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

API307

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

When we say “scale” we usually think “traffic”

We’ve got you covered

SOME NUMBERS

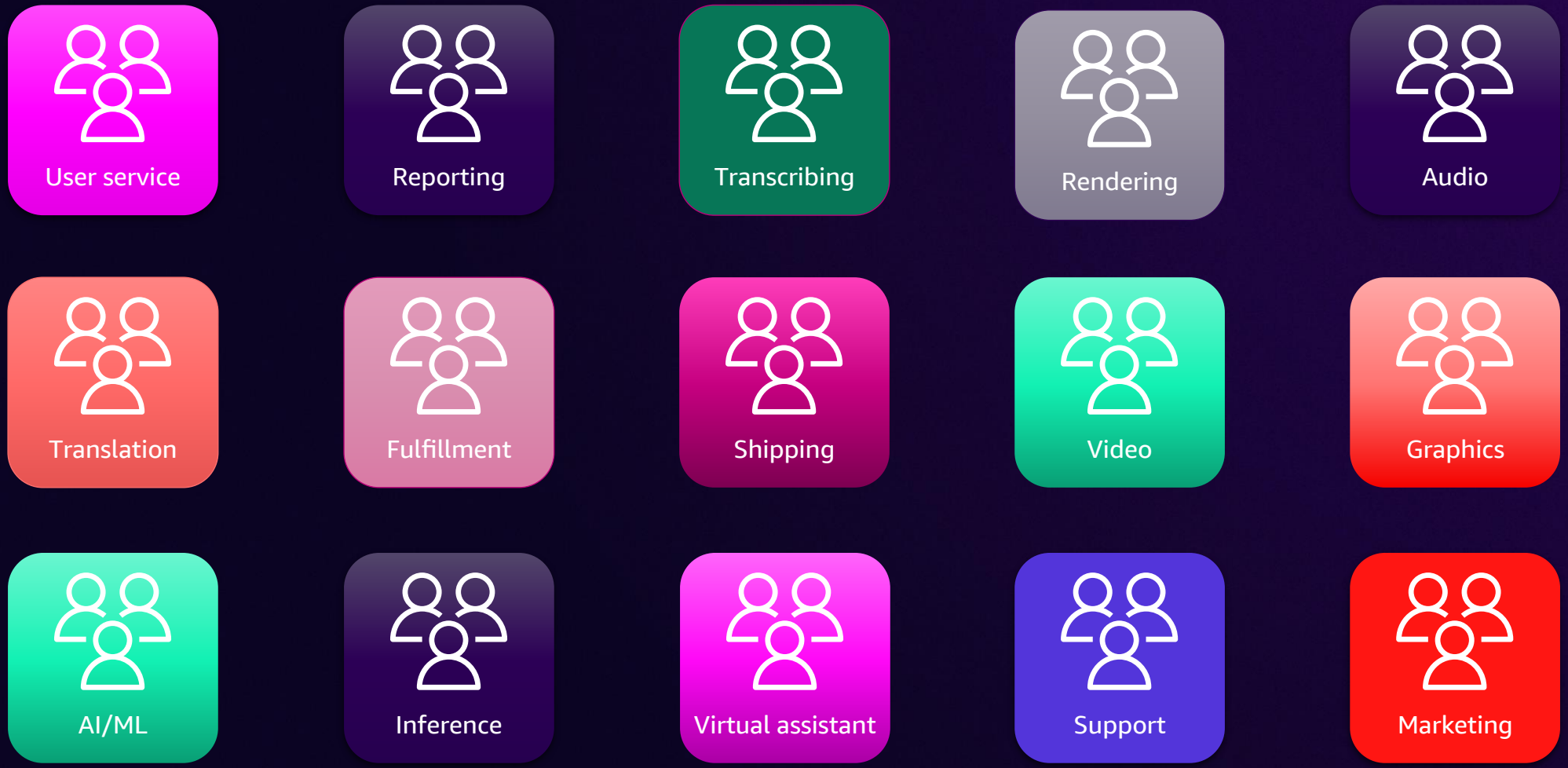
- During Prime Day 2024, Amazon DynamoDB peaked at 146 million requests per second
- AWS Lambda handled over 1.3 trillion invocations during Prime Day 2024
- AWS Lambda processes tens of trillions of requests each month
- Amazon Simple Queue Service (Amazon SQS) processes over 100 million messages per 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

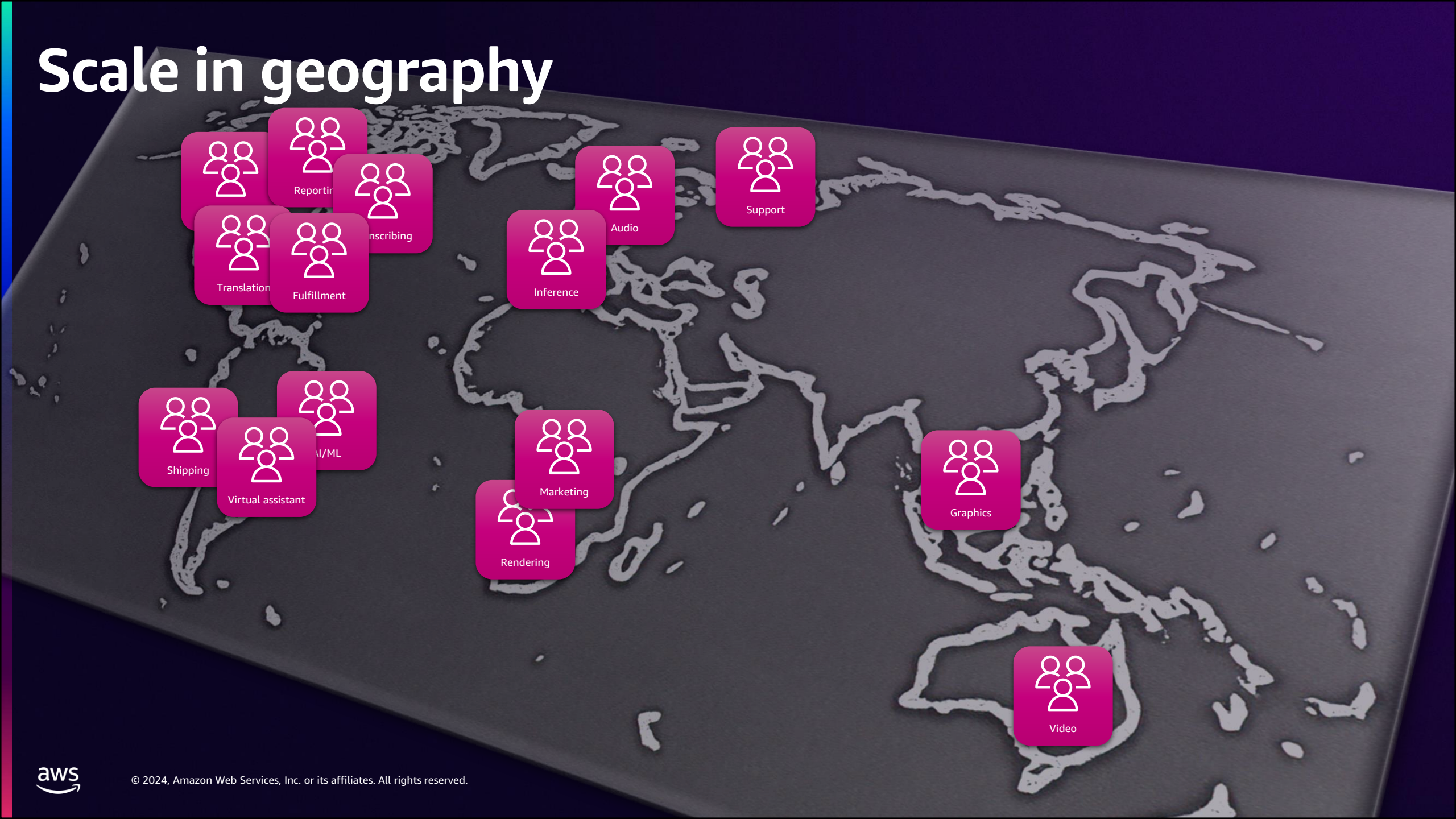
Successful EDAs do not generally look like this

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

And

Something happens

We react

Images created by Amazon Titan Generator G1

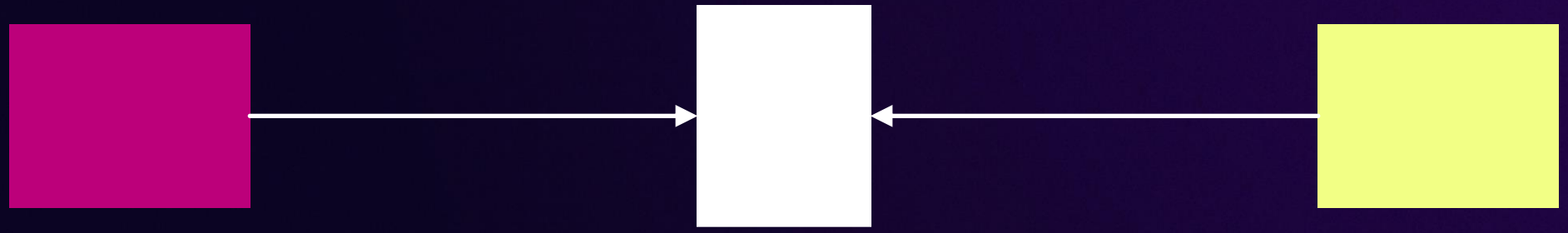
What is an event?

- 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.

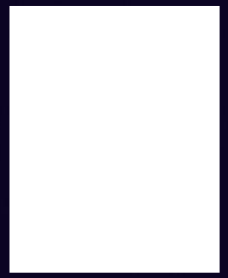
Event broker

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

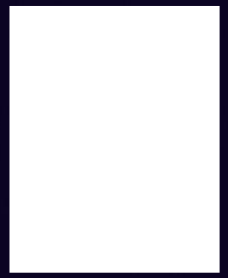
- Amazon Kinesis
- Apache Kafka
- Amazon Managed Streaming for Apache Kafka (MSK)

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.

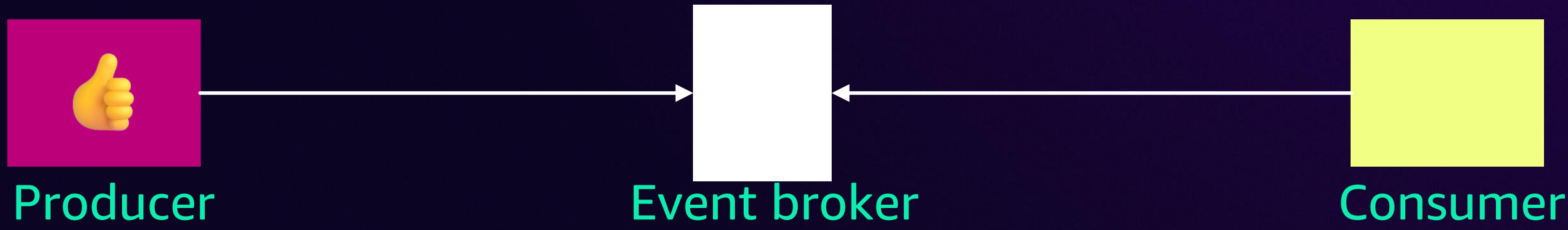
Push-based

EventBridge

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

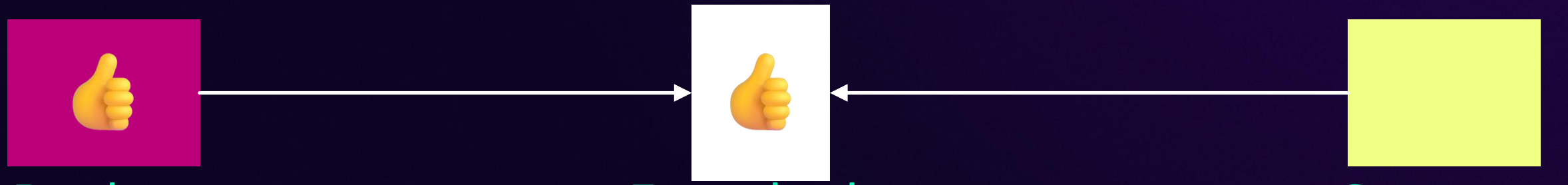
Loose coupling


```
{
  "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"
  }
}
```

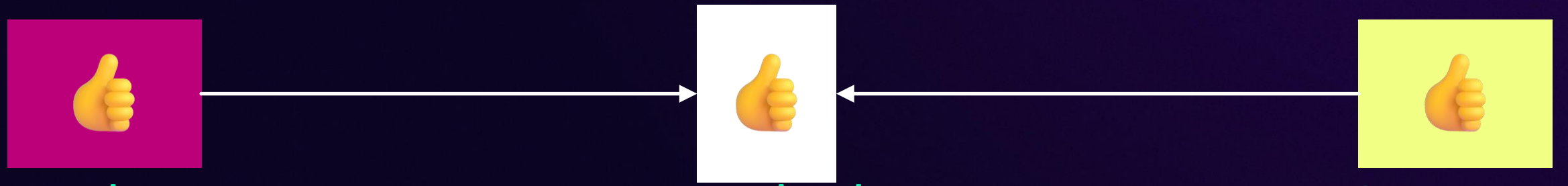
Loose coupling

```
{
  "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"
  }
}
```



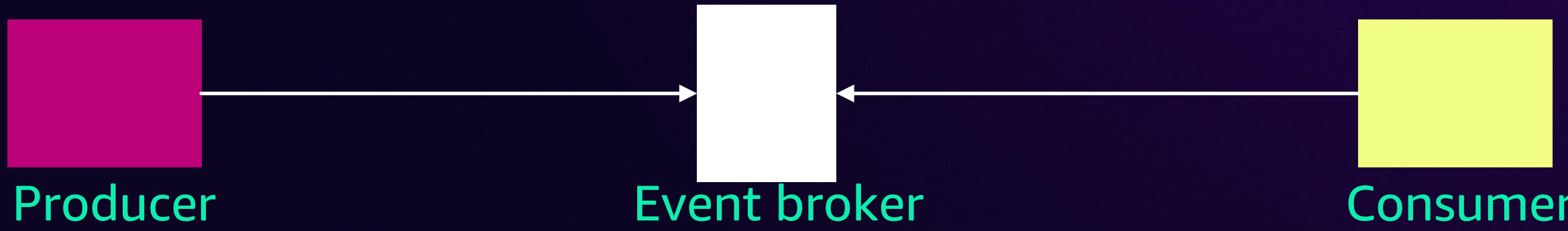
Loose coupling


```
{
  "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"
  }
}
```

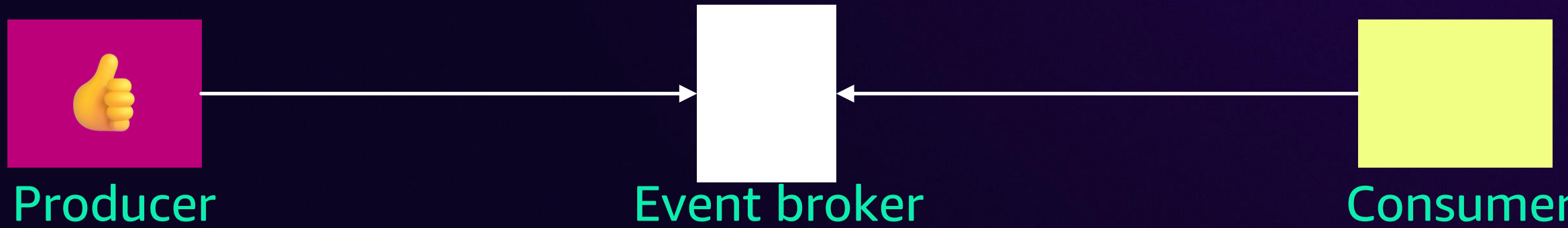
Loose coupling++

```
{
  "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"
  }
}
```



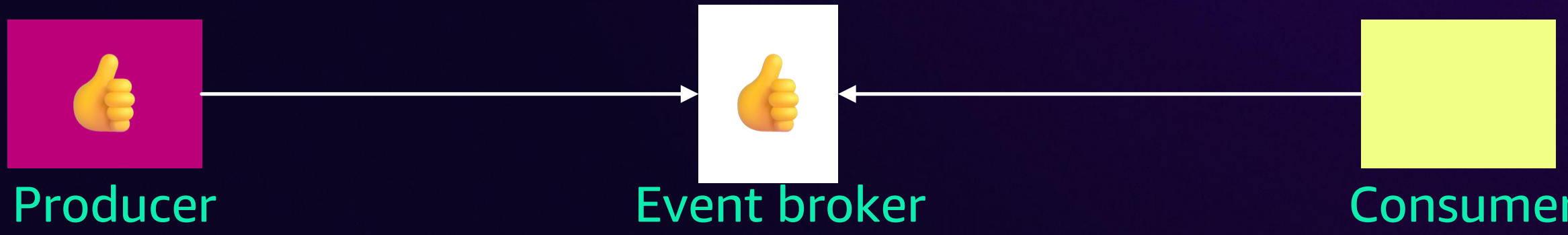
Loose coupling++


```
{
  "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"
  }
}
```

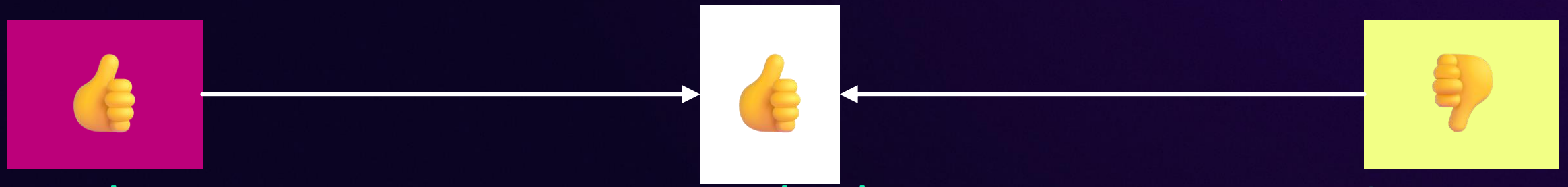



Loose coupling++

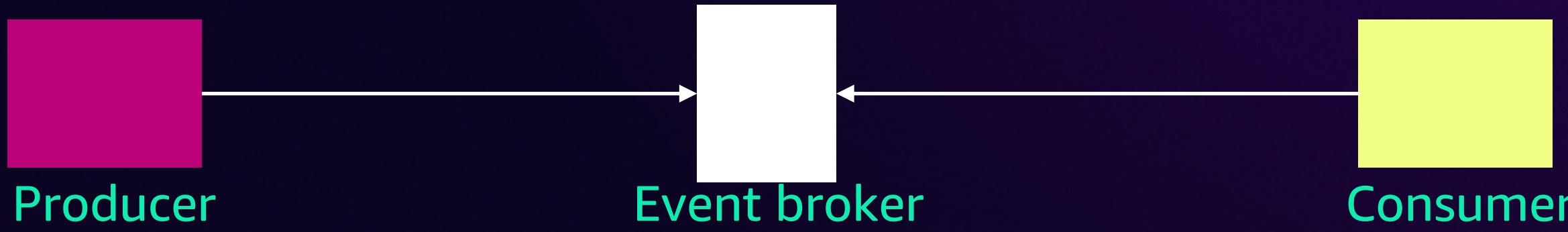

```
{
  "eventType": "OrderCreated",
  "eventId": "c4d7e9c1-2e1d",
  "timestamp": "2023-05-01T14:25:43.123Z",
  "order": {
    "id": "ORD-12345",
    "timestamp": "2023-05-01T…",
  },
  "customerDetails": {
    "email": "customer@example.com",
    "name": "John Doe",
    "phone": null,
    "address":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

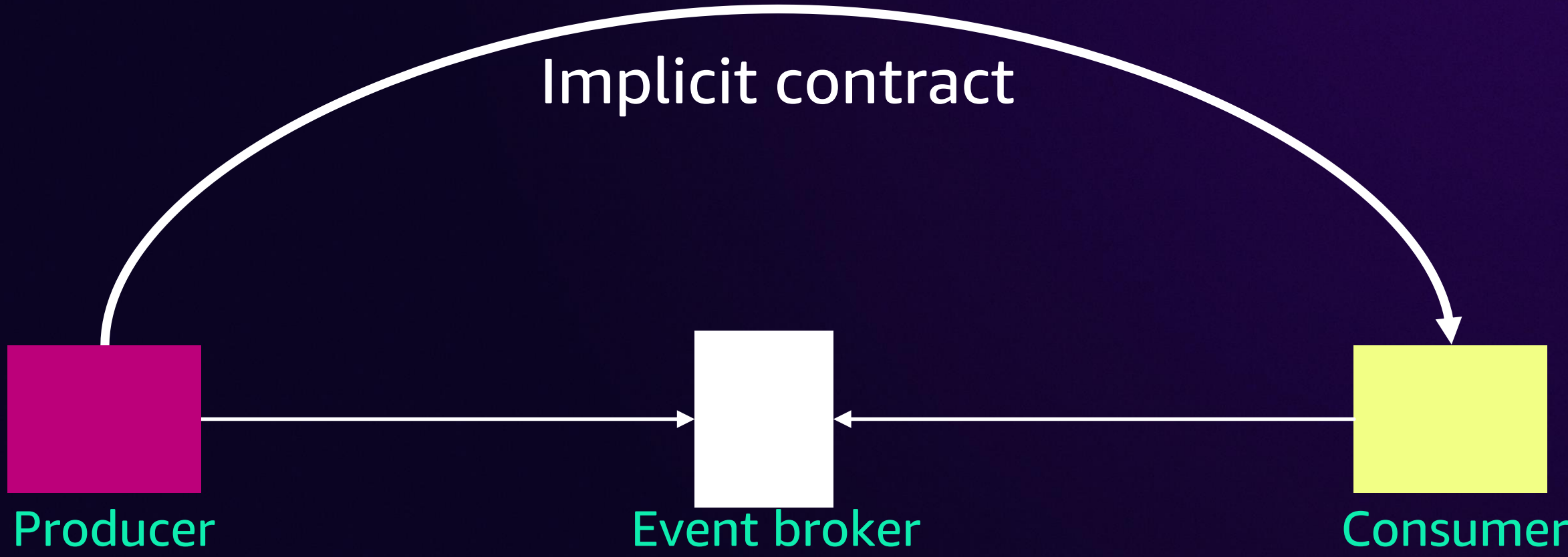
Loose coupling++



Loose coupling != No coupling

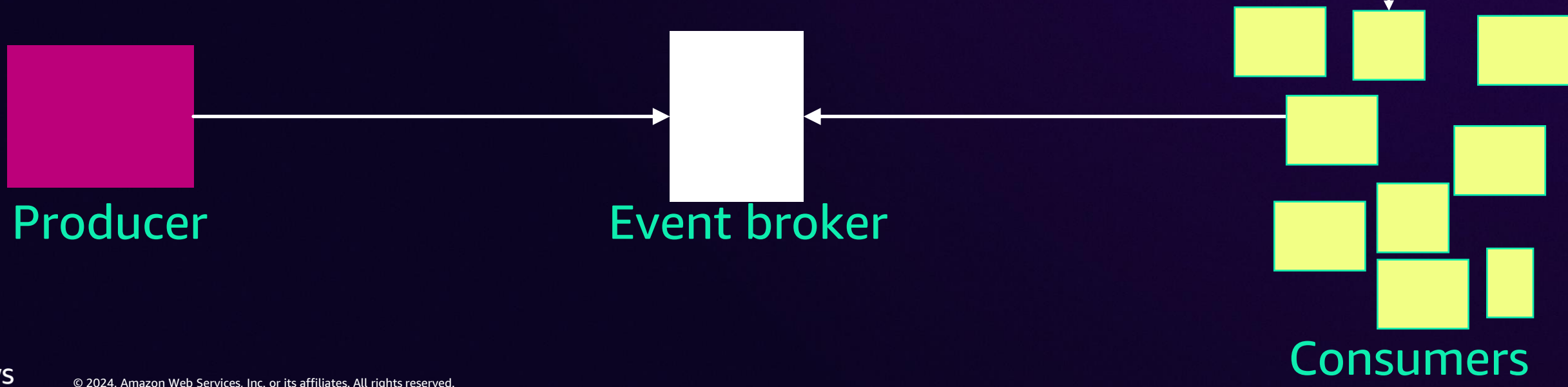


Loose coupling != No coupling

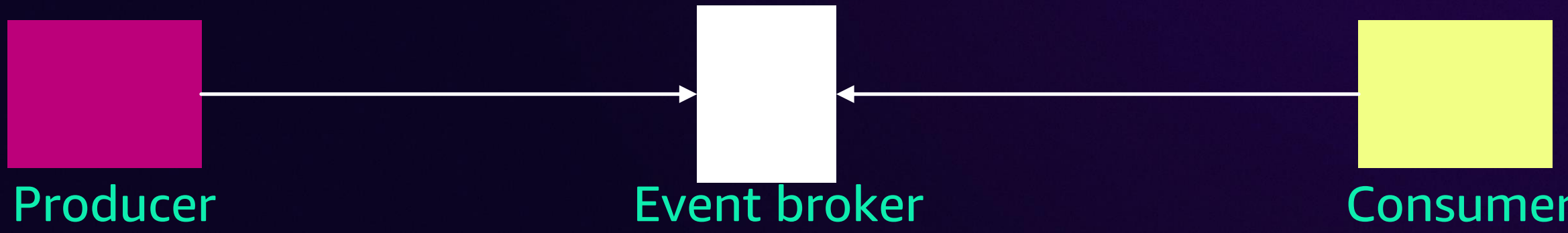


Loose coupling != No coupling

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

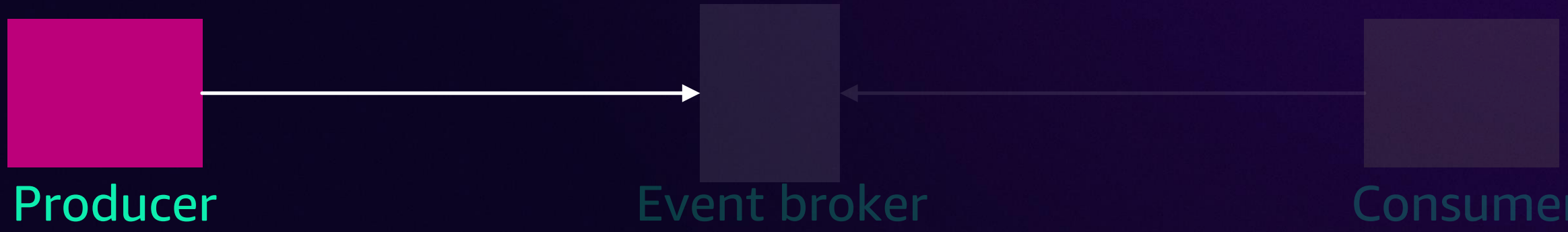


Options



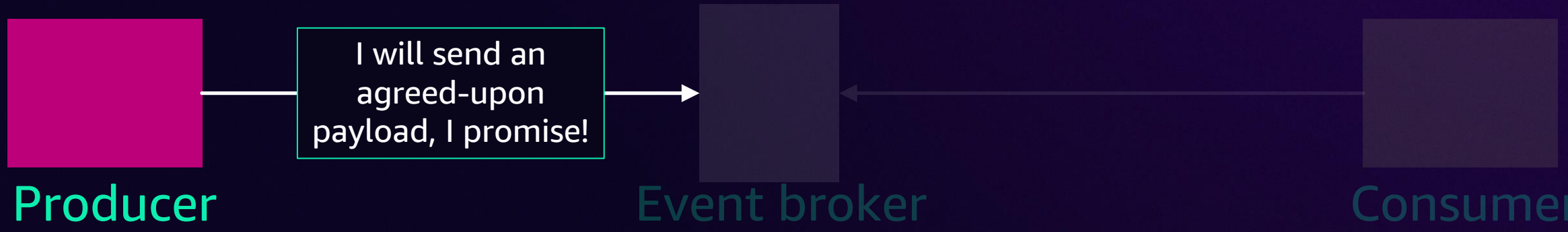
Options

Promises from the producers



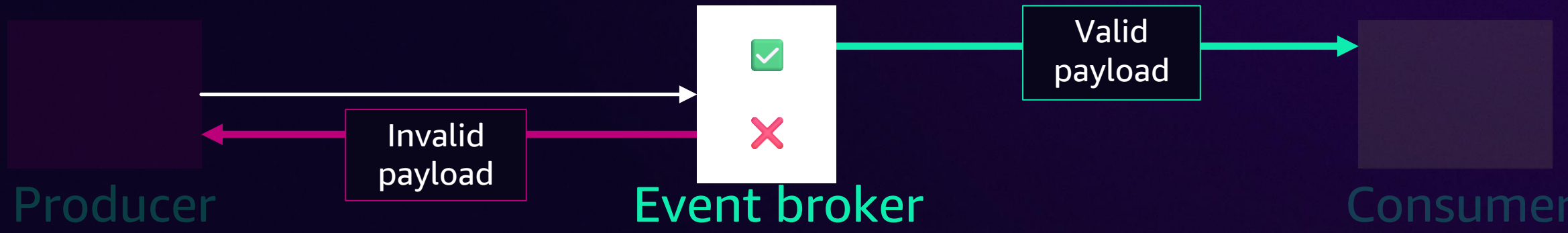
Options

Promises from the producers

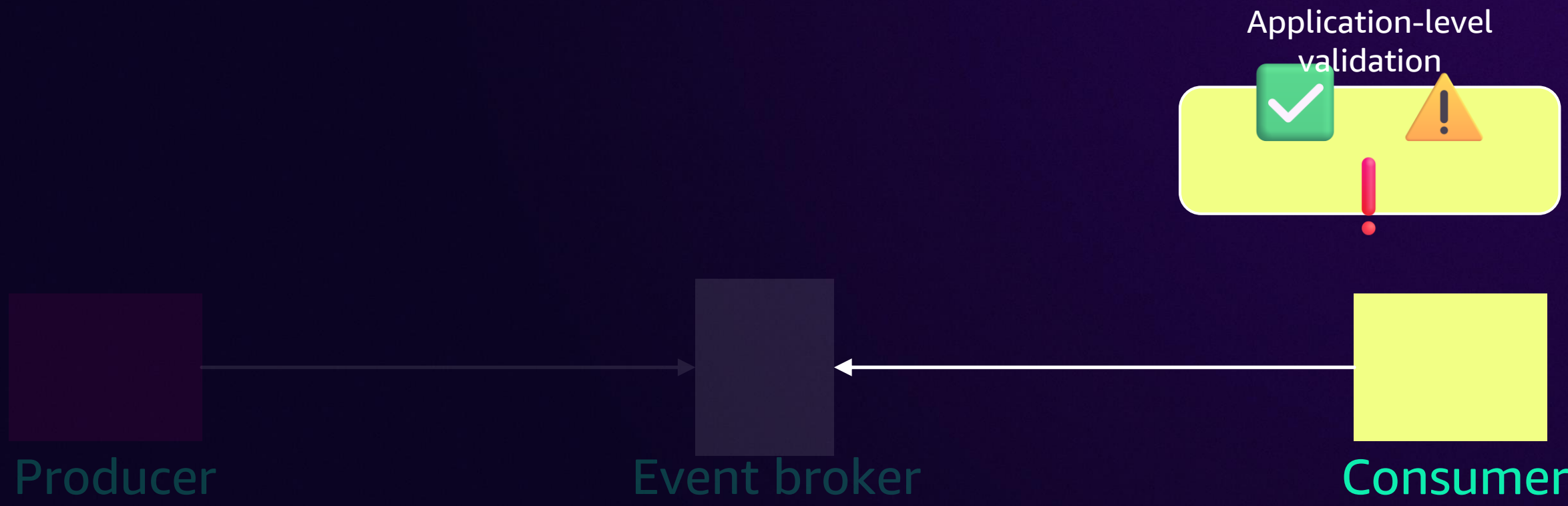


Options

Enforcement



Options

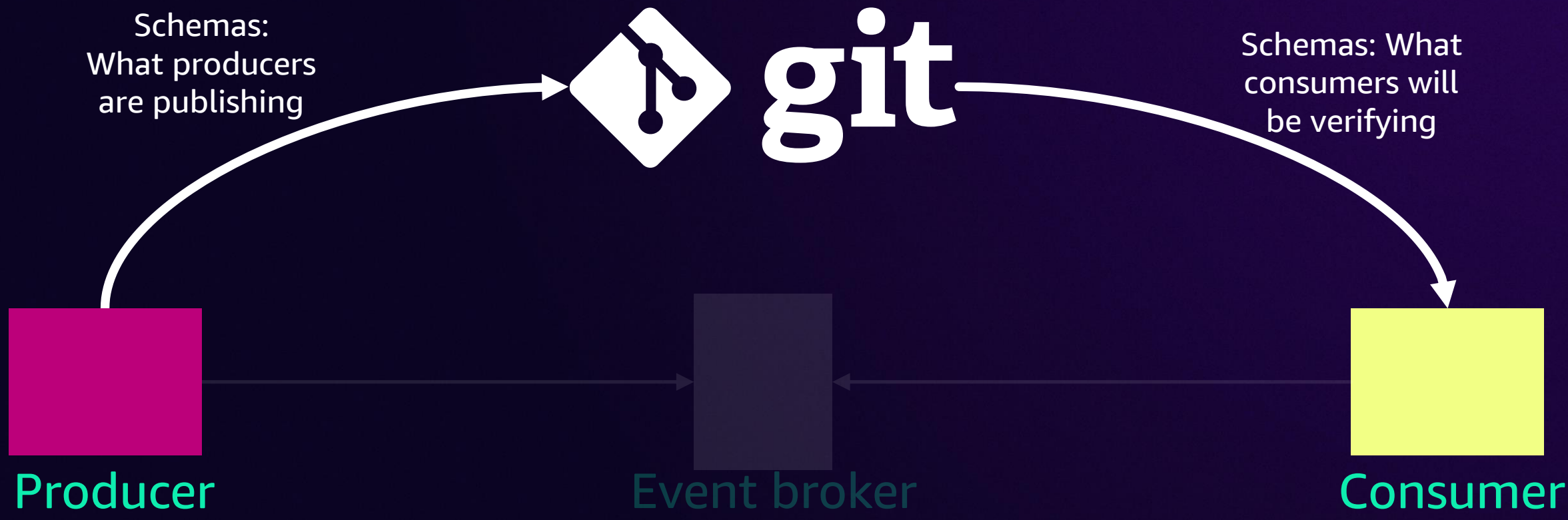


JSON Schema

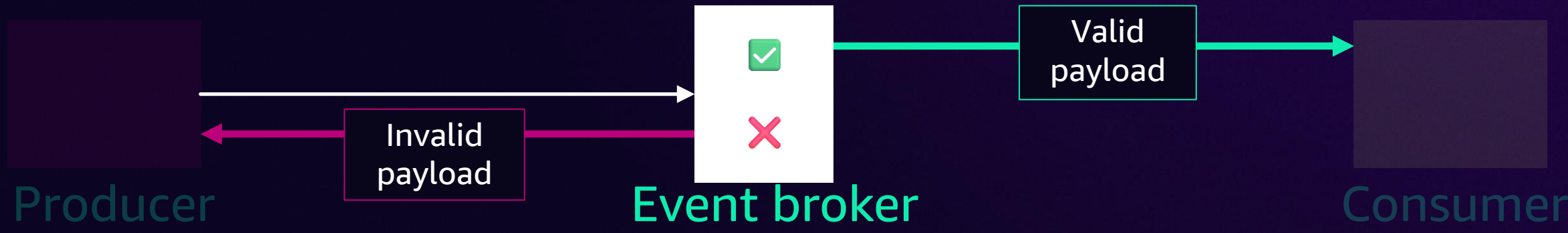
- Defines the structure and format of JSON data, allowing you to describe expected properties, types, and constraints
- Helps to maintain data integrity and improve reliability by preventing issues like missing or additional fields or invalid types in API payloads

```
{
 "$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

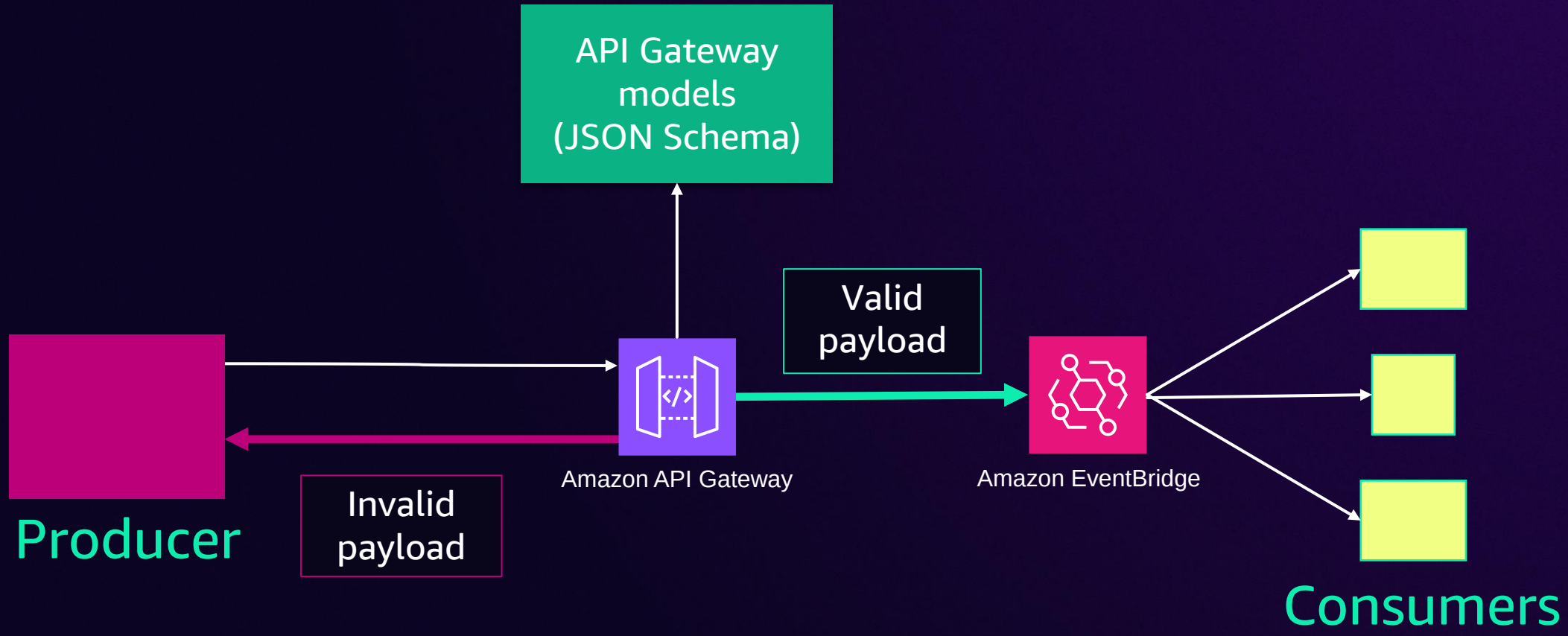


Let’s look at enforcement

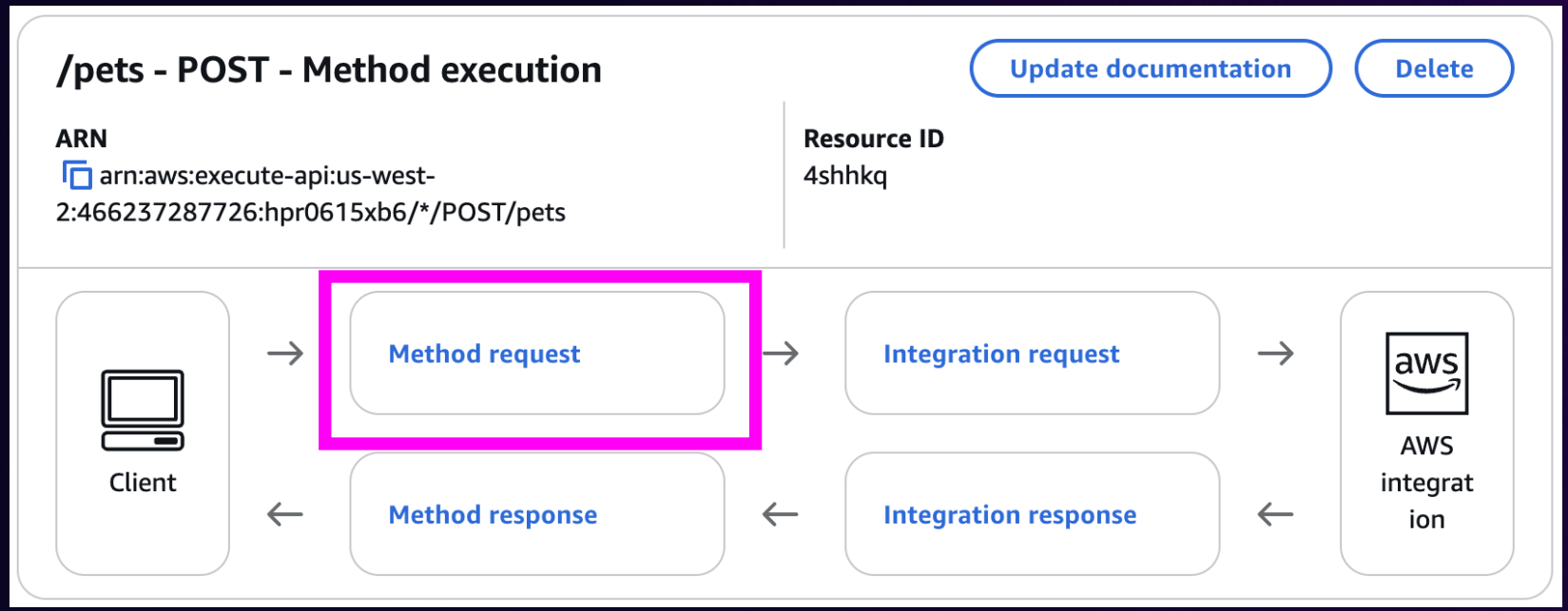


Schema enforcement with API Gateway

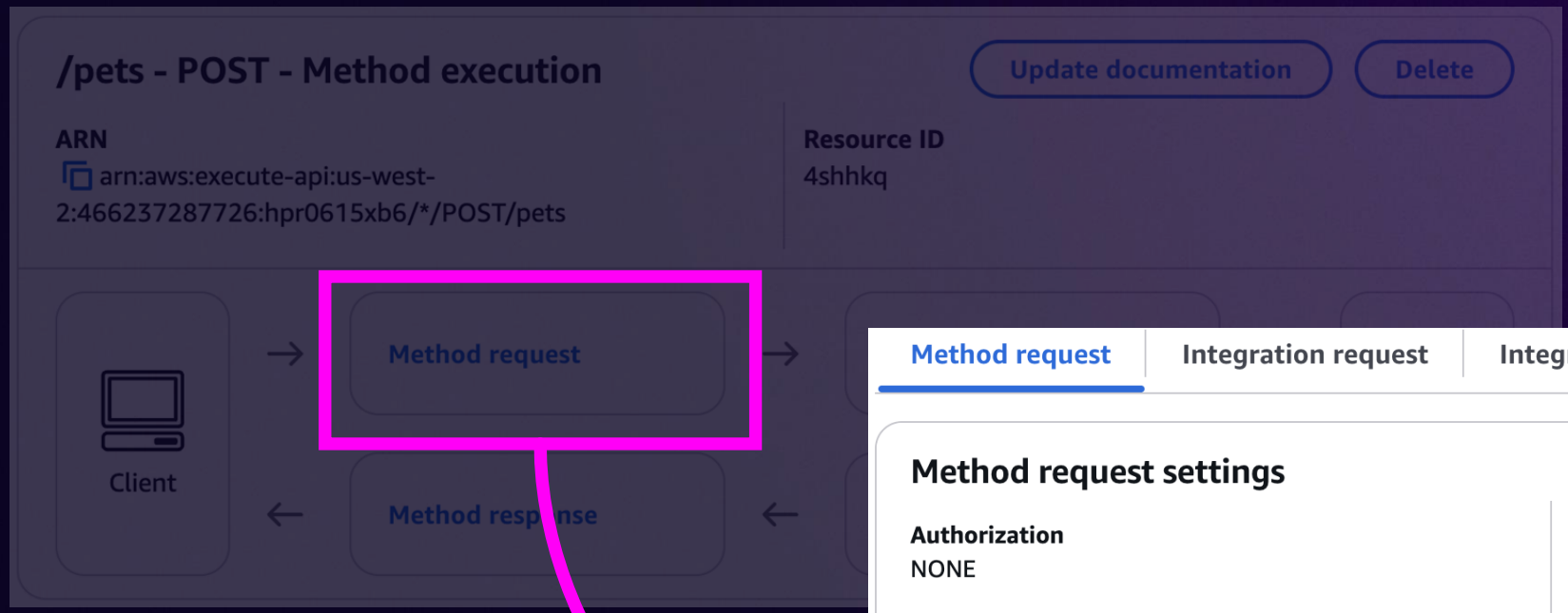
Schema enforcement with API Gateway

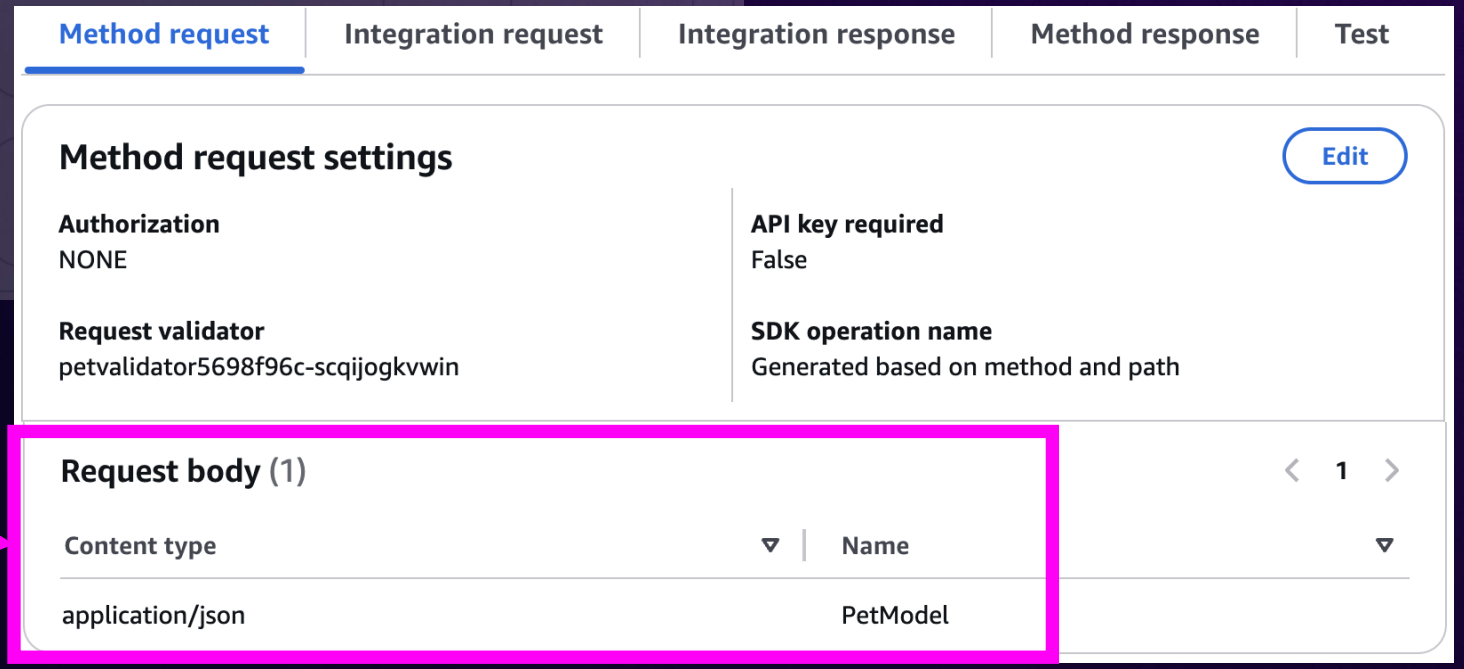


API Gateway method request

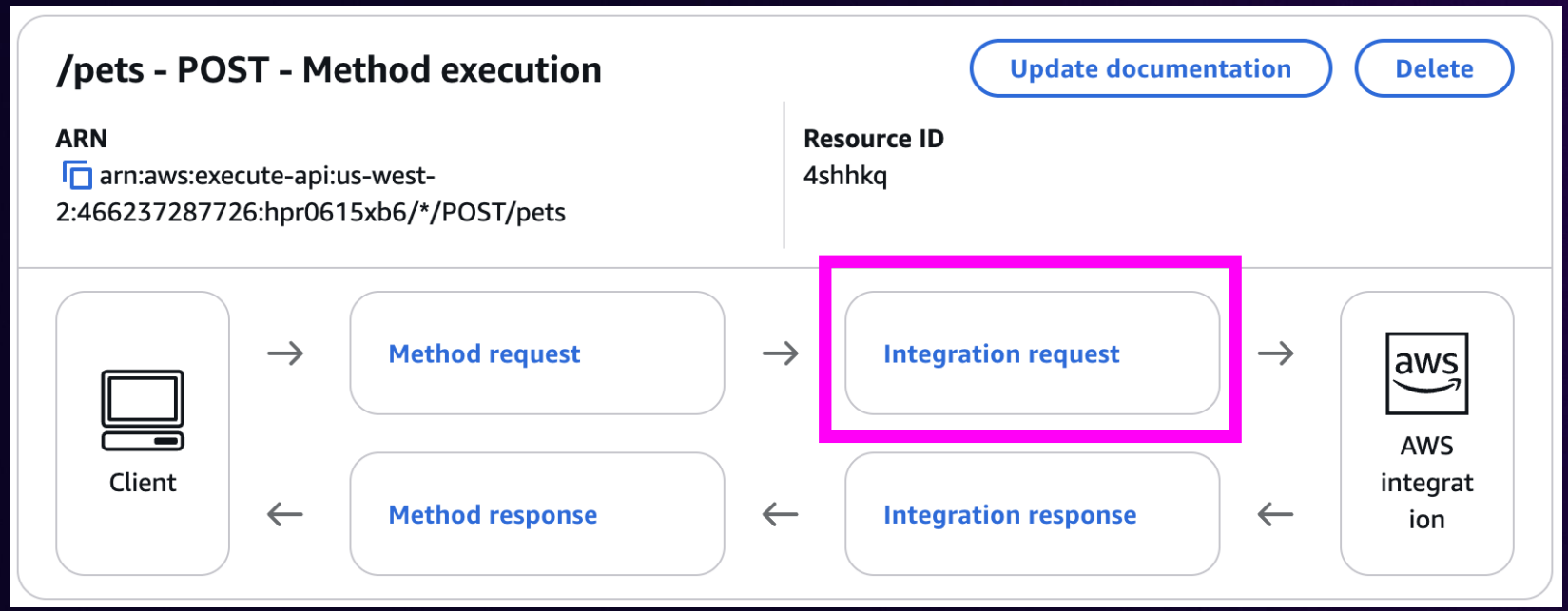


API Gateway method eequest

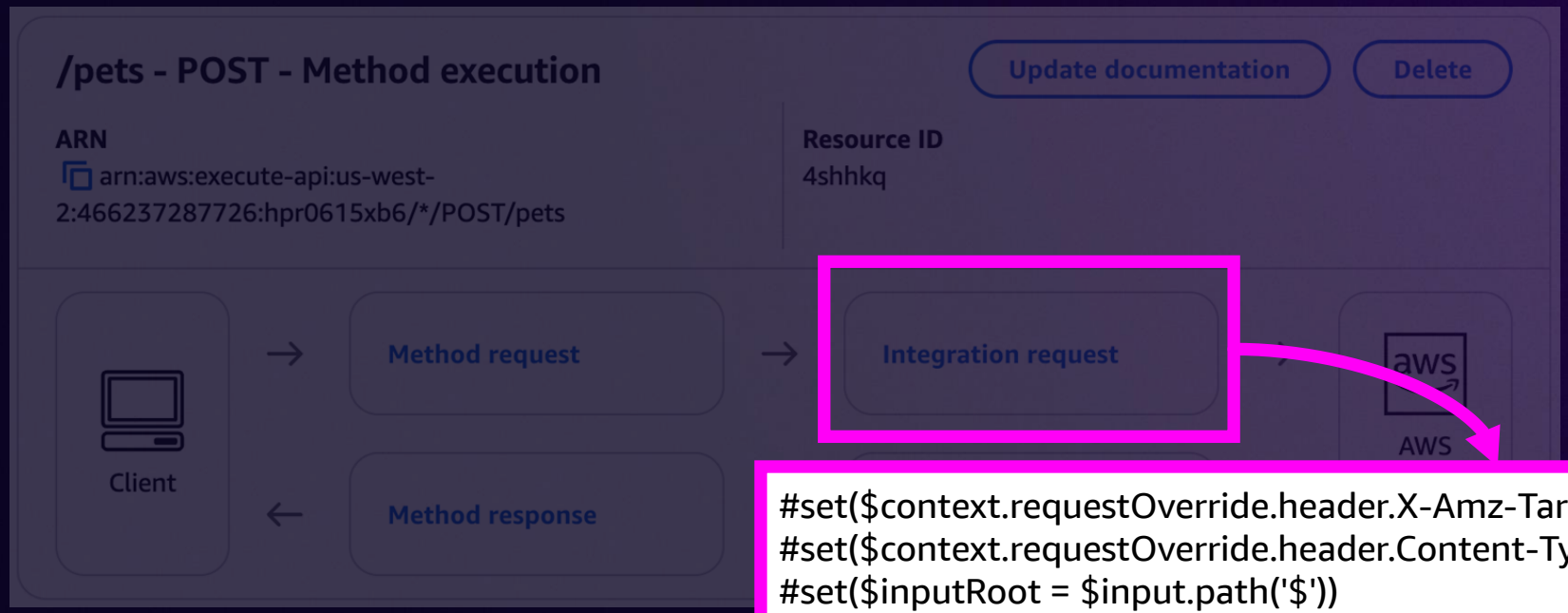




API Gateway integration request



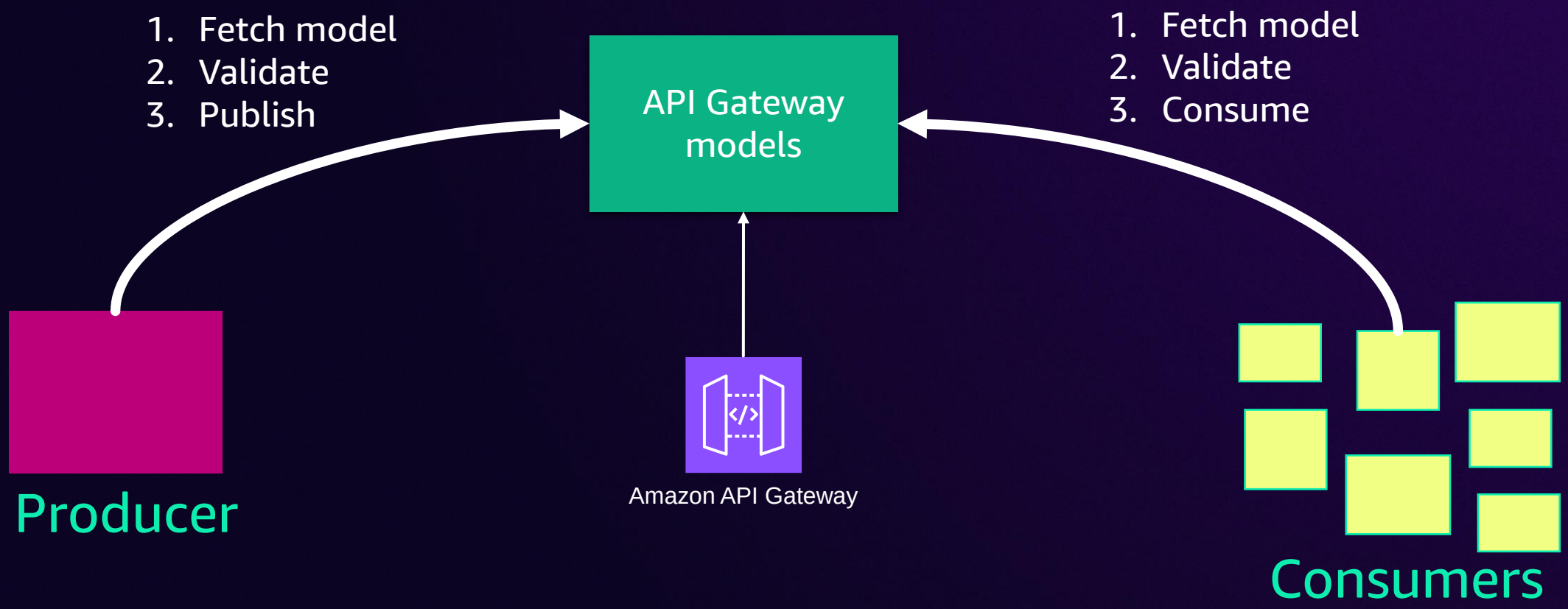
API Gateway integration request



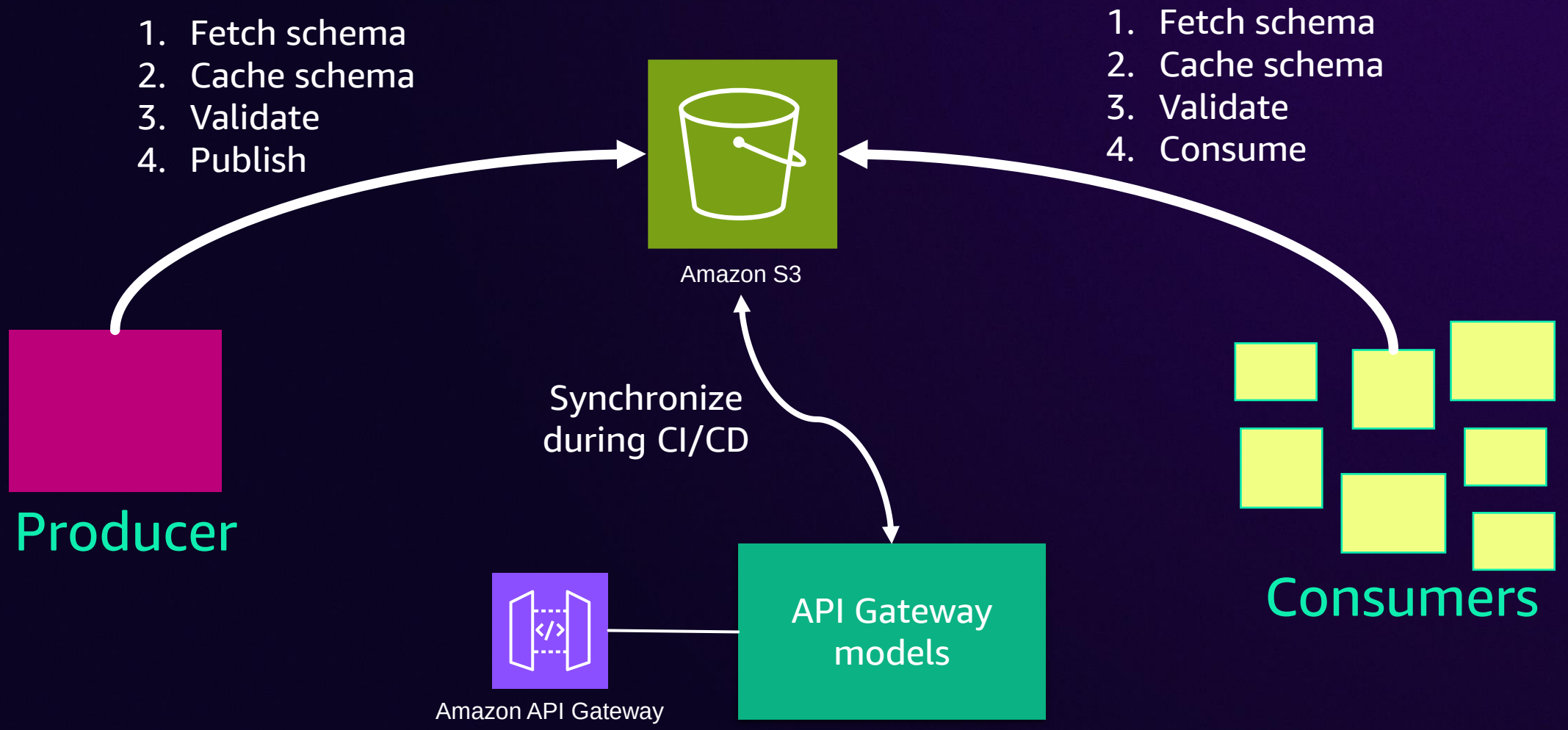
Mapping template

```
#set($context.requestOverride.header.X-Amz-Target = "AWSEvents.PutEvents")
#set($context.requestOverride.header.Content-Type = "application/x-amz-json-1.1")
#set($inputRoot = $input.path('$'))
#set($detailKeys = $inputRoot.detail.keySet())
 {
  "Entries": [{
    "Detail": "$util.escapeJavaScript($input.json("$.detail")).replaceAll("\'","'")",
    "DetailType": "$inputRoot.detail-type",
    "EventBusName": "apigw-eb-proxy-stack-central-bus",
    "Source": "$inputRoot.source"
 }]
}
```

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": "F430C4DC",
    "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": { … }
}
```

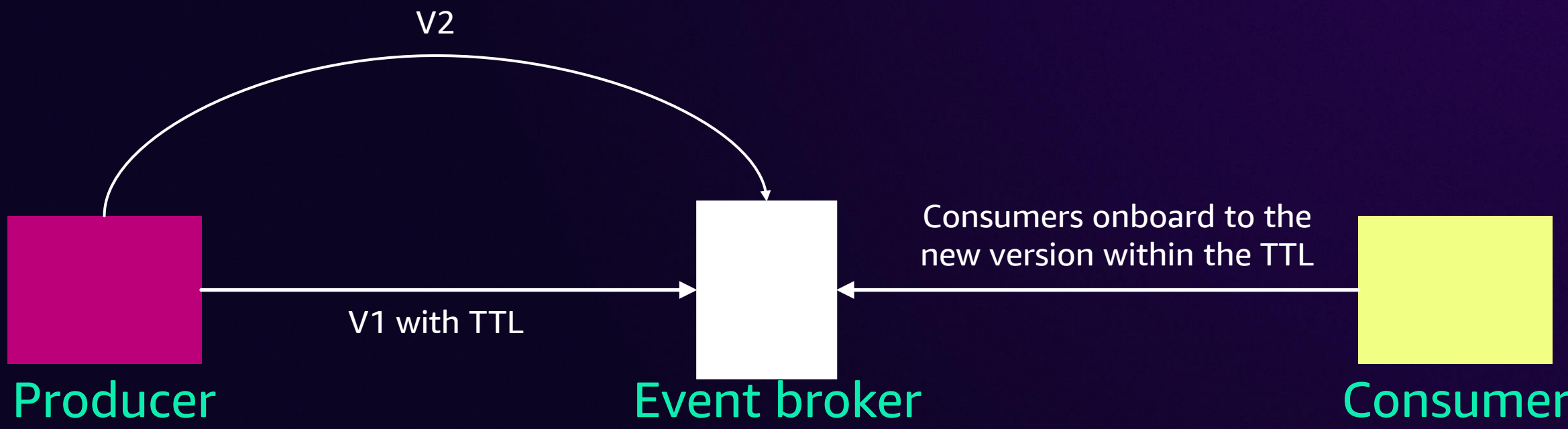
Expiration of version 1

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

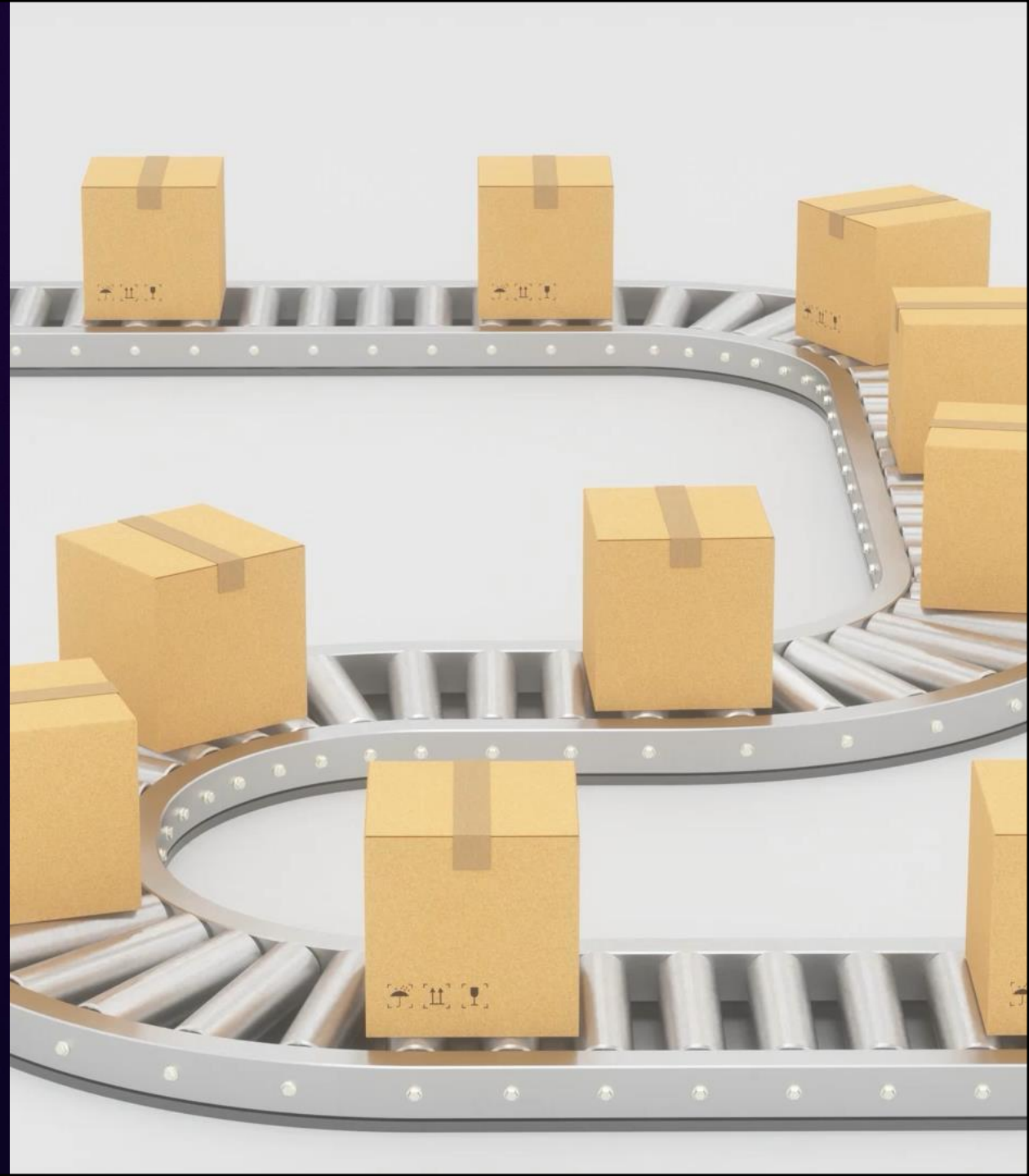
Double publishing events



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

Search portfolios

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

Search products

1

Product name	Product ID	Product type	Created time	Description	Constraints
Reporting Rule	prod-k6z4ainehbzac	CLOUD_FORMATION_TEMPLATE	Thu, Oct 3, 2024, 8:37:46 PM PDT	AWS EventBrisge rule to capture reporting events	0
Transcription Rule	prod-kpga3kq4lebgi	CLOUD_FORMATION_TEMPLATE	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

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

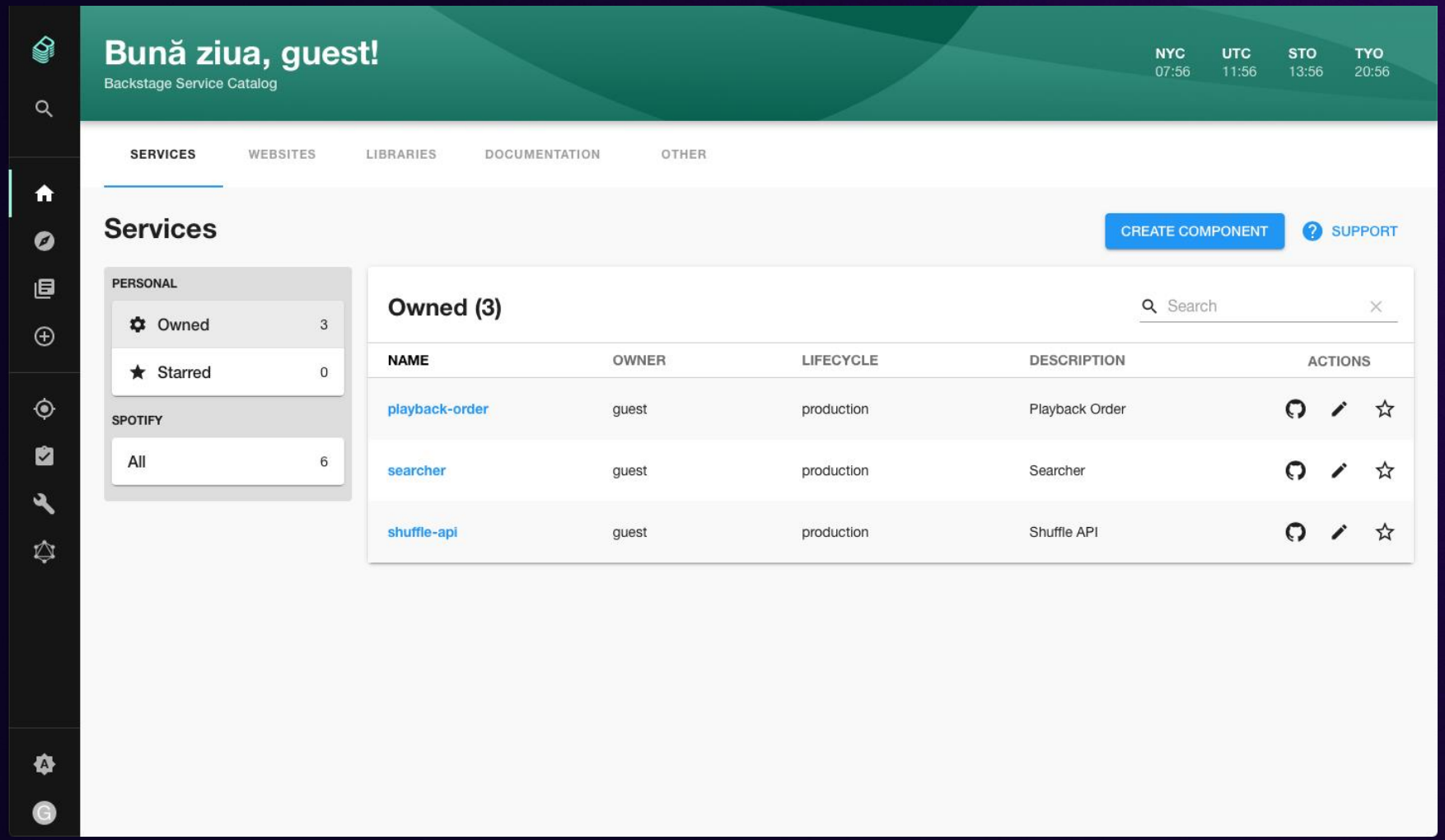
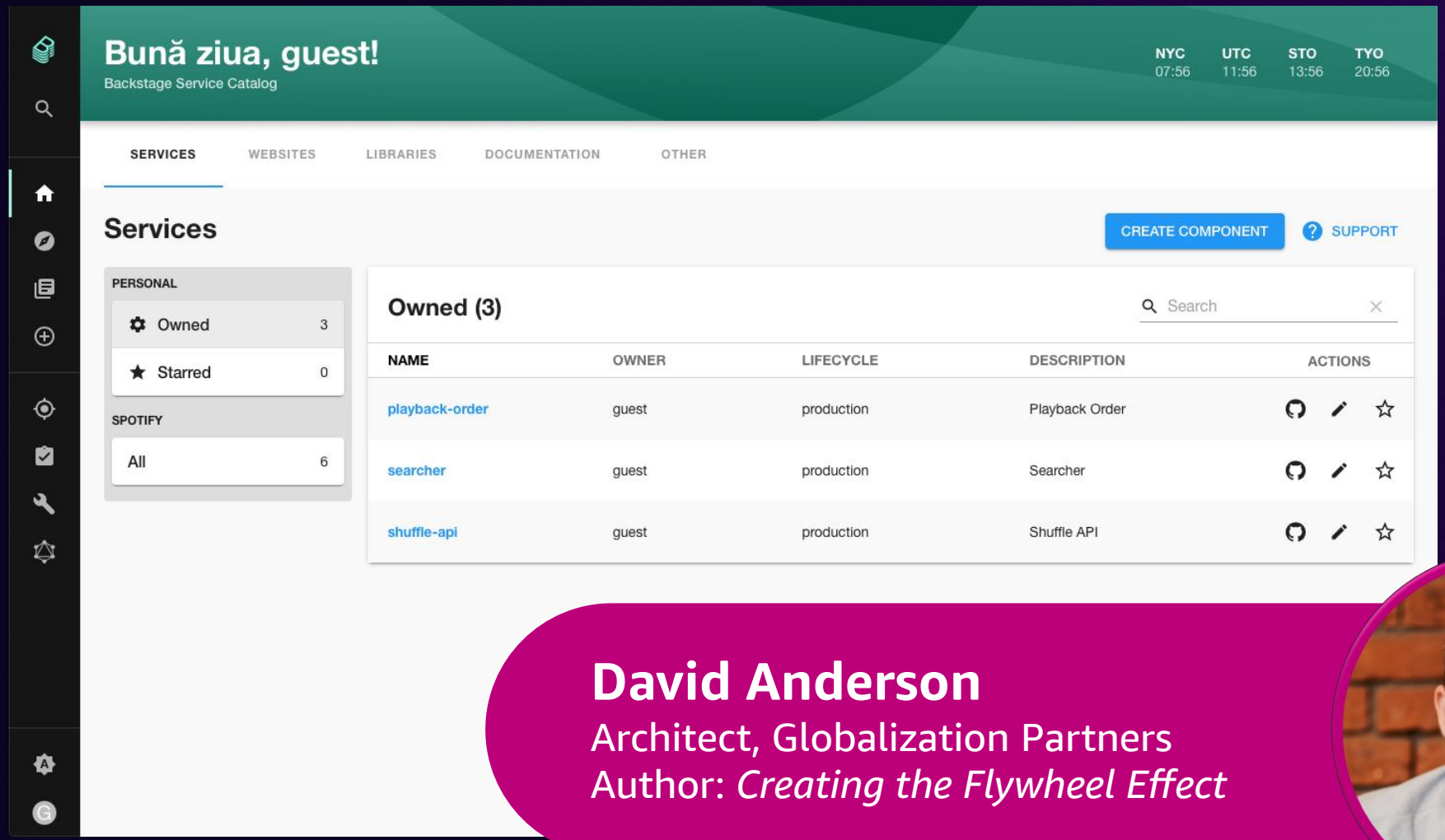


Image credit: https://backstage.io/docs/features/software-catalog/

Backstage

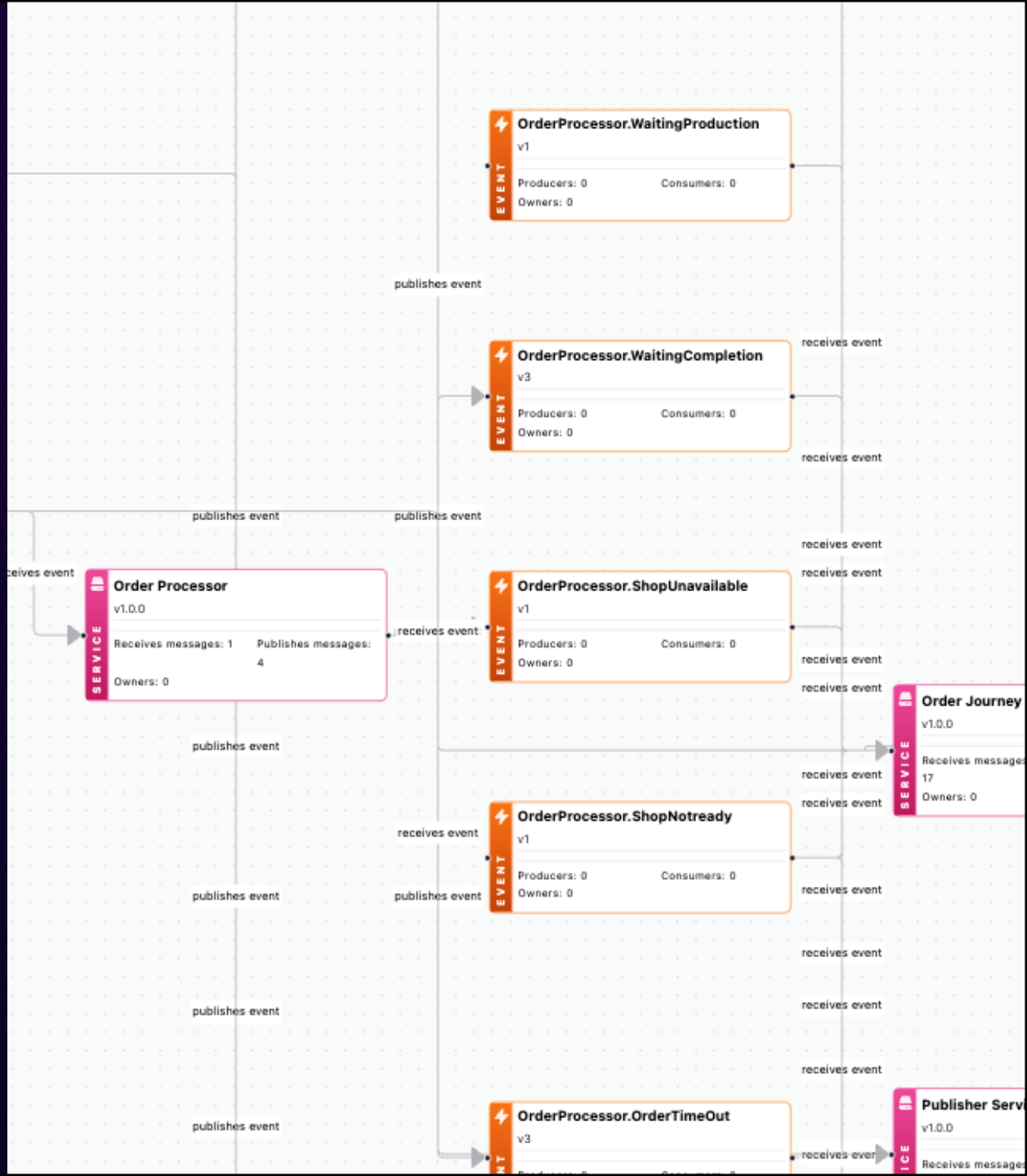


David Anderson
Architect, Globalization Partners
Author: *Creating the Flywheel Effect*

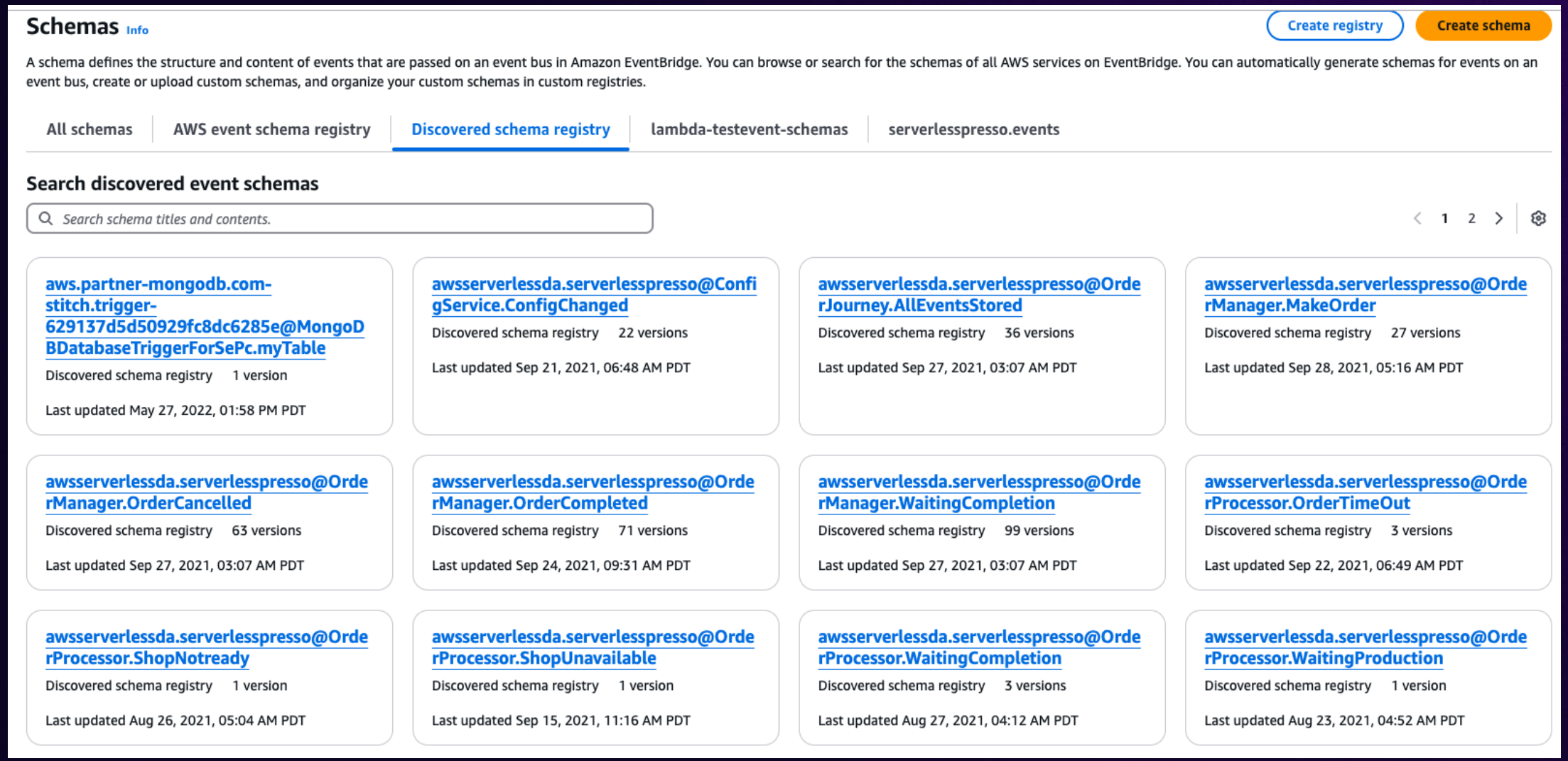
Image credit: https://backstage.io/docs/features/software-catalog/

Event schema discovery

- Documentation
- Versioning
- Mapping



Amazon EventBridge schema registry



Amazon EventBridge schema registry

awsserverlessda.serverlesspresso@OrderManager.OrderCompleted

Delete all versions

Schema details

Schema name
awsserverlessda.serverlesspresso@OrderManager.OrderCompleted

Last modified
Sep 24, 2021, 09:31 AM PDT

Schema ARN
arn:aws:schemas:us-west-2:468083054740:schema/discovered-schemas/awsserverlessda.serverlesspresso@OrderManager.OrderCompleted

Description
-

Schema registry
discovered-schemas

Number of versions
71

Schema type
OpenAPI 3.0

Version 71

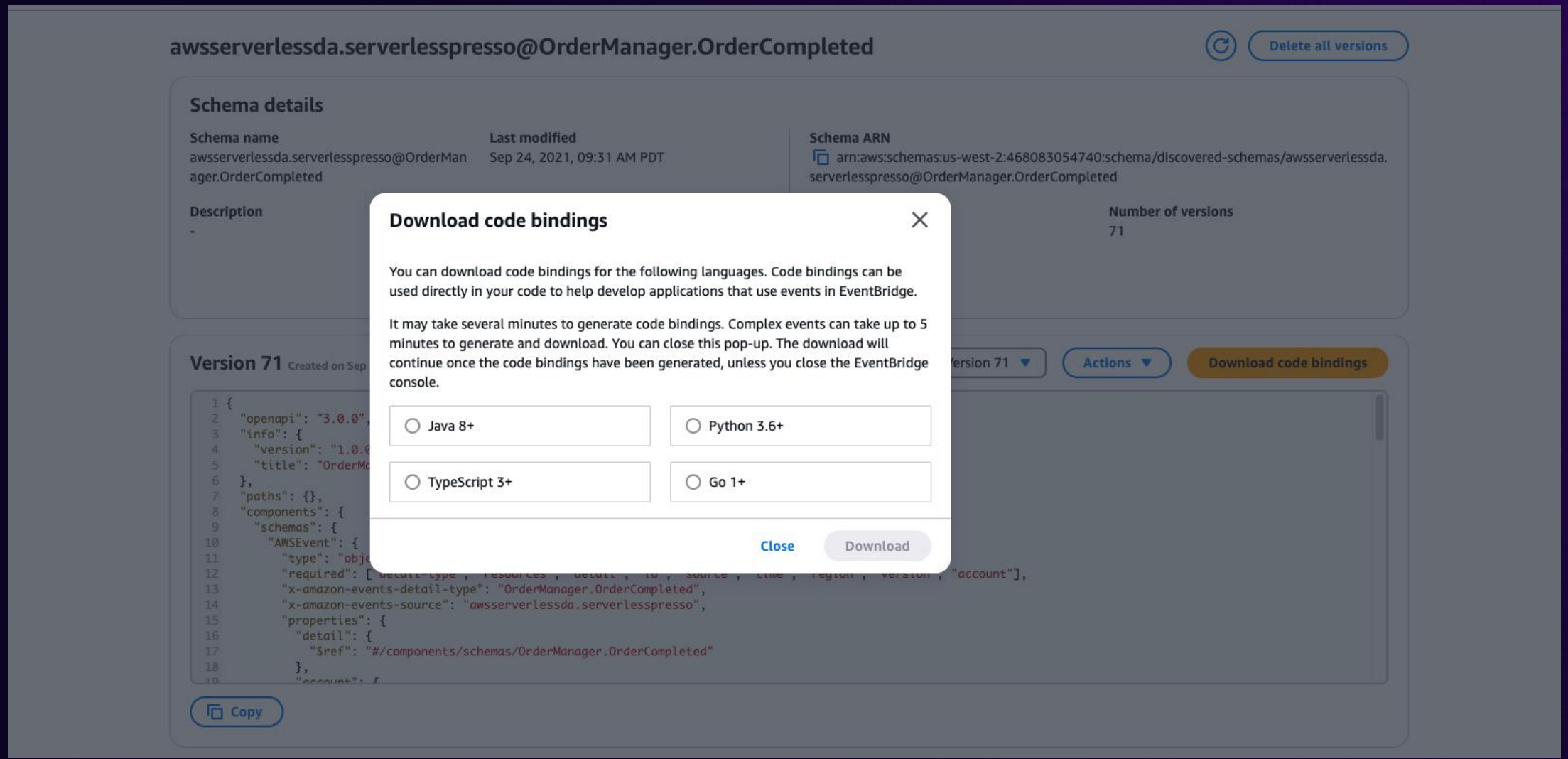
Created on Sep 24, 2021, 09:31 AM PDT

Version 71 | Actions | Download code bindings

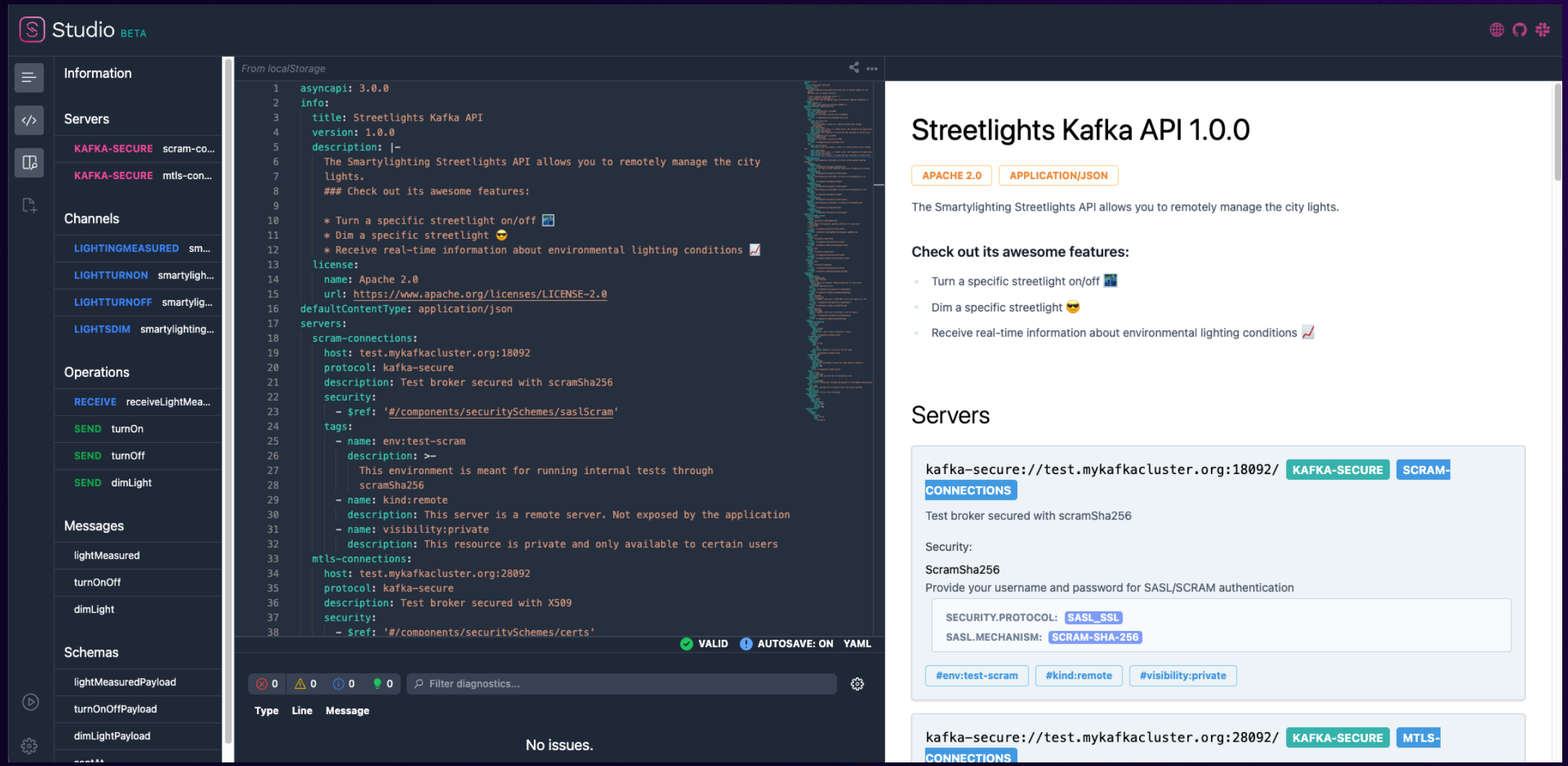
```
{
  "openapi": "3.0.0",
  "info": {
    "version": "1.0.0",
    "title": "OrderManager.OrderCompleted"
  },
  "paths": {},
  "components": {
    "schemas": {
      "AWSEvent": {
        "type": "object",
        "required": ["detail-type", "resources", "detail", "id", "source", "time", "region", "version", "account"],
        "x-amazon-events-detail-type": "OrderManager.OrderCompleted",
        "x-amazon-events-source": "awsserverlessda.serverlesspresso",
        "properties": {
          "detail": {
            "$ref": "#/components/schemas/OrderManager.OrderCompleted"
          },
```

Copy

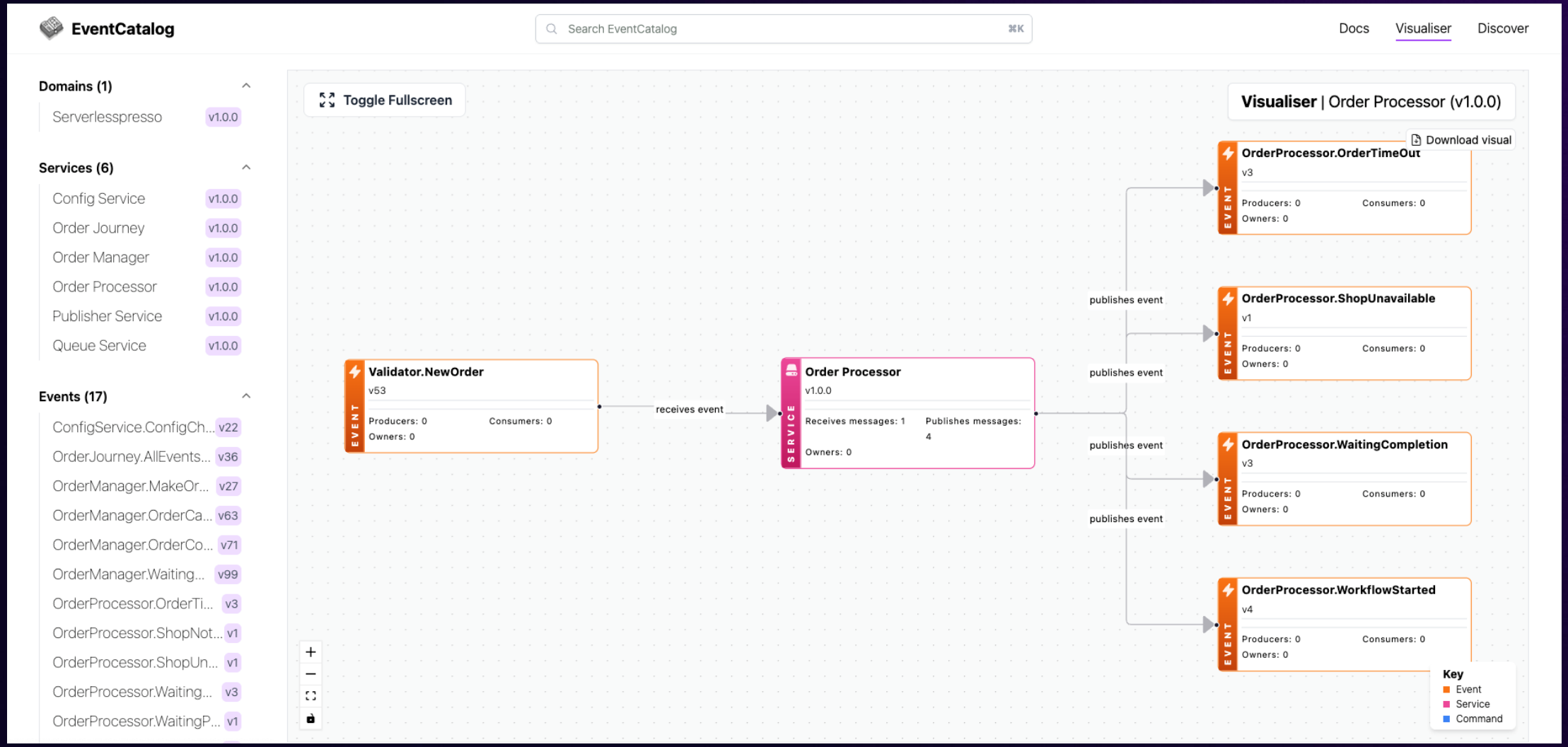
Amazon EventBridge schema registry



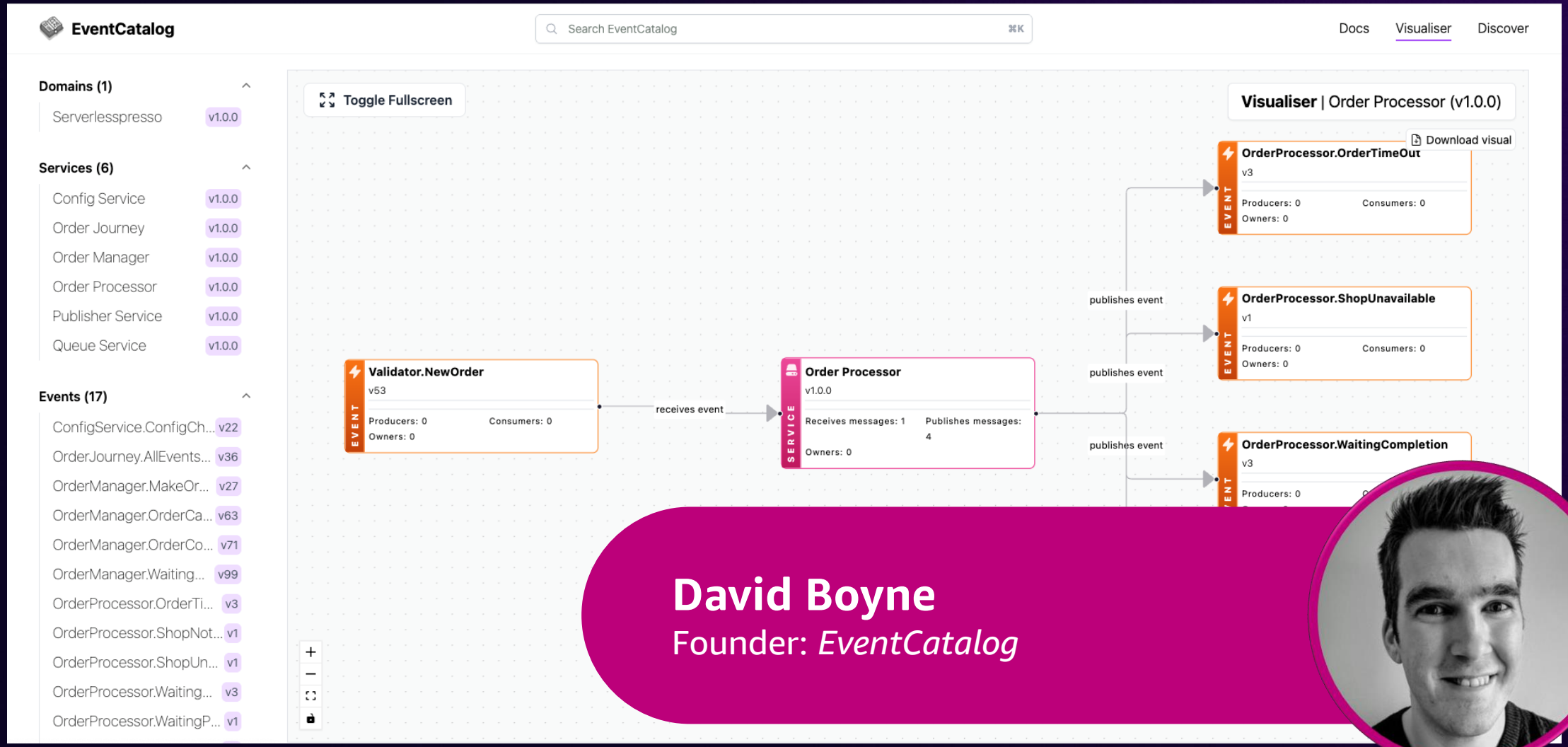
AsyncAPI



EventCatalog



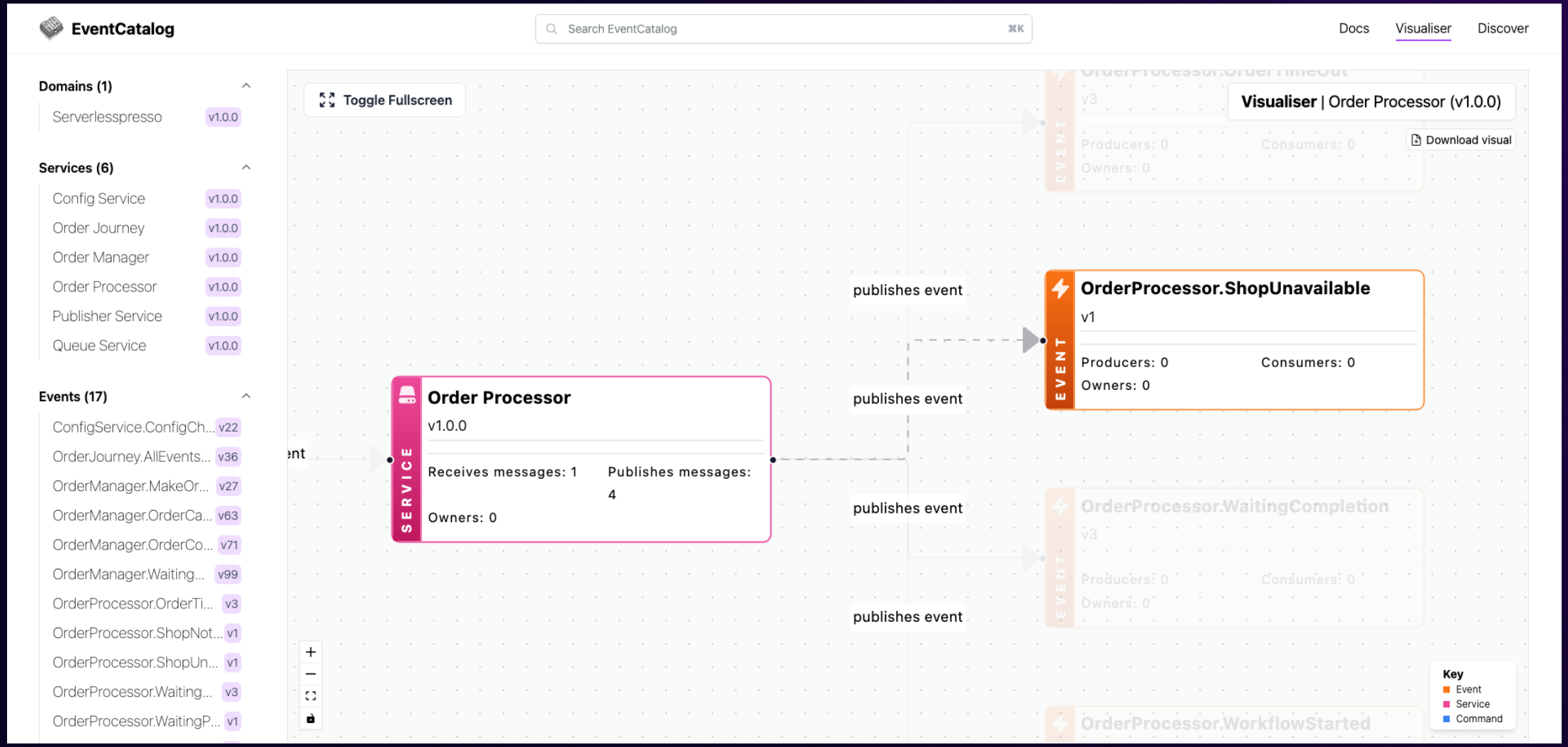
EventCatalog



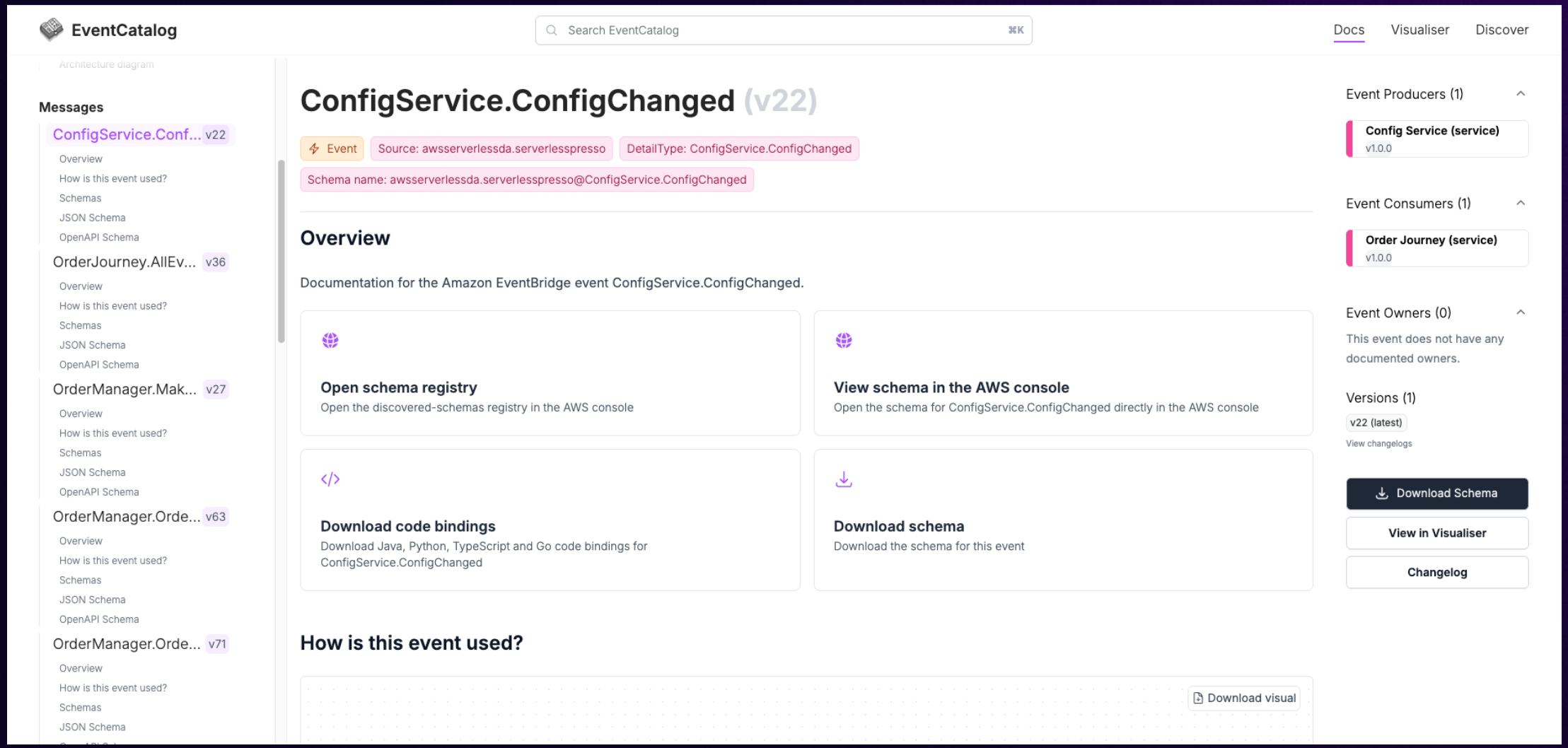
David Boyne

Founder: *EventCatalog*

EventCatalog



EventCatalog



Combining

Amazon EventBridge schema registry

EventCatalog

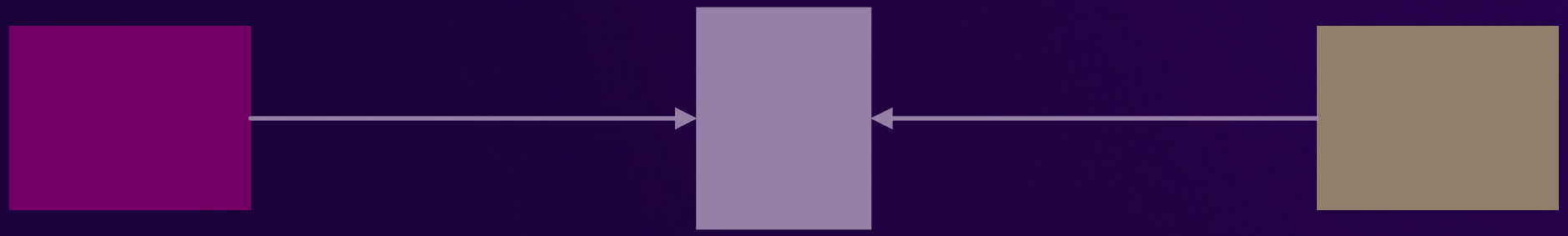
AsyncAPI

- 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

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

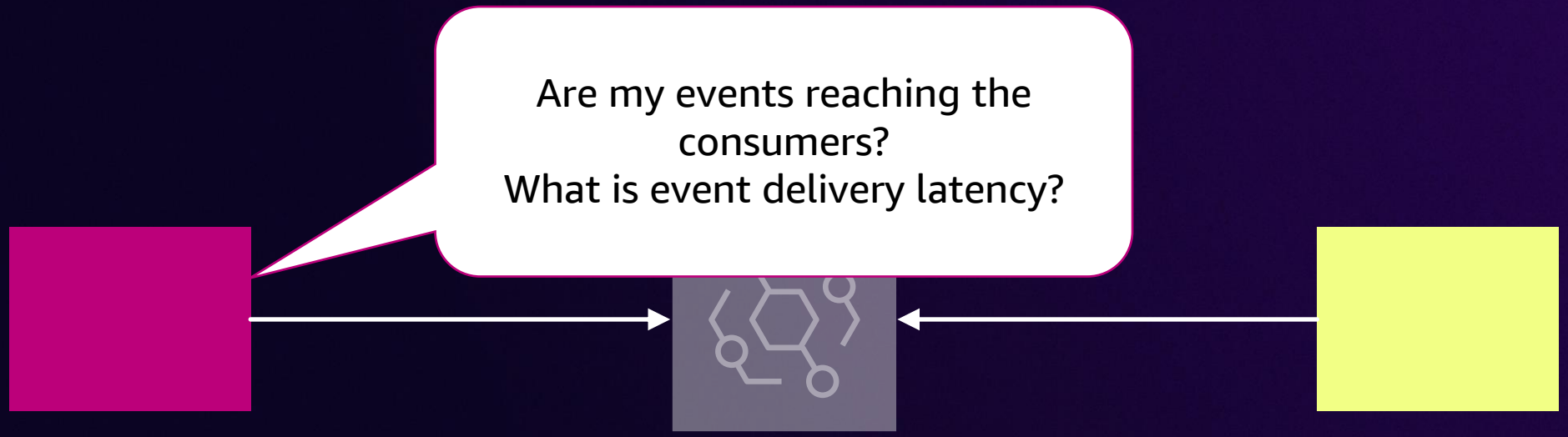
AWS managed

Health of the producer

Health of EventBridge event bus

Health of the consumer

Thinking about observability in EDAs



Health of the producer

Health of EventBridge event bus

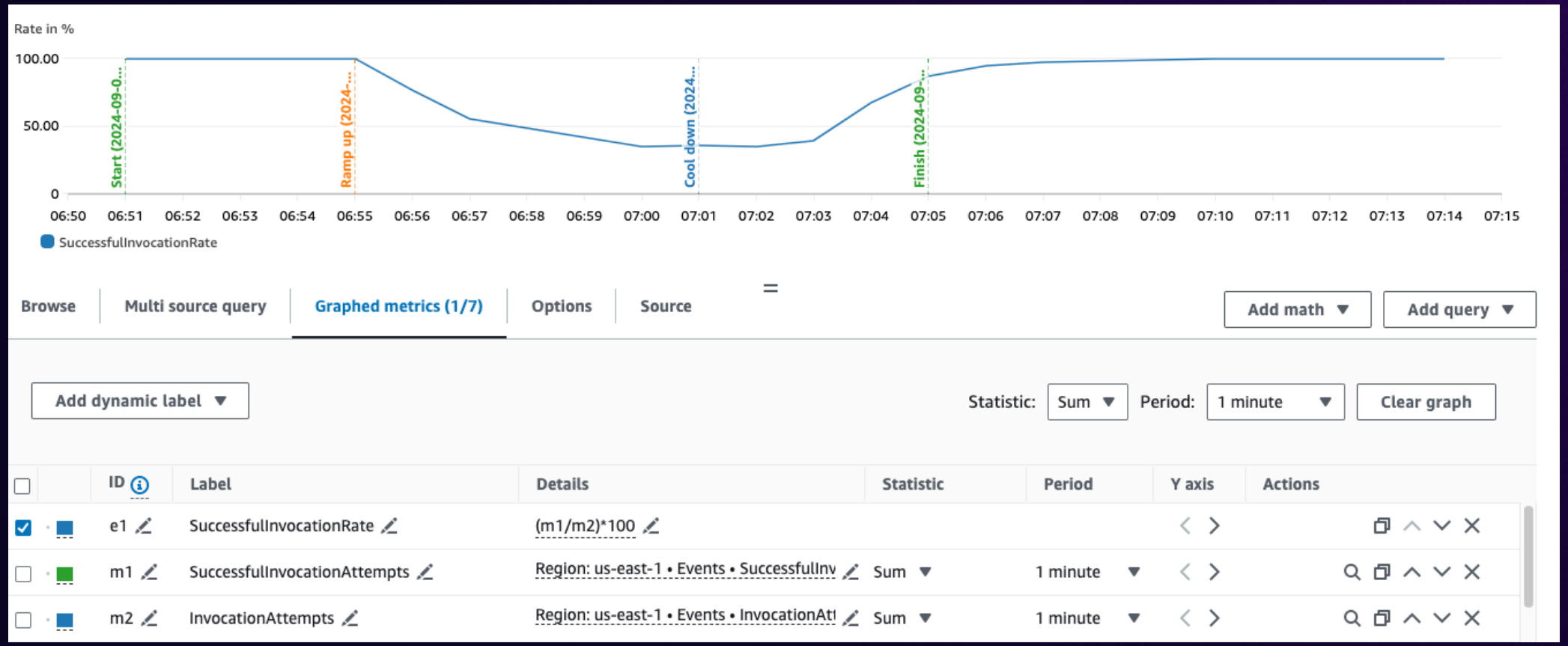
Health of the consumer

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

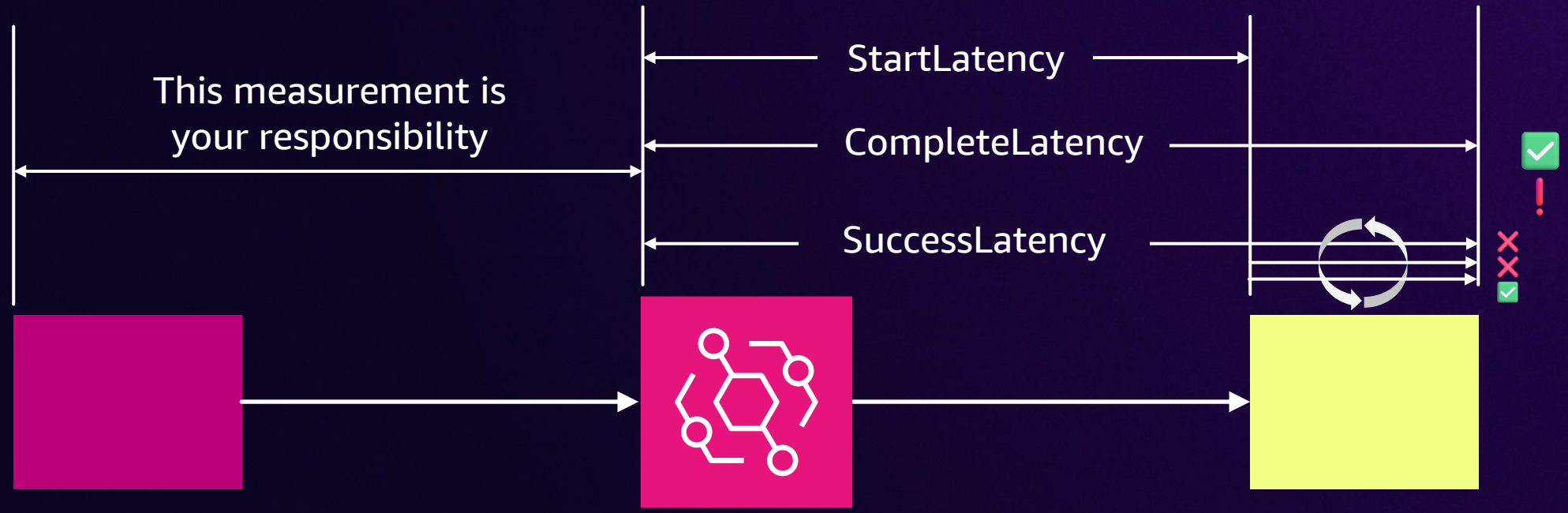
Metric	Dimension	Units
FailedInvocations	RuleName	Count
InvocationsSentToDlq	RuleName	Count
InvocationsFailedToBeSentToDlq	RuleName	Count

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

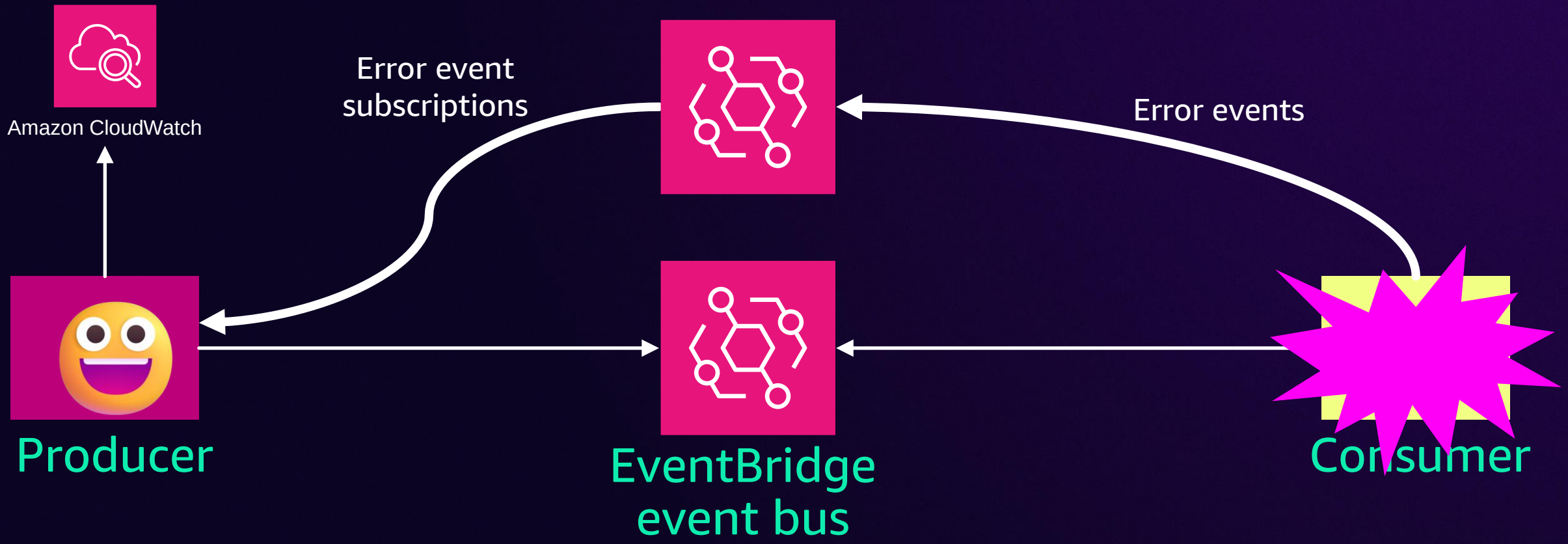
Detecting delivery delays

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

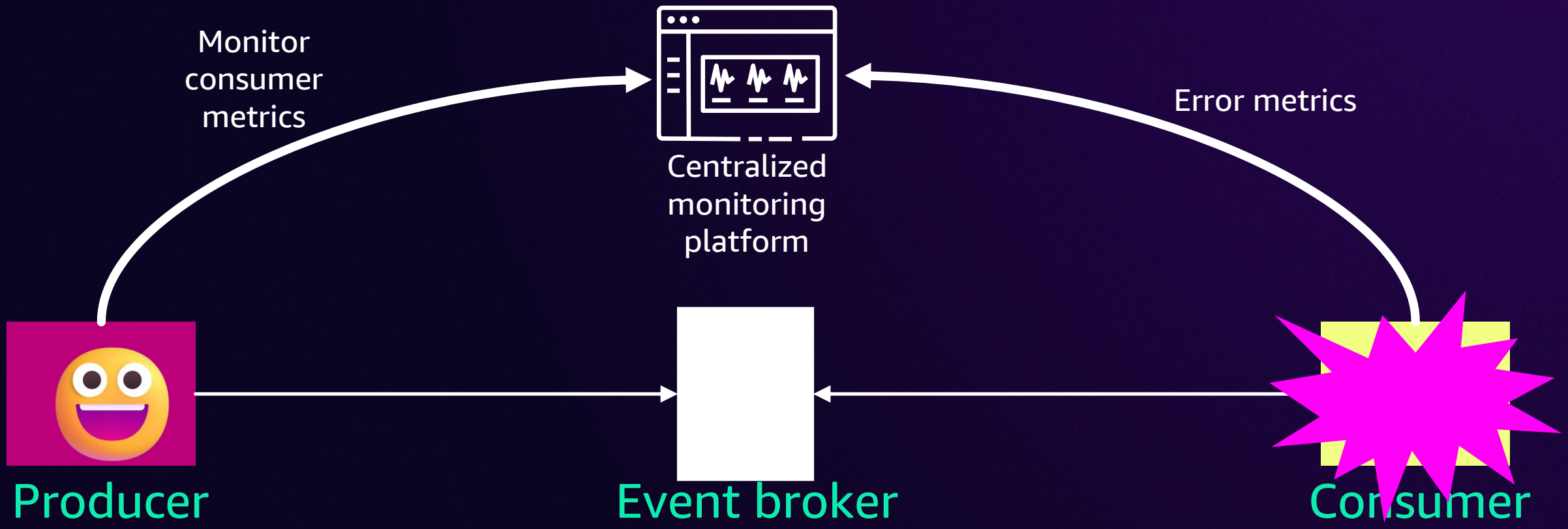
IngestiontoInvocation



Reporting client errors



Reporting client errors



Wrap

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

Earn AWS Serverless badge

Demonstrate your knowledge by achieving **digital badges**

https://s12d.com/serverless-learning

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!

Please complete the session survey in the mobile app

Eric Johnson
@edjgeek

Brian Zambrano
@brianzambrano
zambb@amazon.com