationCompleteLatency	EventBusName, None, RuleName	Milliseconds
IngestionToInvocationSuccessLatency	EventBusName, None, RuleName	Milliseconds
IngestiontoInvocationStartLatency	EventBusName, None, RuleName	Milliseconds

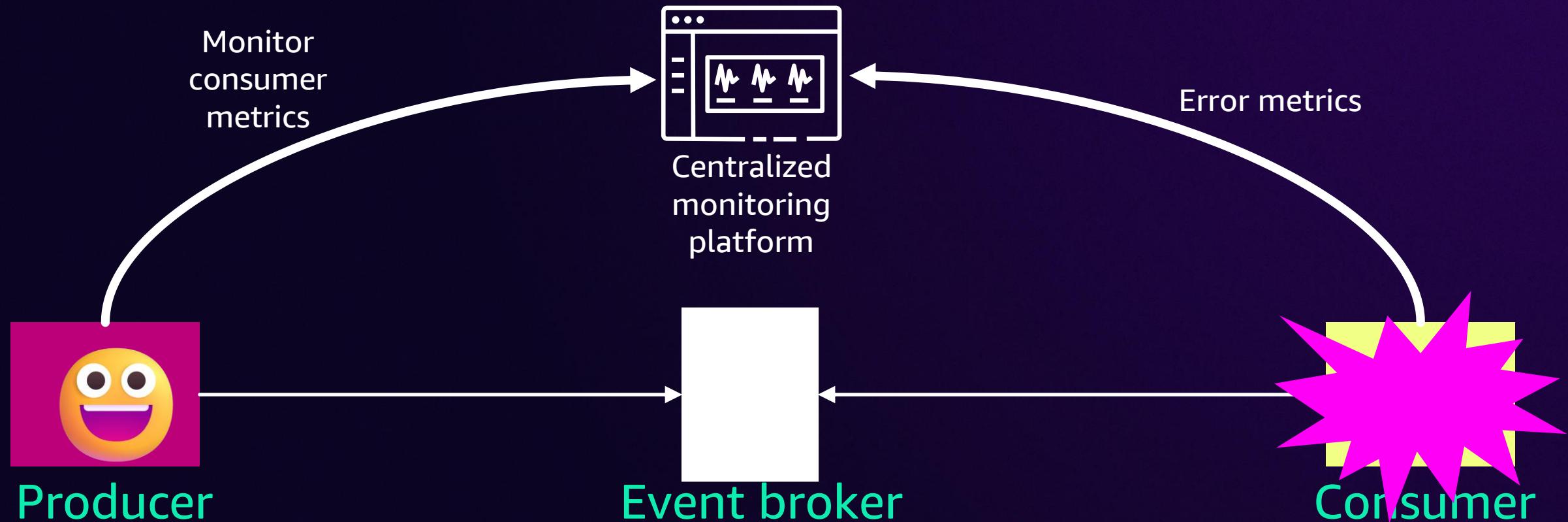
Ingestion to Invocation



Reporting client errors



Reporting client errors



Wrap



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Takeaways

- Reduce coupling as much as possible
- Enforce schemas
- Understand and plan for your events and event evolution (use the tools)
- Start with observability in mind

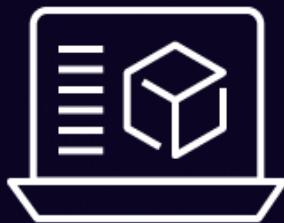
Resources



<https://s12d.com/api307-24>

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

<https://s12d.com/serverless-learning>

Earn AWS
Serverless badge



Demonstrate your
knowledge by achieving
digital badges



Check out these other sessions

COP411: Monitoring event flows: Observability for event-driven architectures

Mon, Dec 2 12:00 Mandalay Bay Lower Level North South Pacific D

SVS339 Building event-driven architectures using Amazon ECS with AWS Fargate

Tue, Dec 3 13:30 Mandalay Bay Lower Level North Islander G

ARC329-R1: Modernizing microservices with event-driven architecture

Wed, Dec 4 14:30 MGM Grand Level 3 307

Thank you!

Eric Johnson

@edjgeek

Brian Zambrano

@brianzambrano

zambb@amazon.com



Please complete the session
survey in the mobile app