goto;

GOTO
EDA Day 2022

Codenode | London

September 1st, 2022

aws

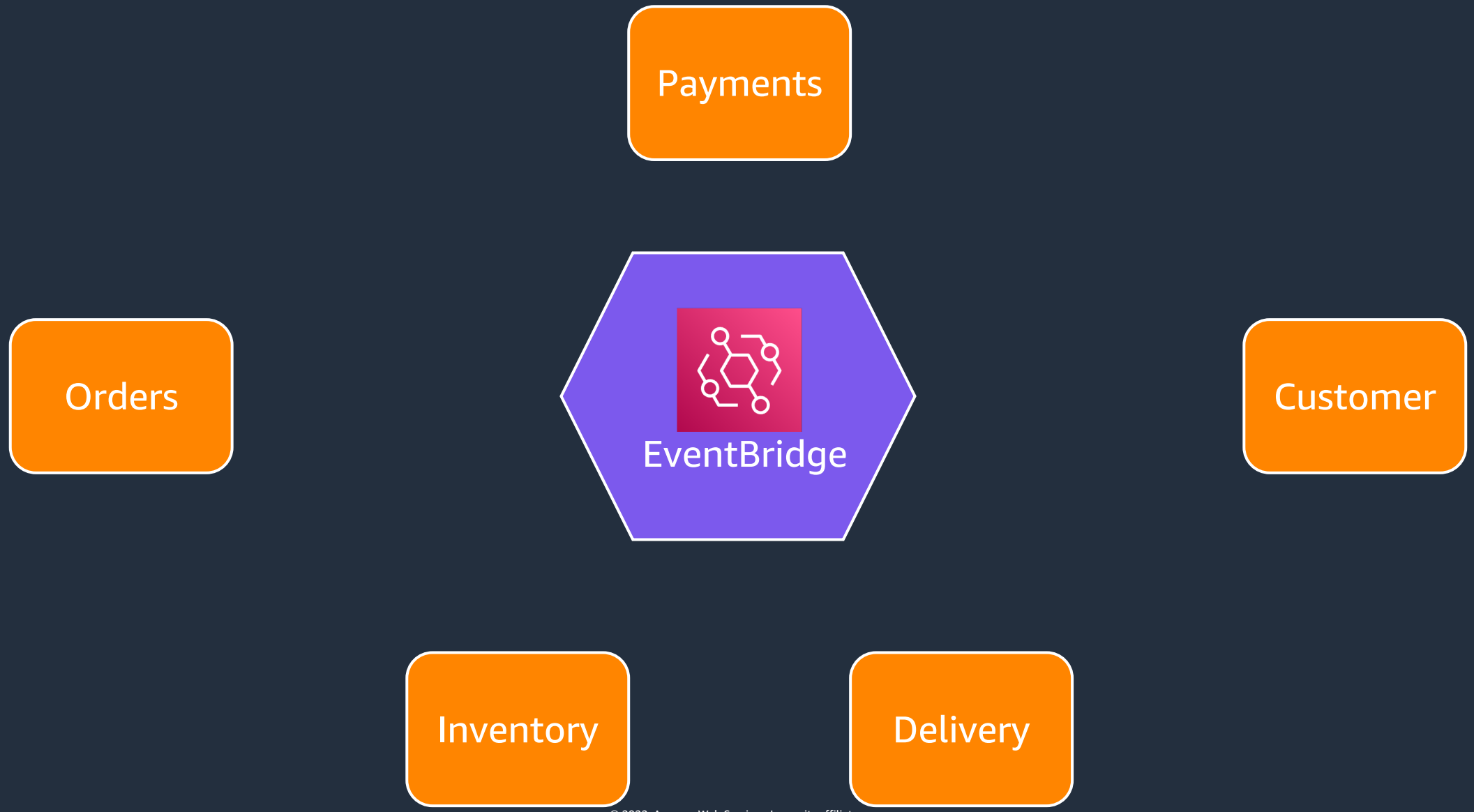
BROUGHT TO YOU BY
AWS AND PARTNERS

Best practices to design and build event-driven applications

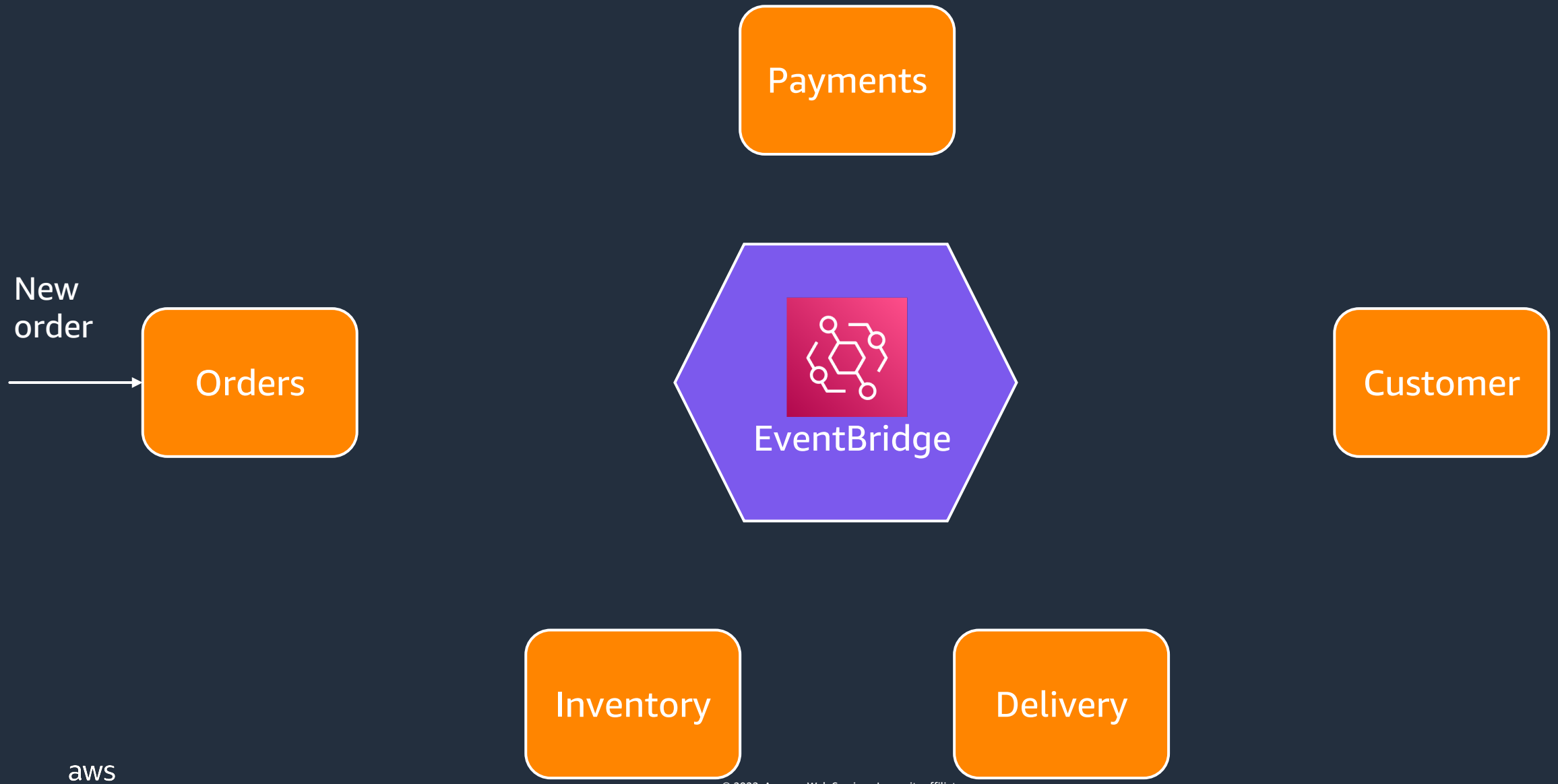
Marcia Villalba (she/her)

Senior Developer Advocate, AWS Serverless
@mavi888uy

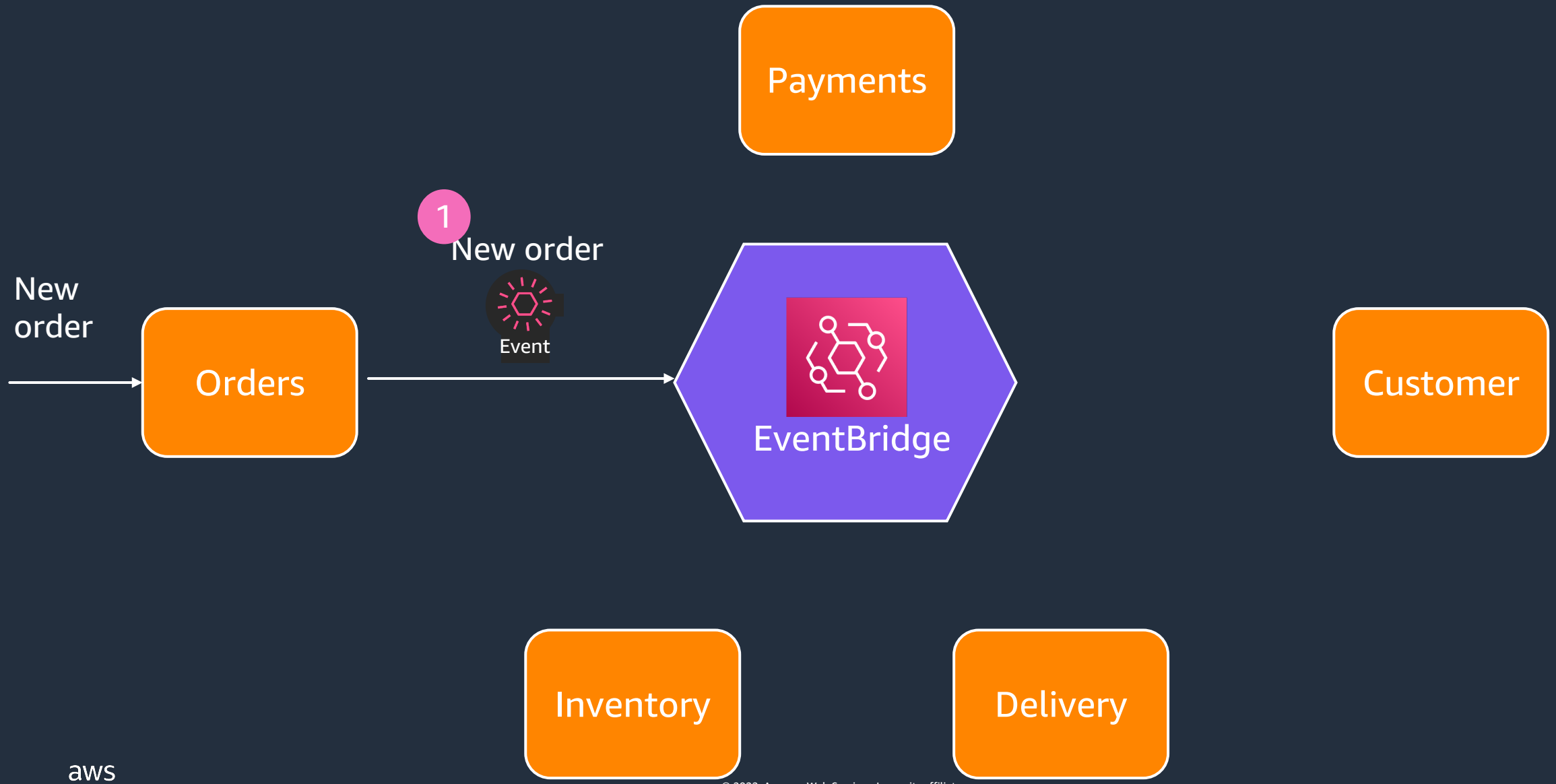
Event Driven Architecture Design



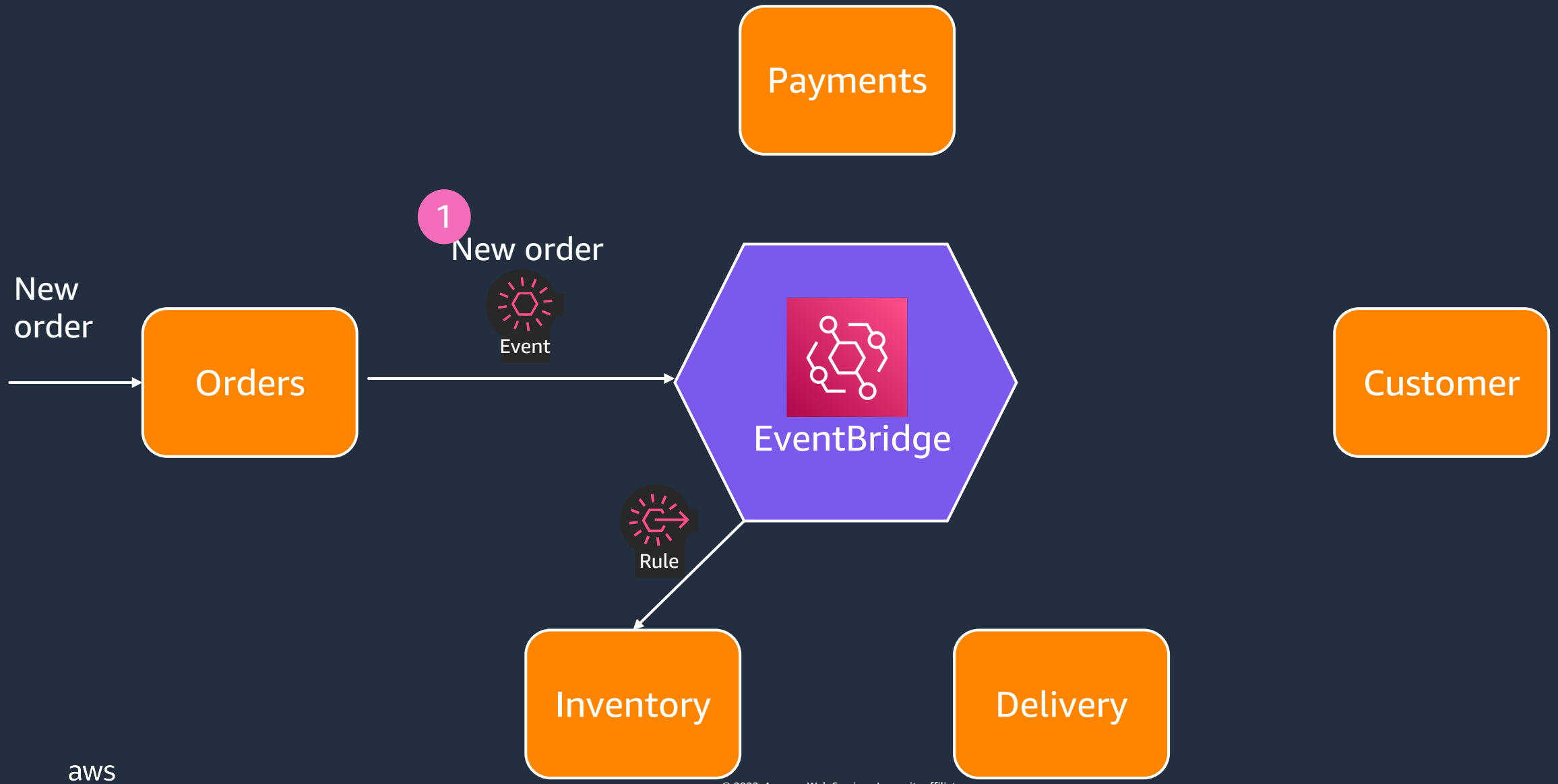
Event Driven Architecture Design



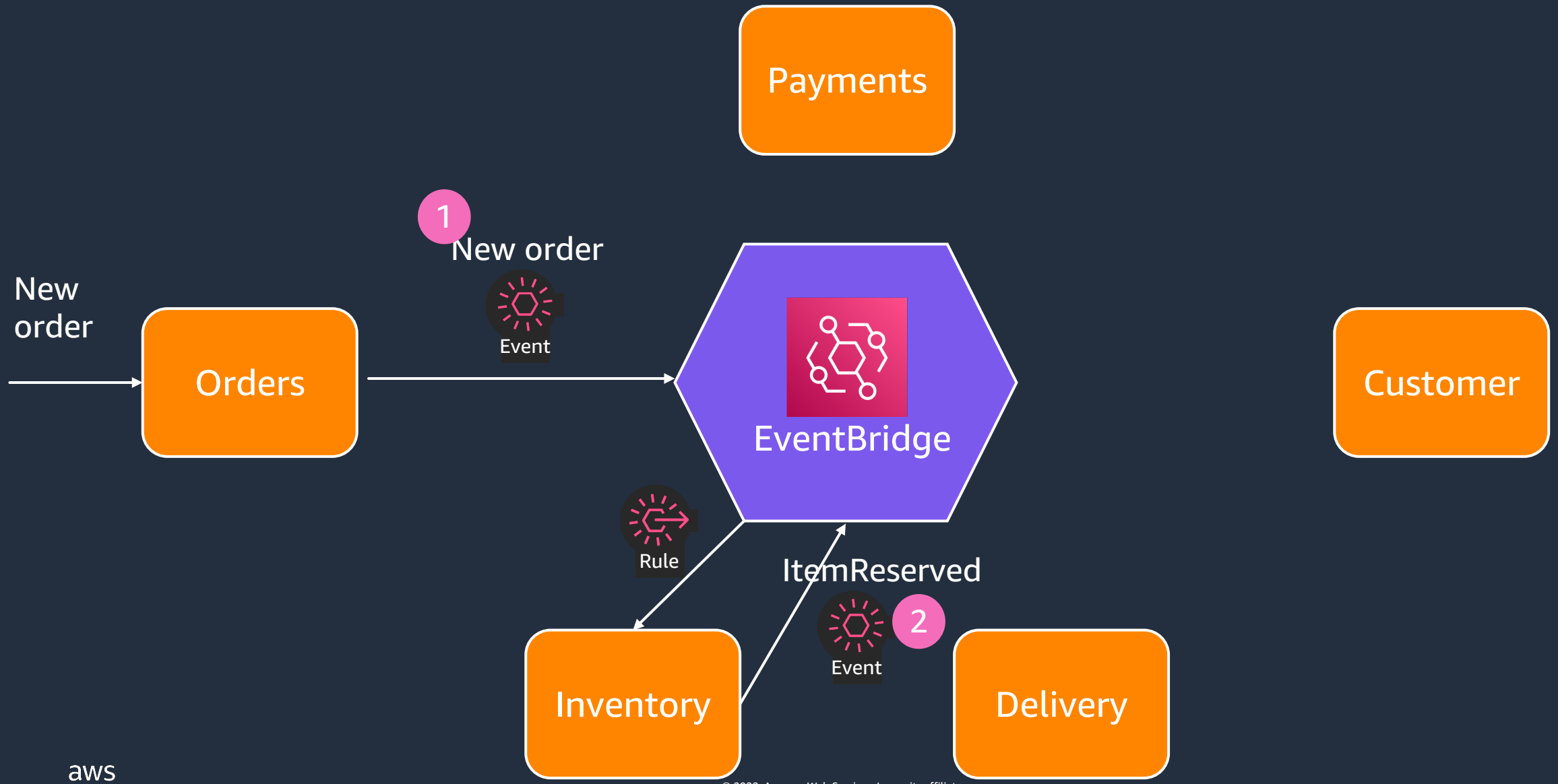
Event Driven Architecture Design



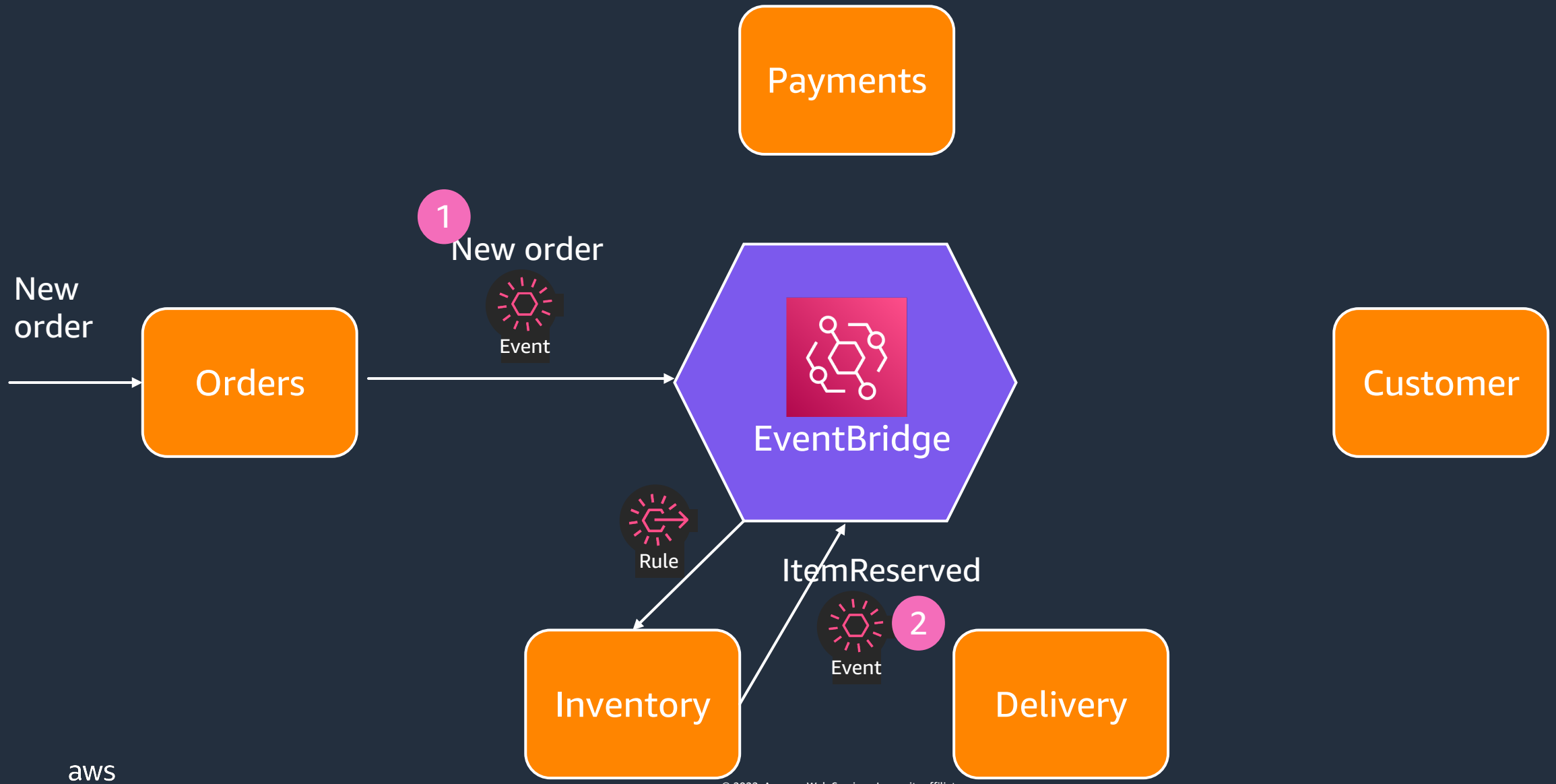
Event Driven Architecture Design



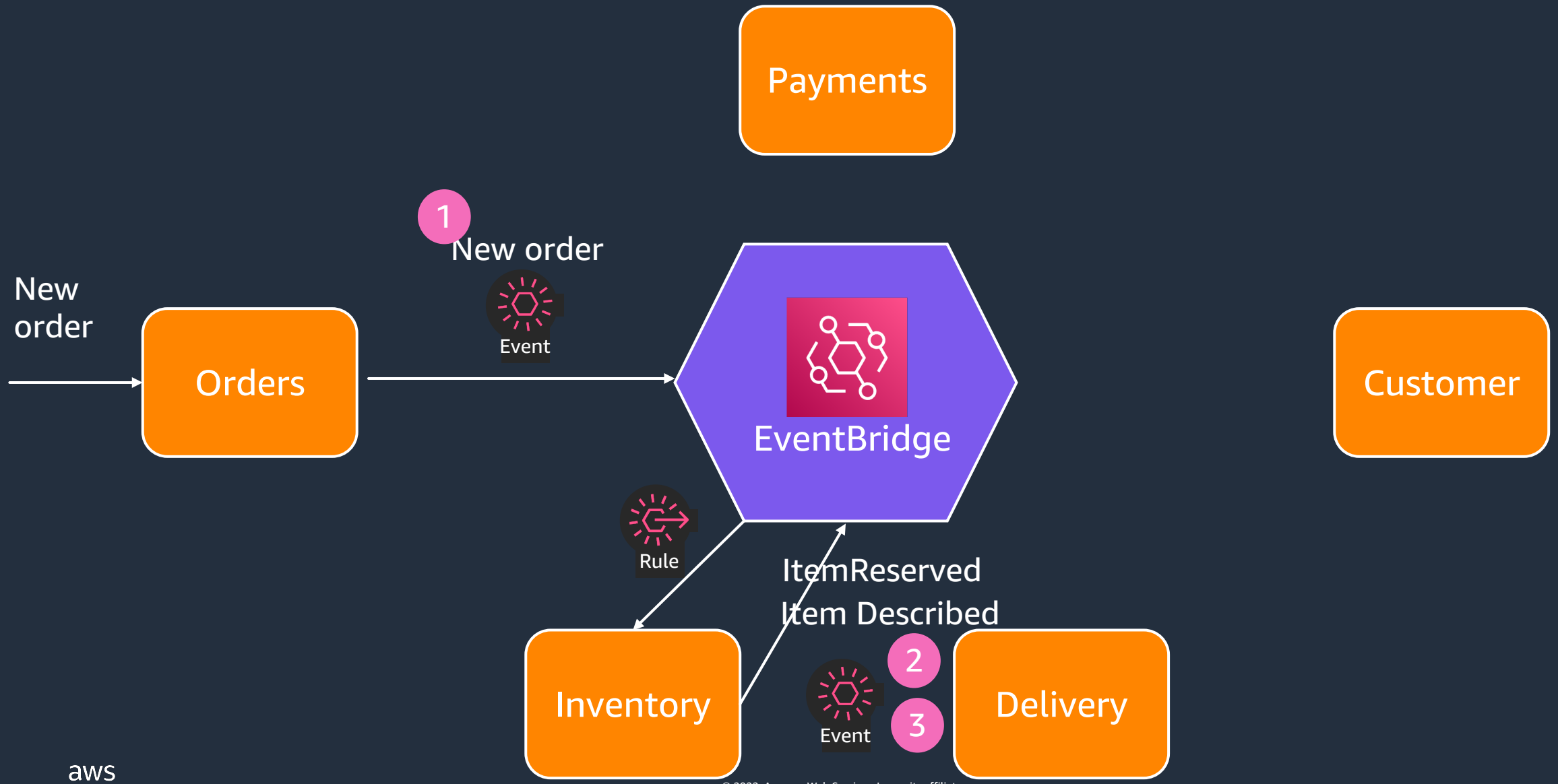
Event Driven Architecture Design



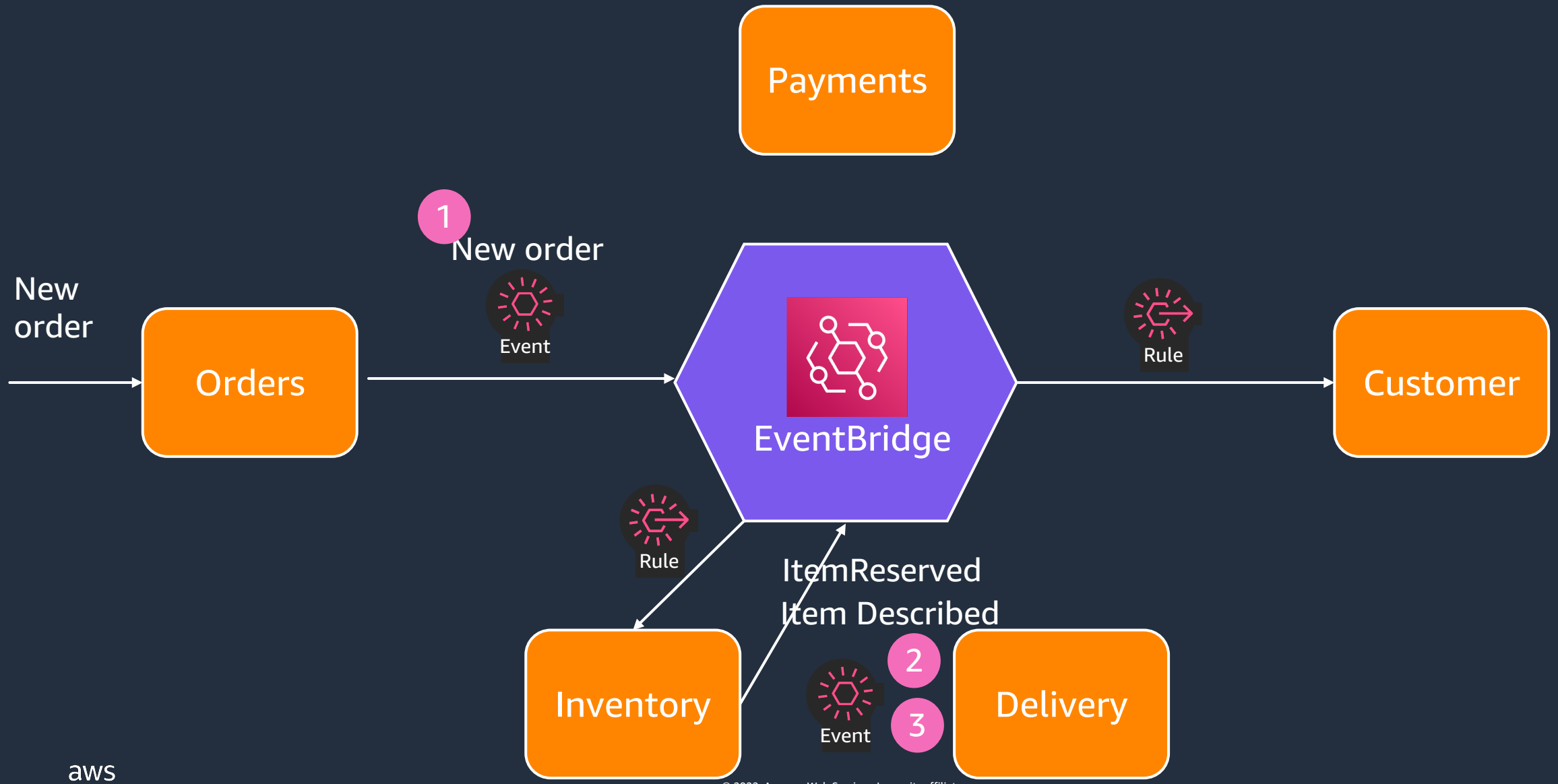
Event Driven Architecture Design



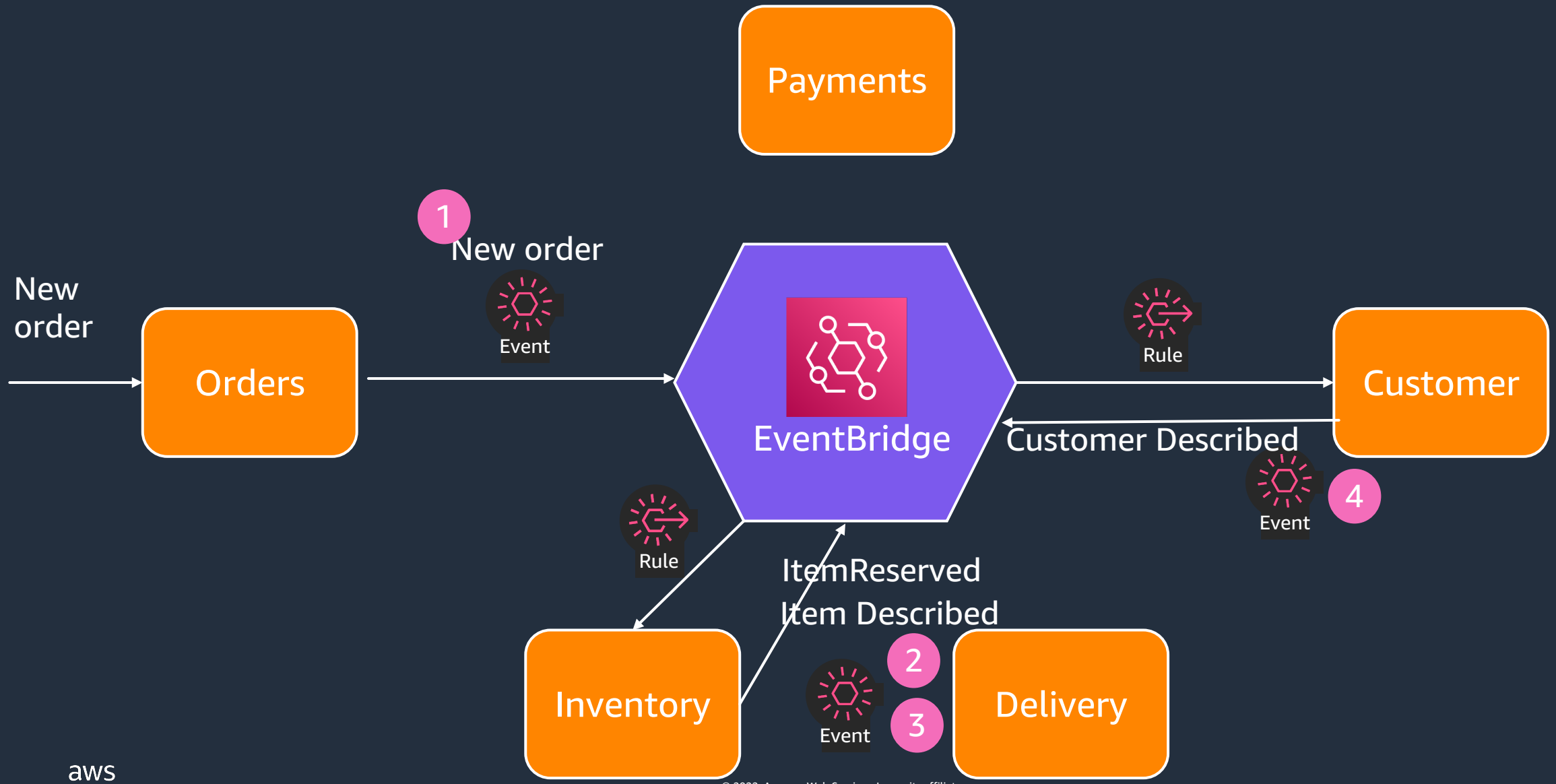
Event Driven Architecture Design



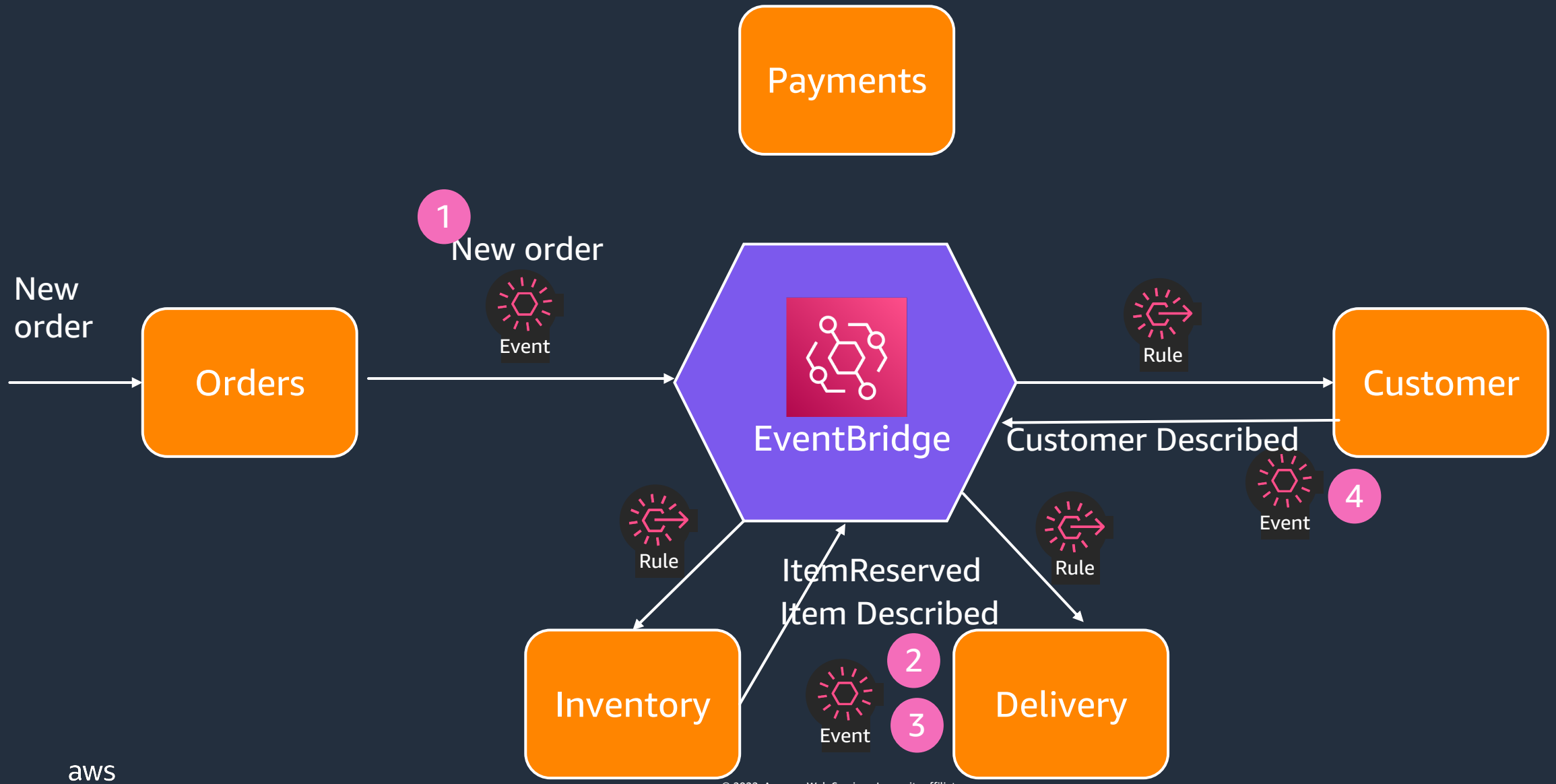
Event Driven Architecture Design



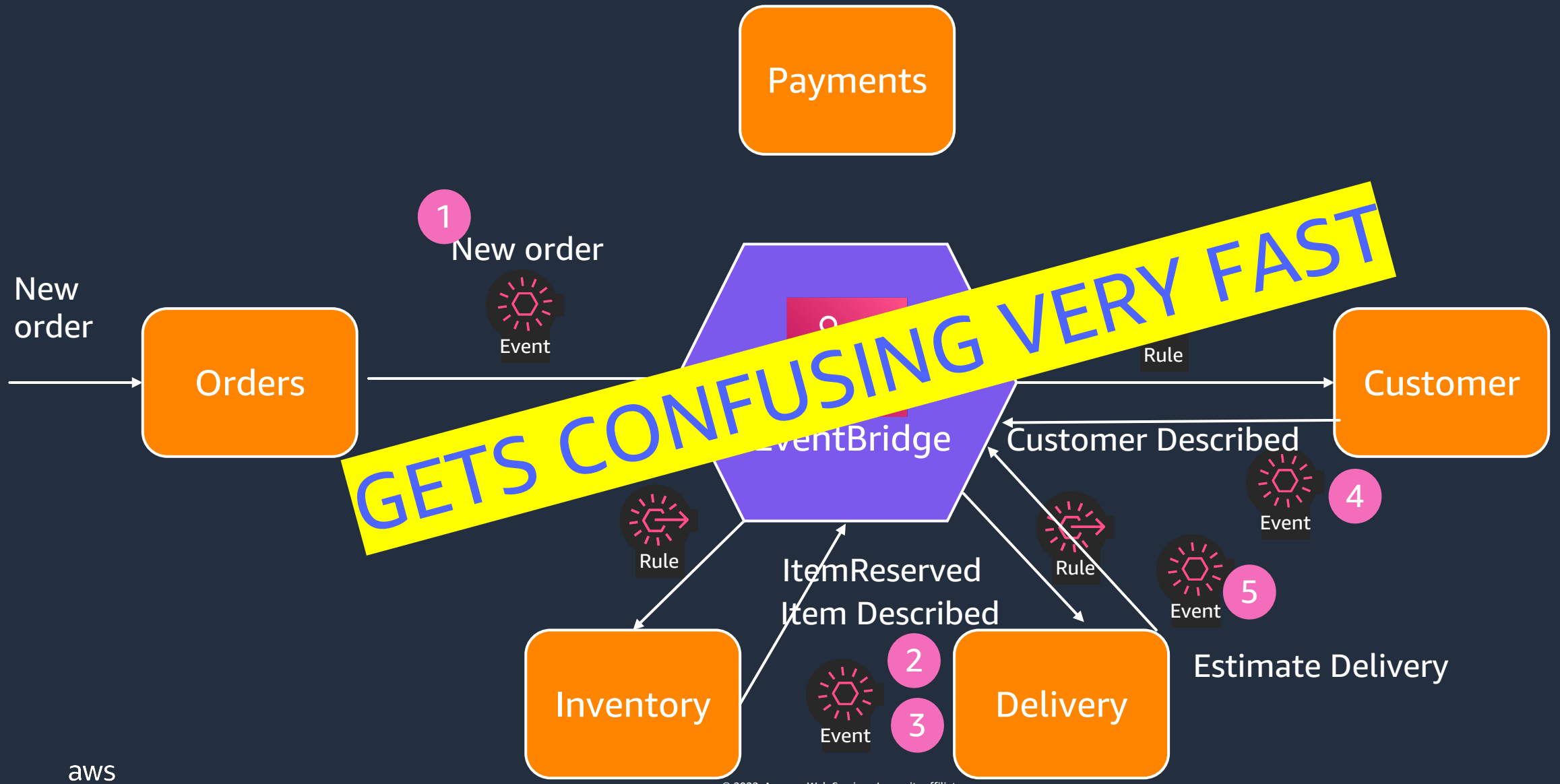
Event Driven Architecture Design



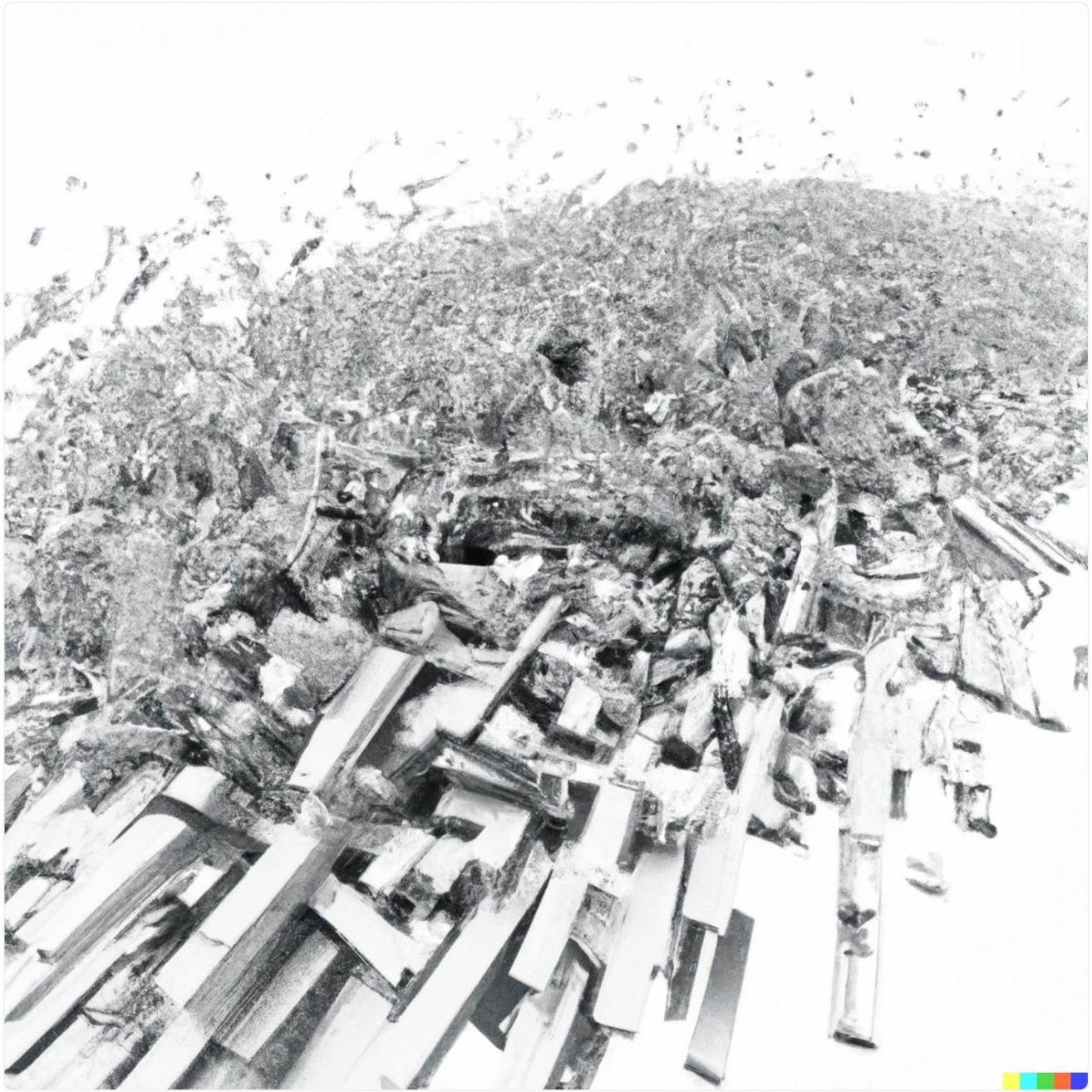
Event Driven Architecture Design



Event Driven Architecture Design



I asked DALL-E
What are
Event-Driven Applications?...



Share

“event driven architecture abstract art ”

Marcia × DALL·E
Human & AI

Agenda

- Brief introduction to Amazon EventBridge
- How to design an event-driven application
- Best practices to use Amazon EventBridge

Introduction to Amazon EventBridge

Introduction to Amazon EventBridge

@mavi888uy
#edaday

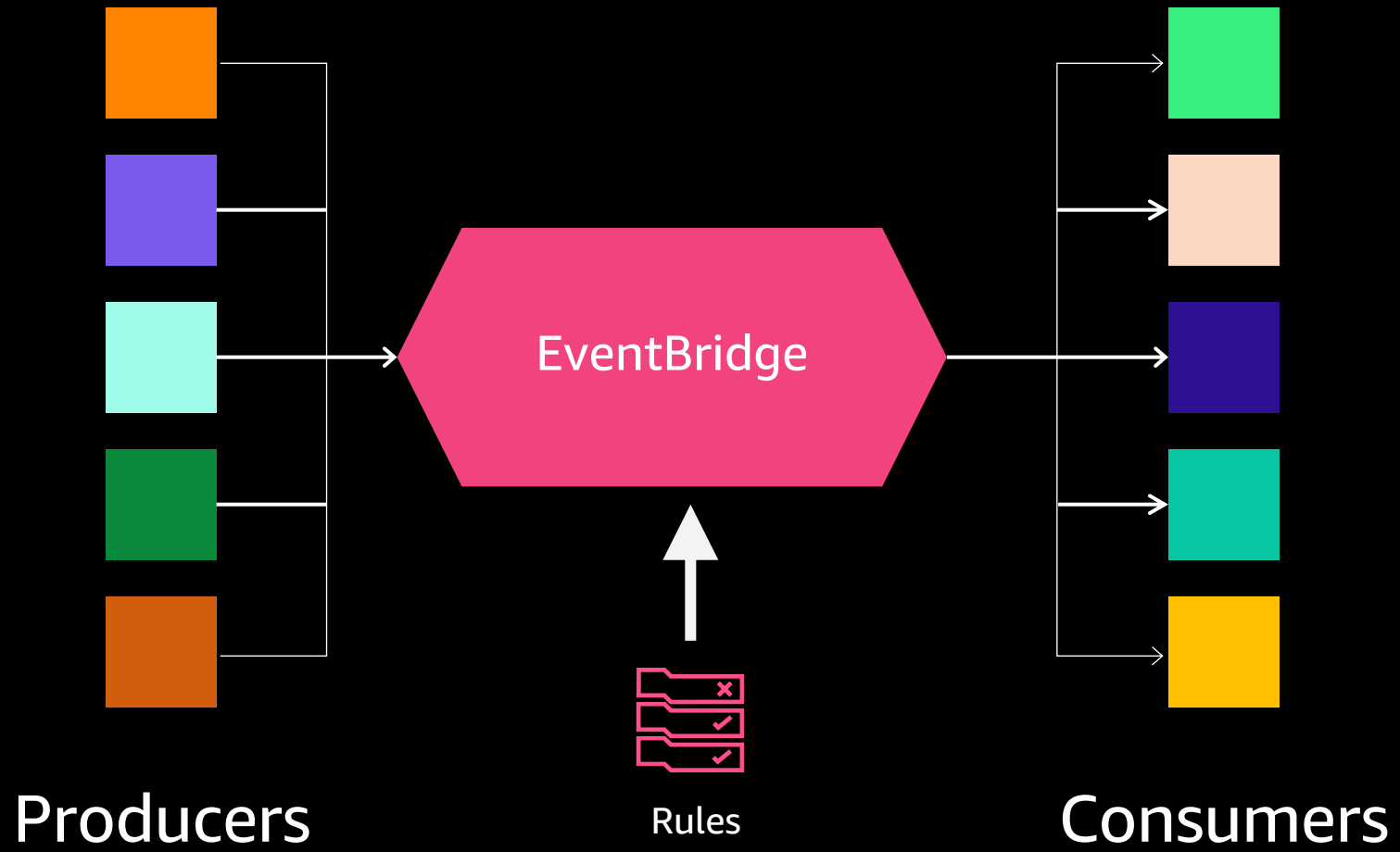
Amazon EventBridge

SERVERLESS EVENT BUS

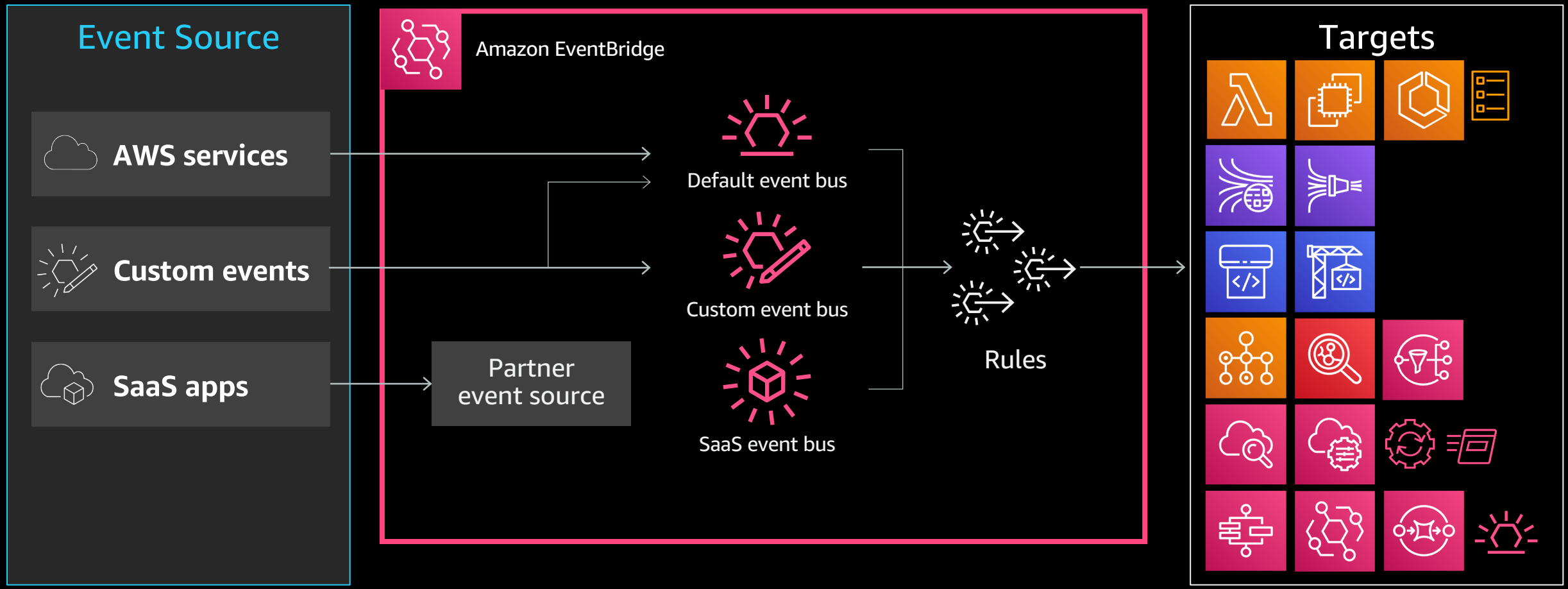
- Serverless—pay only for the events you process
- Simplified scaling avoids increasing costs to sustain and manage resources
- No upfront investments, ongoing licensing, or maintenance costs
- No specialist knowledge needed

@mavi888uy
#edaday

Key features of an event bus

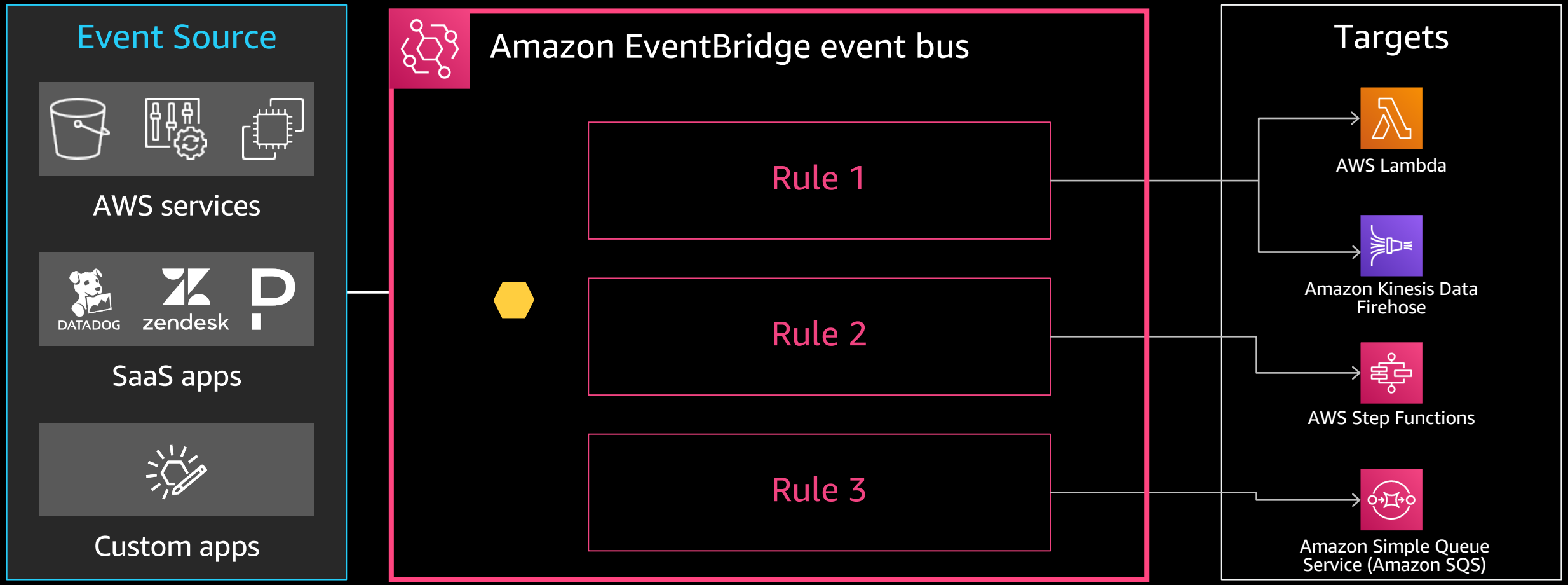


EventBridge architecture



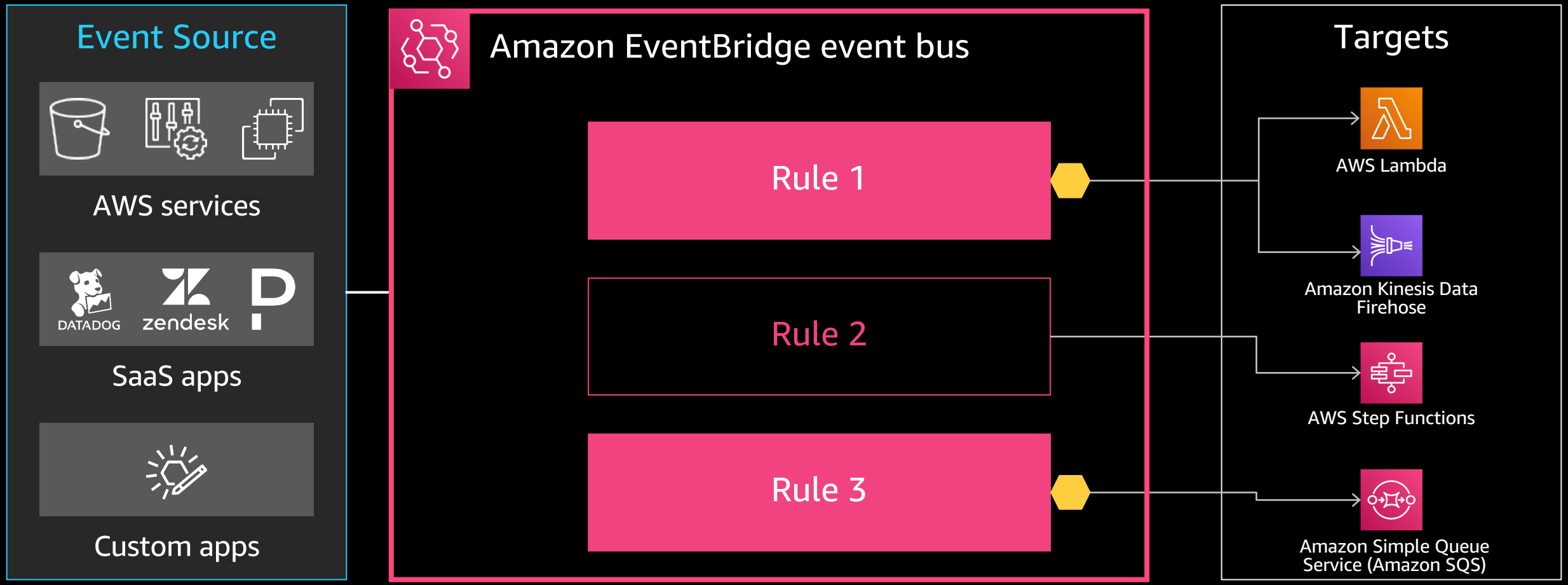
@mavi888uy
#edaday

How the rule matching works?



@mavi888uy
#edaday

How the rule matching works?



@mavi888uy
#edaday

How to design an event-driven application?

Considerations when designing a service with event-driven architectures

Input

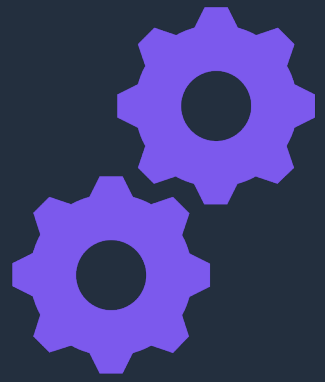
Which events I listen to

Internal

What I do with the events

Output

Which events I produce



Service

Event uniqueness

Idempotency

Operation will return the same results whether it is called once or multiple times

Idempotency key

Assigned to the message by the sender to simplify deduplication by the receiver

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
      "idempotency-key": "c1b95b88"
    },
    "data": {
      "order-id": "1073459984"
    }
}
```

Considerations when designing a service with event-driven architectures

Input

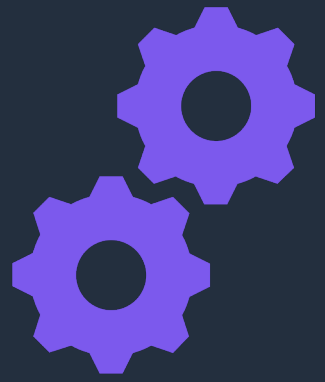
Which events I listen to

Internal

What I do with the events

Output

Which events I produce



Service

Sparse events vs. full state descriptions

Order 123 was created at 10:47 a.m. by customer 456

Sparse events

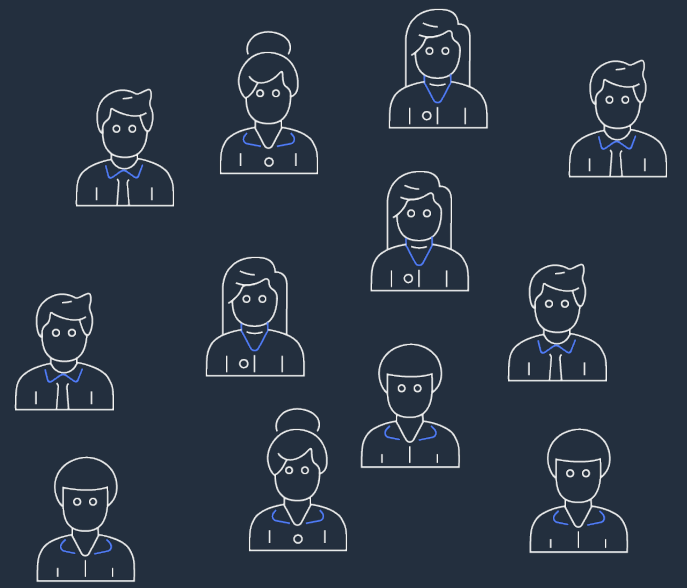
Order 123 was created at 10:47 a.m. by customer 456. The current status is Open, the total was $237.51, the items were …

Full state description

Considerations with sparse events

Order 123 was created by customer 456

What are the details for order 123?



Considerations with full state descriptions

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
      "idempotency-key": "c1b95b88"
    },
    "data": {
      "order-id": "1073459984",
      "status": "Open",
      "total": "237.51"
    }
}
```

- Event schemas should be backwards compatible
- EventBridge payload size 256KB

Considerations when designing a service with event-driven architectures

Input

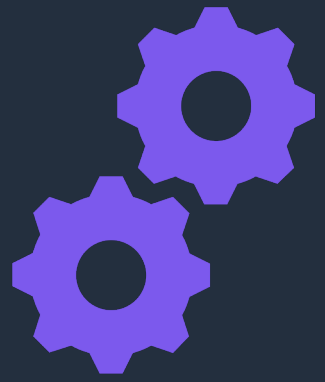
Which events I listen to

Internal

What I do with the events

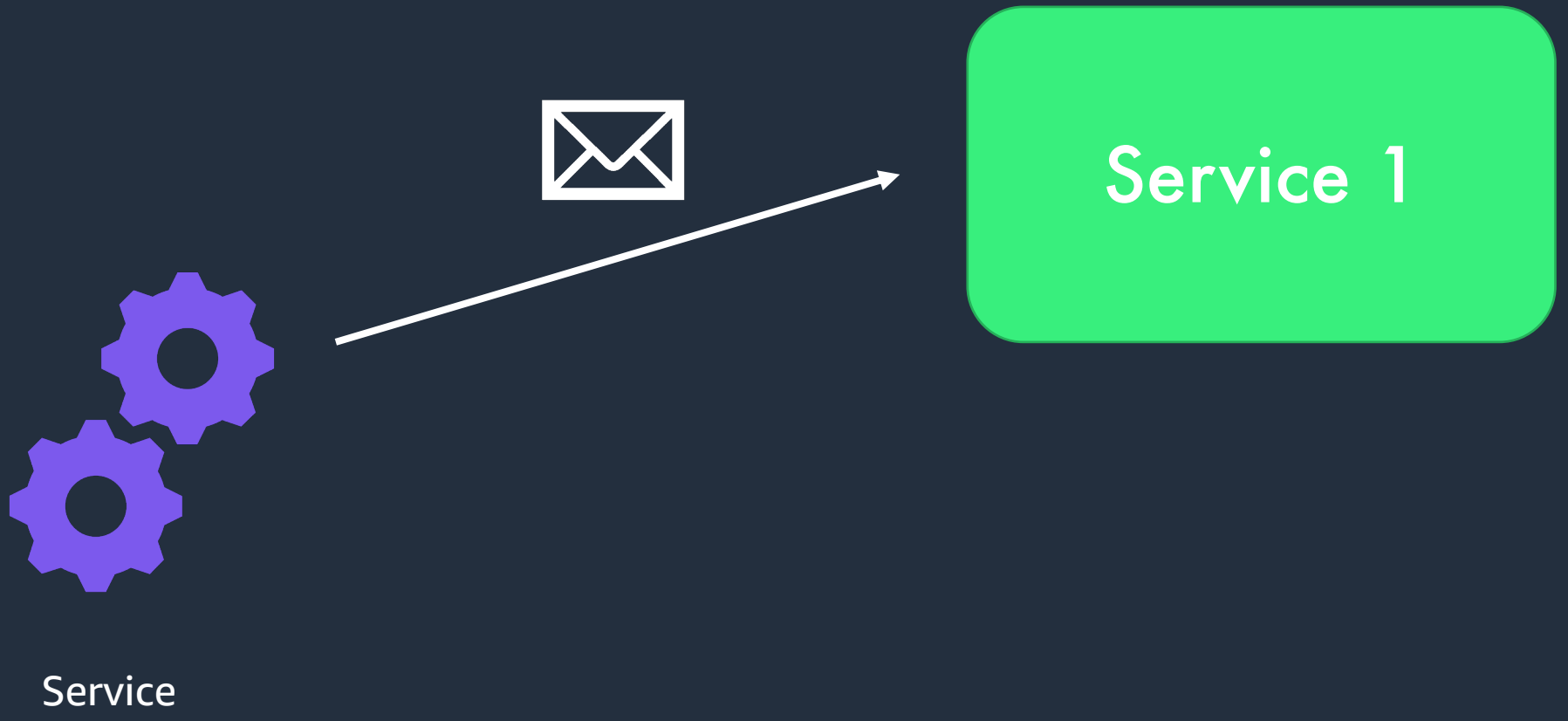
Output

Which events I produce

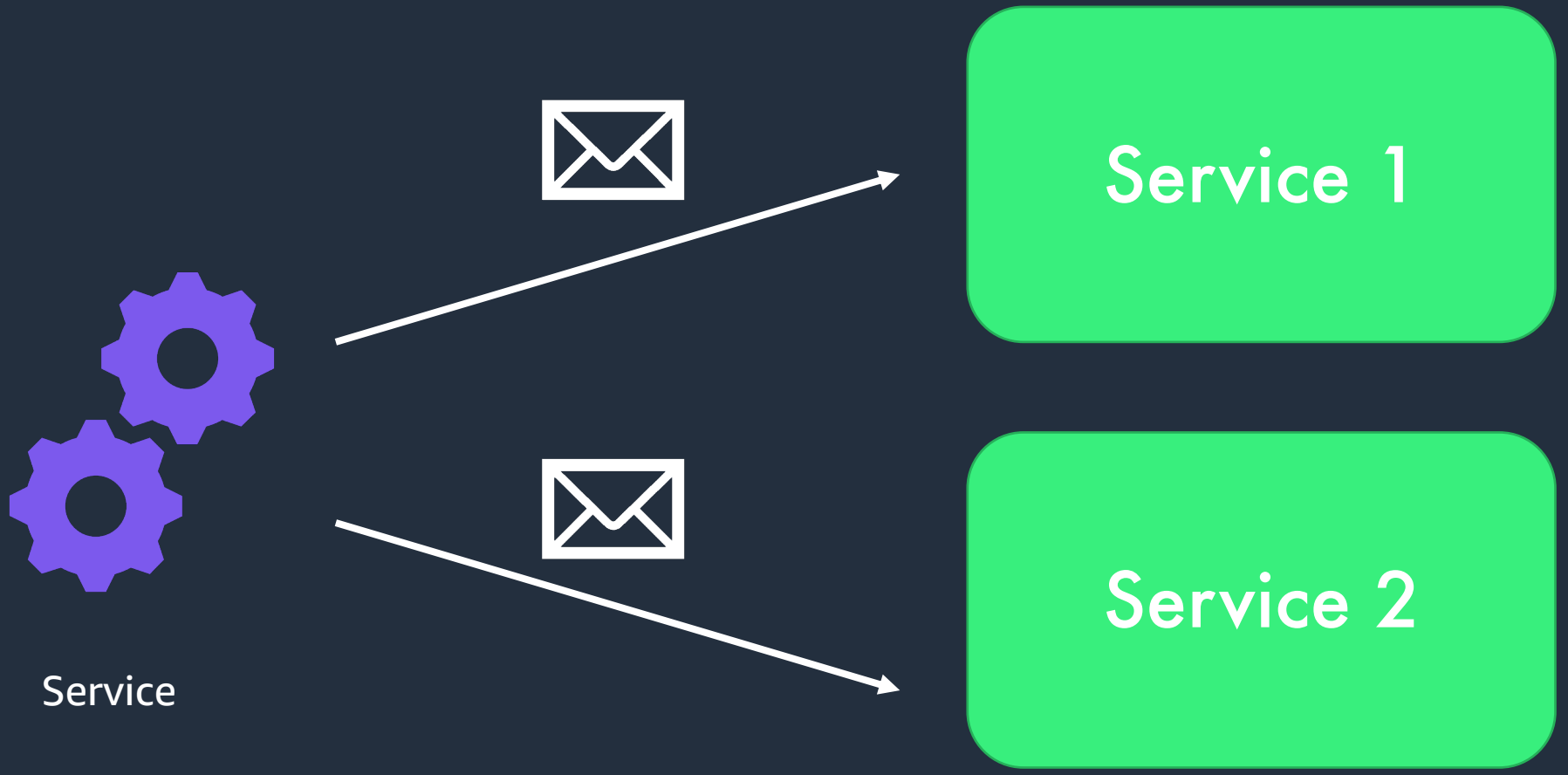


Service

Considerations when outputting events



Considerations when outputting events



Considerations when outputting events

High level architecture

Web Store

Create
View
Cancel

View
Check

Orders

Check
Update
Reserve

Inventory

Pay
Cancel

Payments

Get information

Customer

Start
View
Cancel

Delivery

Application flows

Create an order

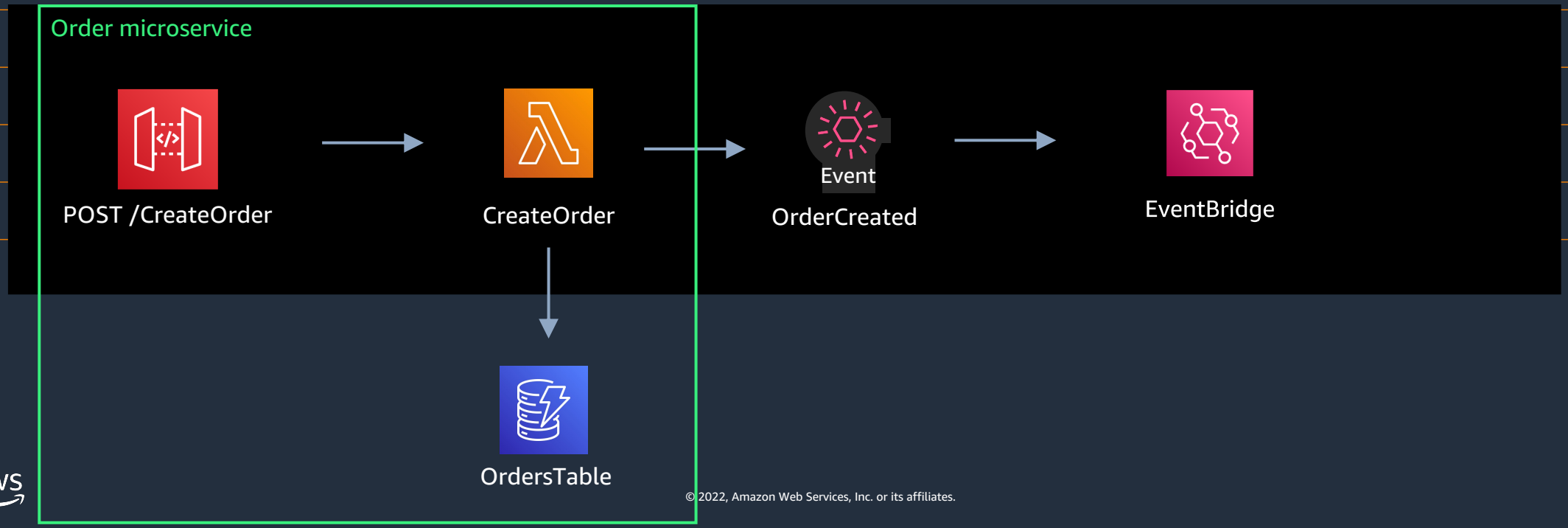
DESIGN TABLE

API (sync)	Input Events	Service	Action	Output Events

Create an order

DESIGN TABLE

API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated



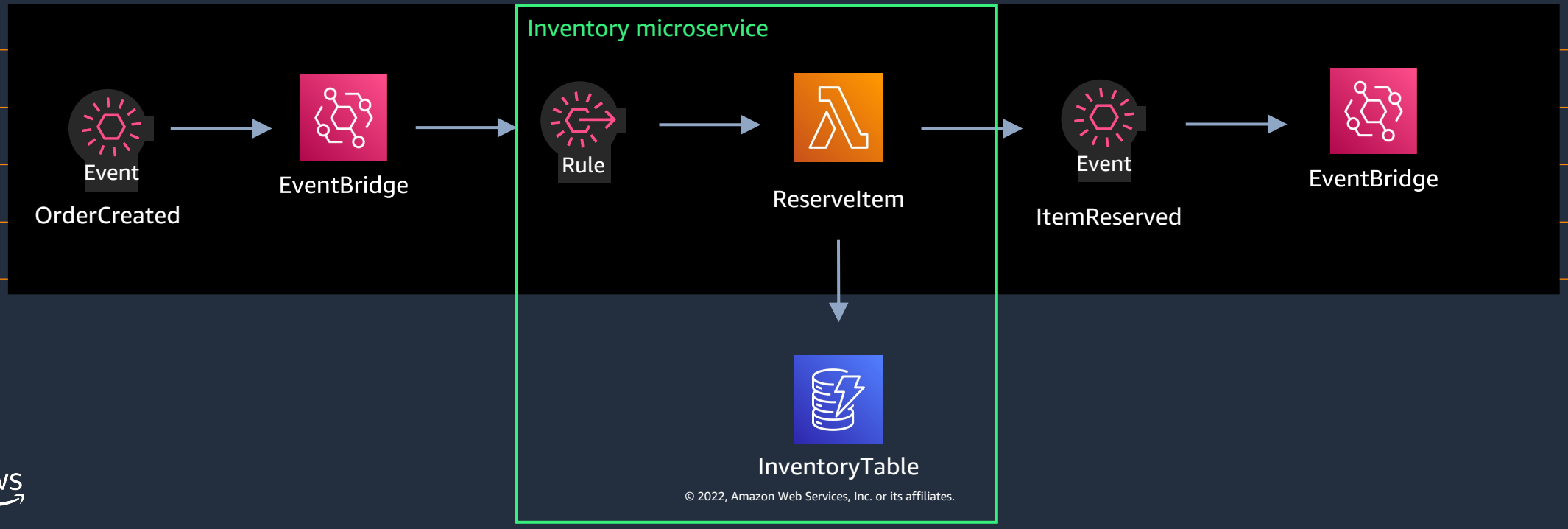
Event Table

Name	Attributes	Service	Action
OrderCreated	CustomerId ProductId OrderId	Order	Create Order

Create an order

DESIGN TABLE

API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated
	OrderCreated	Inventory	ReserveItem	ItemReserved ItemNotAvailable



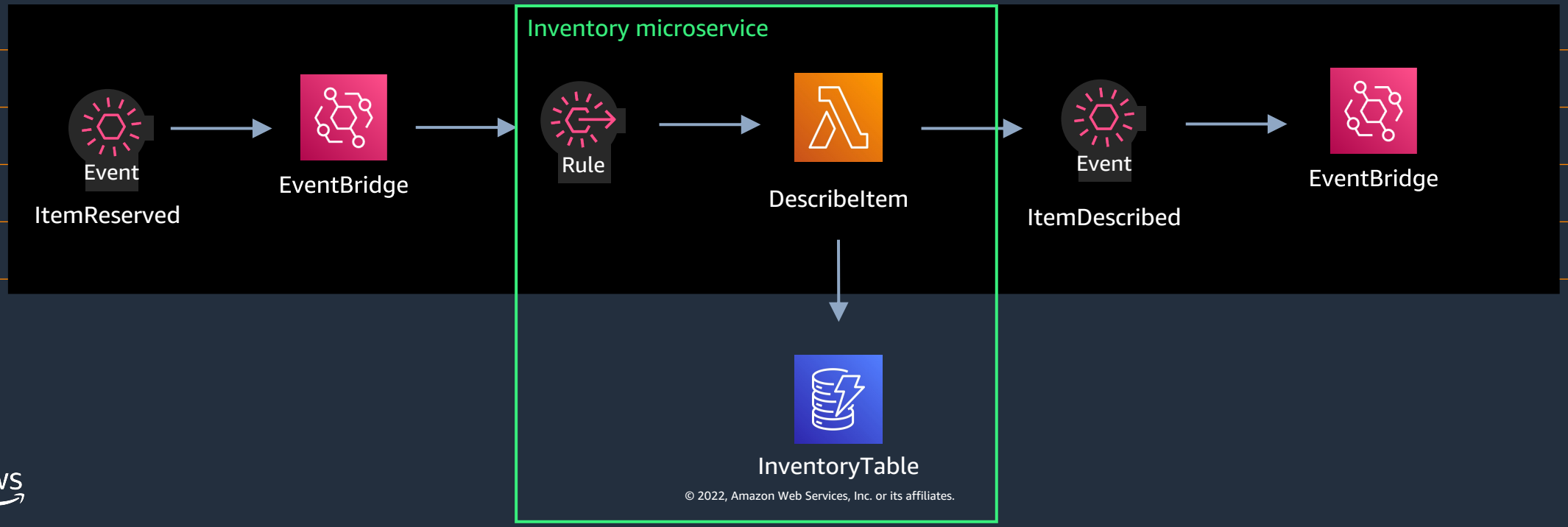
Event Table

Name	Attributes	Service	Action
OrderCreated	CustomerId ProductId OrderId	Order	Create Order
ItemReserved	CustomerId ProductId OrderId	Inventory	Reserve Item

Create an order

DESIGN TABLE

API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated
	OrderCreated	Inventory	ReserveItem	ItemReserved ItemNotAvailable
	ItemReserved	Inventory	DescribeItem	ItemDescribed



Create an order

DESIGN TABLE

API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated
	OrderCreated	Inventory	ReserveItem	ItemReserved ItemNotAvailable
	ItemReserved	Inventory	DescribeItem	ItemDescribed
	ItemDescribed	Customer	DescribeCustomer	CustomerDescribed ErrorCustomerDescribed

Event Table

Name	Attributes	Service	Action
OrderCreated	CustomerId ProductId OrderId	Order	Create Order
ItemReserved	CustomerId ProductId OrderId	Inventory	Reserve Item
ItemDescribed	CustomerId ProductId Price OrderId	Inventory	Describe Item

Create an order

DESIGN TABLE

API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated
	OrderCreated	Inventory	ReserveItem	ItemReserved ItemNotAvailable
	ItemReserved	Inventory	DescribeItem	ItemDescribed
	ItemDescribed	Customer	DescribeCustomer	CustomerDescribed ErrorCustomerDescribed
	CustomerDescribed	Delivery	EstimateDelivery	DeliveryEstimated

Create an order

DESIGN TABLE

API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated
	OrderCreated	Inventory	ReserveItem	ItemReserved ItemNotAvailable
	ItemReserved	Inventory	DescribeItem	ItemDescribed
	ItemDescribed	Customer	DescribeCustomer	CustomerDescribed ErrorCustomerDescribed
	CustomerDescribed	Delivery	EstimateDelivery	DeliveryEstimated
	DeliveryEstimated	Payment	MakePayment	PaymentMade PaymentFailed

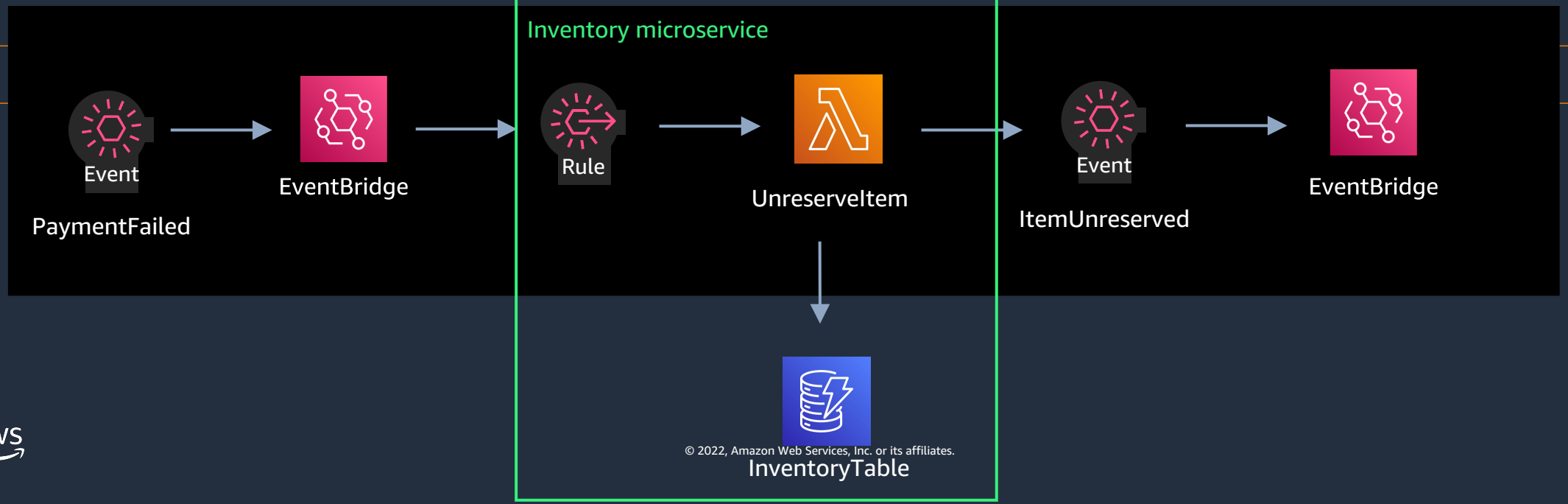
Create an order

DESIGN TABLE

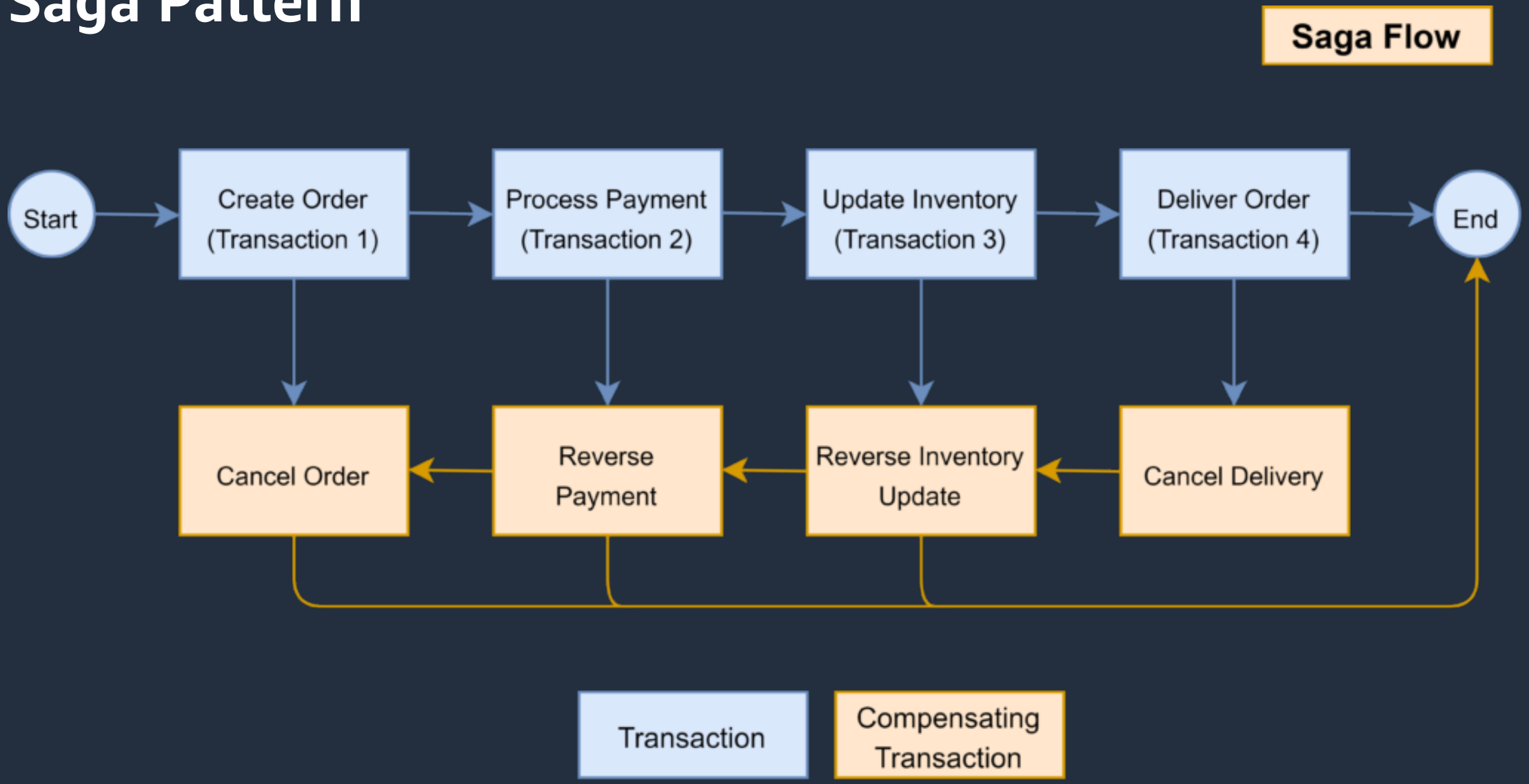
API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated
	OrderCreated	Inventory	ReserveItem	ItemReserved ItemNotAvailable
	ItemReserved	Inventory	DescribeItem	ItemDescribed
	ItemDescribed	Customer	DescribeCustomer	CustomerDescribed ErrorCustomerDescribed
	CustomerDescribed	Delivery	EstimateDelivery	DeliveryEstimated
	DeliveryEstimated	Payment	MakePayment	PaymentMade PaymentFailed
	PaymentMade	Inventory	RemoveItem	ItemRemoved ErrorItemUnreserved
	ItemRemoved	Delivery	StartDelivery	DeliveryStarted
…				

Error handling with events

API (sync)	Input events	Service	Action	Output events
	…			
	DeliveryEstimated	Payment	MakePayment	PaymentMade PaymentFailed
	PaymentFailed	Inventory	UnreserveItem	ItemUnreserved (END) ErrorItemUnreserved
…				



Saga Pattern



https://www.baeldung.com/cs/saga-pattern-microservices

Add a new business requirement

Sending emails to keep customers updated

→ Adding a **notification service**

Add a new business requirement

ADD A NOTIFICATION SERVICE

API (sync)	Input events	Service	Action	Output events
CreateOrder		Order	CreateOrder	OrderCreated
	OrderCreated	Inventory	ReserveItem	ItemReserved ItemNotAvailable
	ItemReserved	Inventory	DescribeItem	ItemDescribed
	ItemDescribed	Customer	DescribeCustomer	CustomerDescribed ErrorCustomerDescribed
	CustomerDescribed	Delivery	EstimateDelivery	DeliveryEstimated
	DeliveryEstimated	Payment	MakePayment	PaymentMade PaymentFailed
	PaymentMade	Inventory	RemoveItem	ItemRemoved ErrorItemUnreserved
	ItemRemoved	Delivery	StartDelivery	DeliveryStarted
...				

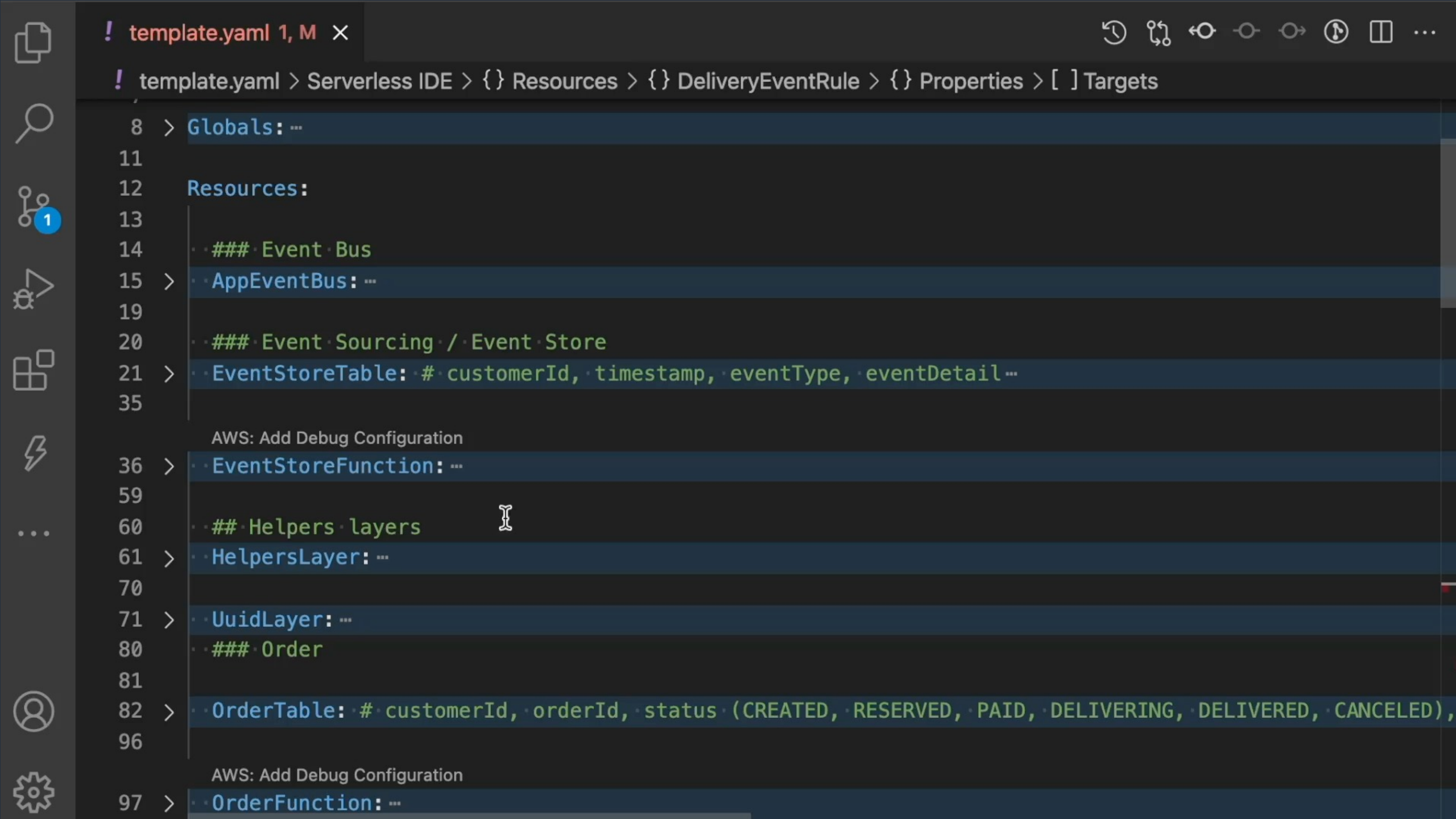
Add a new business requirement

ADD A NOTIFICATION SERVICE

API (sync)	Input Events	Service	Action	Output Events
	ItemNotAvailable PaymentMade PaymentFailed …	Notifications	SendEmail	EmailSent

Looking for existing events to listen to:
No changes, based on existing flow

DEMO


template.yaml 1, M
template.yaml > Serverless IDE > { } Resources > { } DeliveryEventRule > { } Properties > [ ] Targets
8 Globals: ···
11
12 Resources:
13
14 ### Event Bus
15 AppEventBus: ···
19
20 ### Event Sourcing / Event Store
21 EventStoreTable: # customerId, timestamp, eventType, eventDetail ···
35
AWS: Add Debug Configuration
36 EventStoreFunction: ···
59
60 ## Helpers layers
61 HelpersLayer: ···
70
71 UuidLayer: ···
80 ### Order
81
82 OrderTable: # customerId, orderId, status (CREATED, RESERVED, PAID, DELIVERING, DELIVERED, CANCELED),
96
AWS: Add Debug Configuration
97 OrderFunction: ···

template.yaml 1, M | 2igkaf6p9j.execute-api.eu.. | app.js

GET https://h8a3qpbf44.execute-api.eu-west-1.amazonaws.com/order/create/customer-1/item-2 Send

Query Headers 2 Auth Body Tests

Query Parameters

parameter value

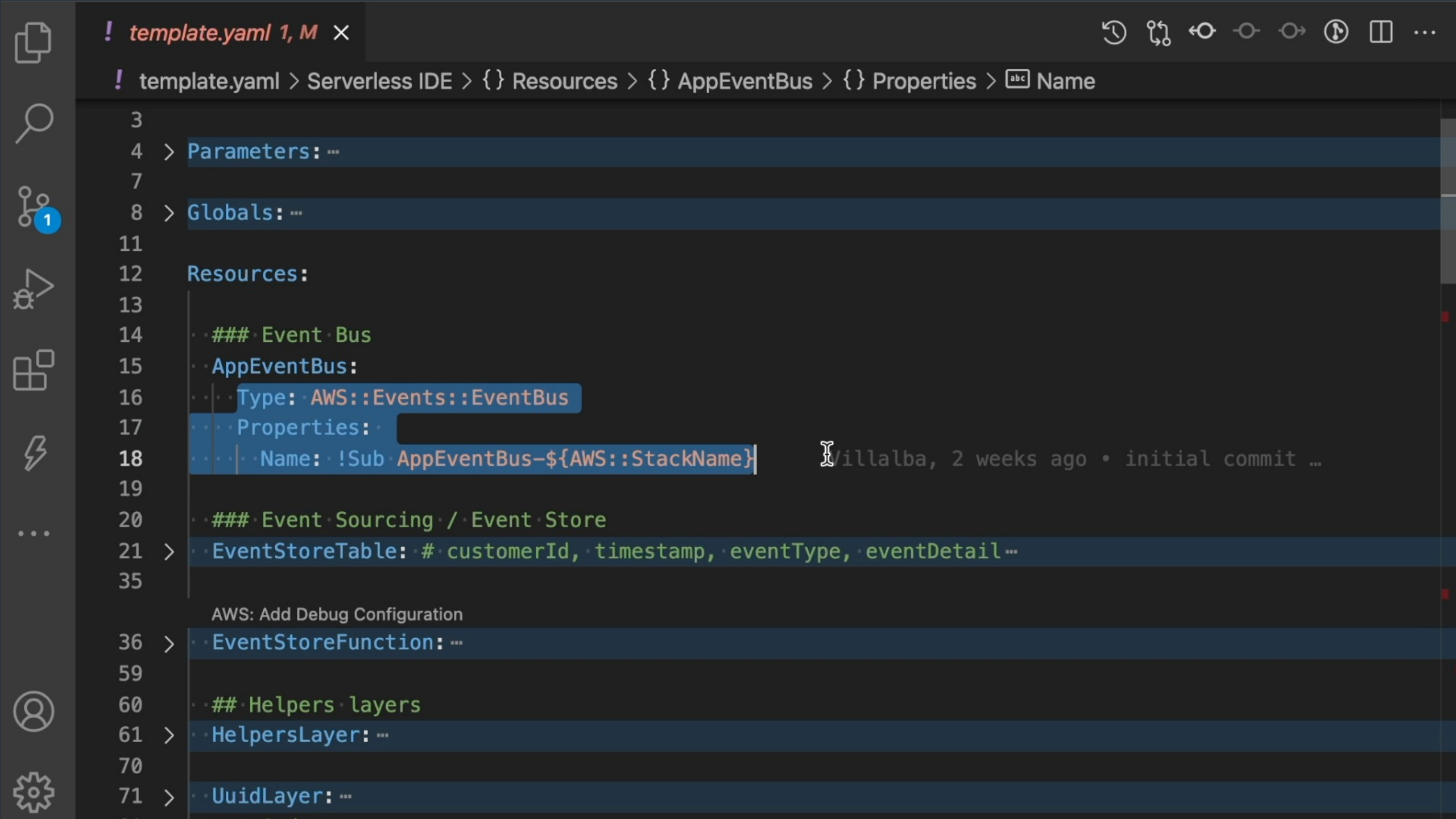
Status: 200 OK Size: 82 Bytes Time: 2.00 s

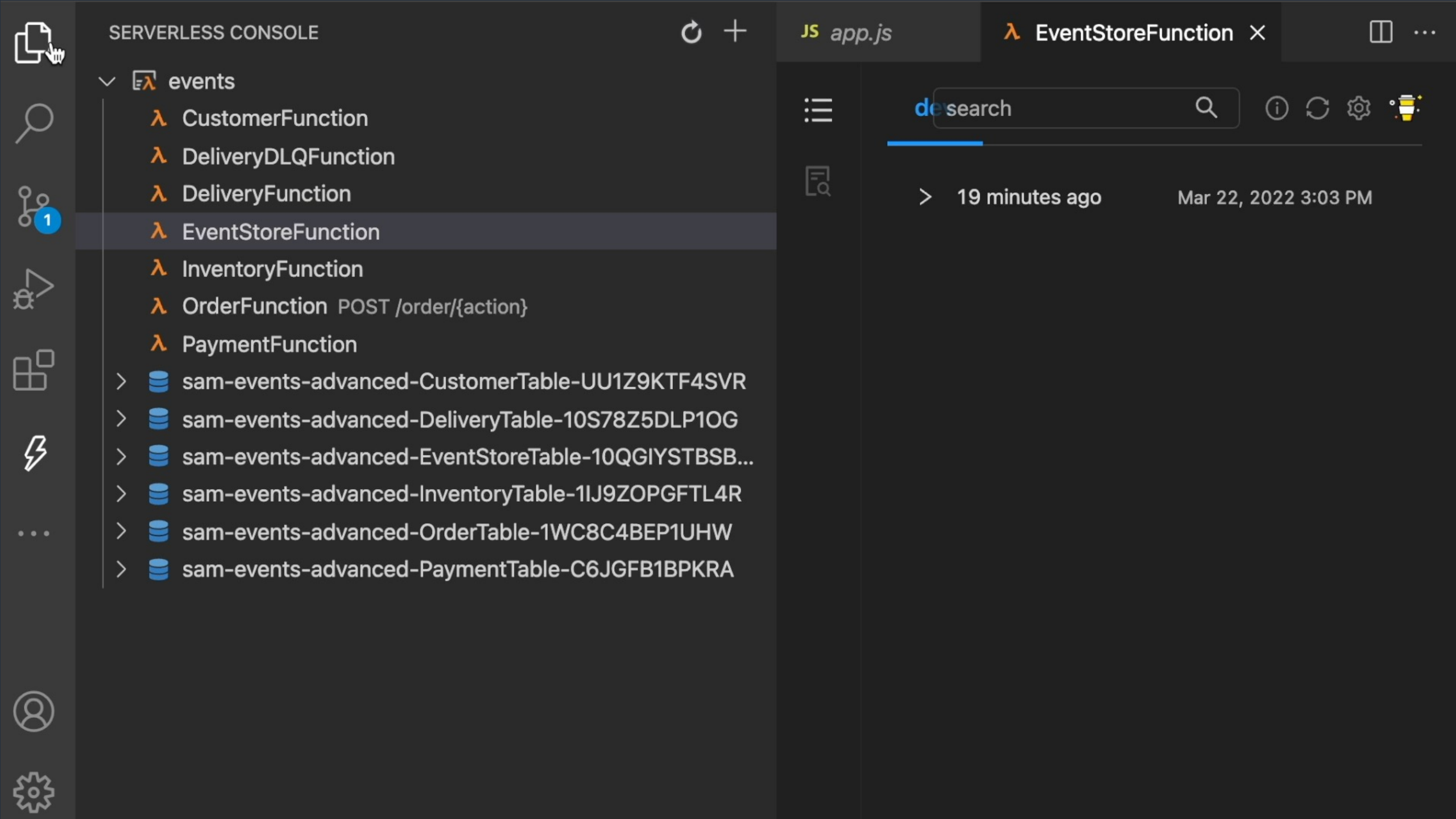
Response Headers 5 Cookies Test Results

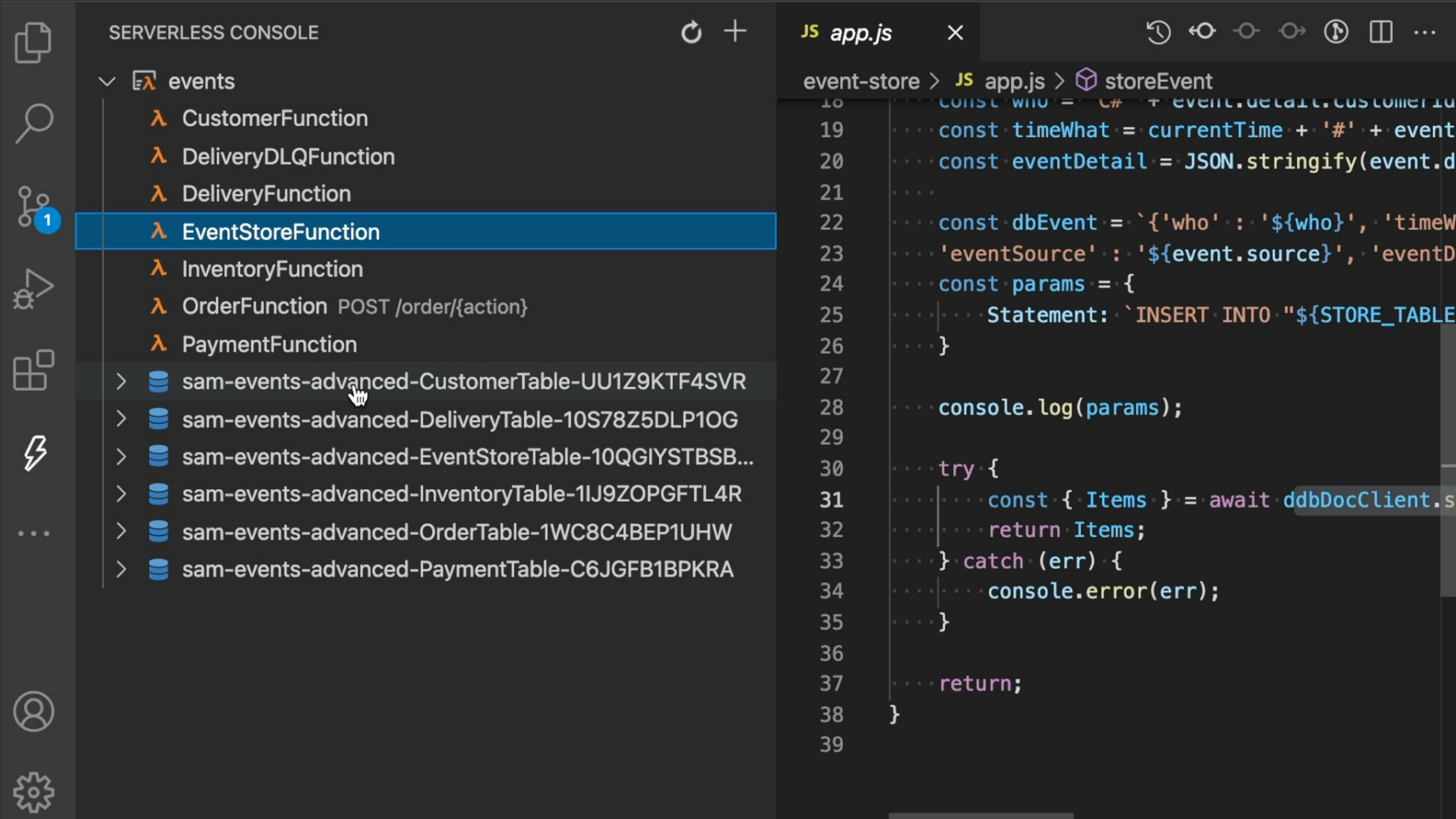

```
{
  "customerId": "customer-1",
  "orderId": "2022-03-22T13:02:51.919Z",
  "itemId": "item-2"
}
```

Best practices for using Amazon EventBridge

Keep track of all events


template.yaml 1, M
template.yaml > Serverless IDE > { } Resources > { } AppEventBus > { } Properties > abc Name
3
4 Parameters: ···
7
8 Globals: ···
11
12 Resources:
13
14 ### Event Bus
15 AppEventBus:
16 Type: AWS::Events::EventBus
17 Properties:
18 Name: !Sub AppEventBus-${AWS::StackName}
Villalba, 2 weeks ago • initial commit …
19
20 ### Event Sourcing / Event Store
21 EventStoreTable: # customerId, timestamp, eventType, eventDetail ···
35
AWS: Add Debug Configuration
36 EventStoreFunction: ···
59
60 ## Helpers layers
61 HelpersLayer: ···
70
71 UuidLayer: ···


SERVERLESS CONSOLE
events
CustomerFunction
DeliveryDLQFunction
DeliveryFunction
EventStoreFunction
InventoryFunction
OrderFunction POST /order/{action}
PaymentFunction
sam-events-advanced-CustomerTable-UU1Z9KTF4SVR
sam-events-advanced-DeliveryTable-10S78Z5DLP1OG
sam-events-advanced-EventStoreTable-10QGIYSTBSB...
sam-events-advanced-InventoryTable-1IJ9ZOPGFTL4R
sam-events-advanced-OrderTable-1WC8C4BEP1UHW
sam-events-advanced-PaymentTable-C6JGFB1BPKRA
app.js
EventStoreFunction
search
19 minutes ago
Mar 22, 2022 3:03 PM

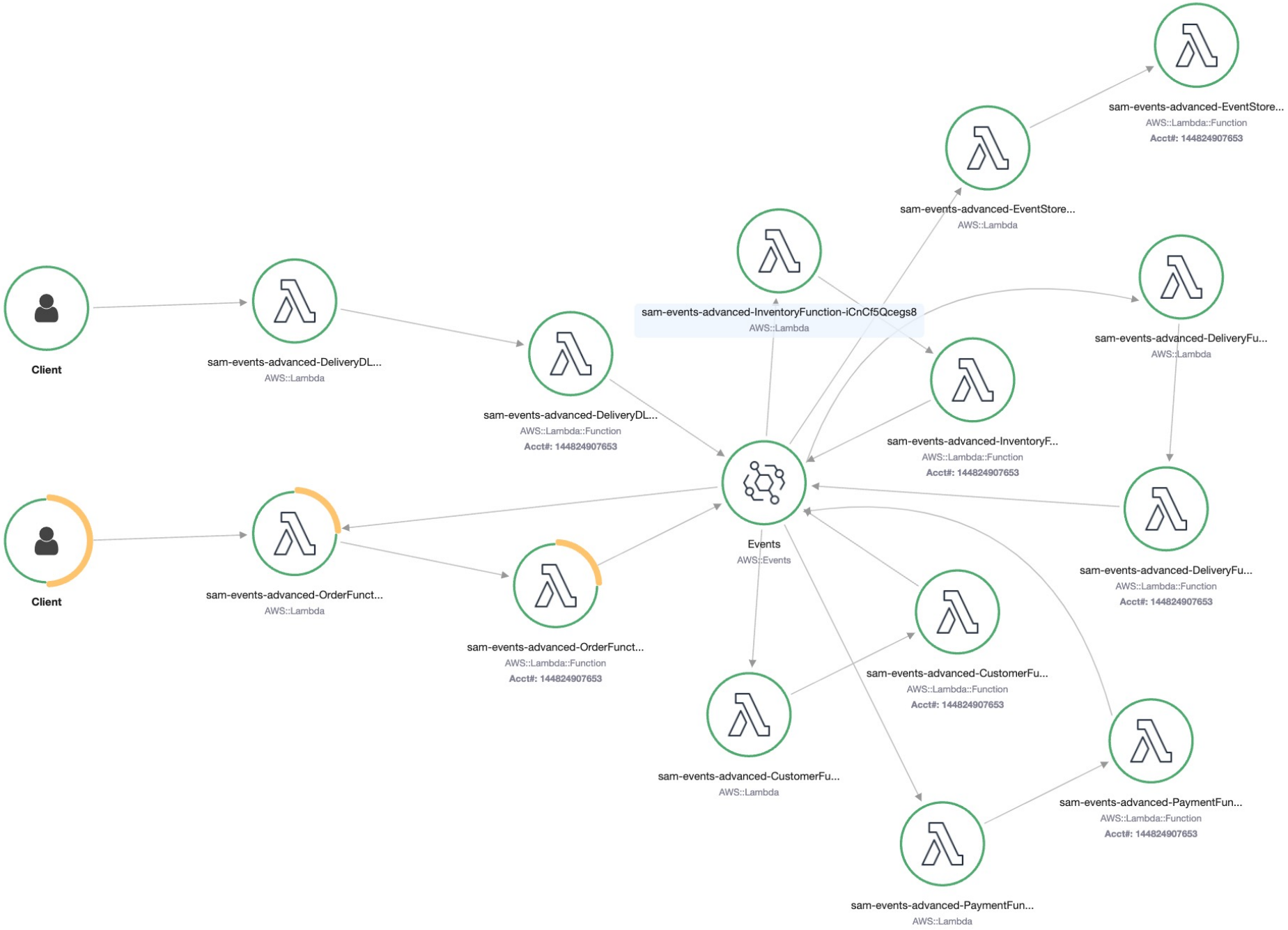

SERVERLESS CONSOLE
events
CustomerFunction
DeliveryDLQFunction
DeliveryFunction
EventStoreFunction
InventoryFunction
OrderFunction POST /order/{action}
PaymentFunction
sam-events-advanced-CustomerTable-UU1Z9KTF4SVR
sam-events-advanced-DeliveryTable-10S78Z5DLP1OG
sam-events-advanced-EventStoreTable-10QGIYSTBSB...
sam-events-advanced-InventoryTable-1IJ9ZOPGFTL4R
sam-events-advanced-OrderTable-1WC8C4BEP1UHW
sam-events-advanced-PaymentTable-C6JGFB1BPKRA
app.js
event-store > app.js > storeEvent
const timeWhat = currentTime + '#' + event
const eventDetail = JSON.stringify(event.d
const dbEvent = `{'who' : '${who}', 'timeW
'eventSource' : '${event.source}', 'eventD
const params = {
Statement: `INSERT INTO "${STORE_TABLE
}
console.log(params);
try {
const { Items } = await ddbDocClient.s
return Items;
} catch (err) {
console.error(err);
}
return;
}

Tracing with AWS X-Ray

- Only for Custom events (PutEvents) that contains the tracing header
- Trace headers are passed to the targets

```
const AWSXRay = require('aws-xray-sdk');
const { EventBridgeClient, PutEventsCommand } = require("@aws-sdk/client-
eventbridge");

const ebClient = AWSXRay.captureAWSv3Client(new EventBridgeClient({}));
```


Client
sam-events-advanced-DeliveryDL...
AWS::Lambda
sam-events-advanced-DeliveryDL...
AWS::Lambda::Function
Acct#: 144824907653
sam-events-advanced-InventoryFunction-iCnCf5Qcegs8
AWS::Lambda
sam-events-advanced-EventStore...
AWS::Lambda
sam-events-advanced-EventStore...
AWS::Lambda::Function
Acct#: 144824907653
sam-events-advanced-DeliveryFu...
AWS::Lambda
sam-events-advanced-InventoryF...
AWS::Lambda::Function
Acct#: 144824907653
Client
sam-events-advanced-OrderFunct...
AWS::Lambda
Events
AWS::Events
sam-events-advanced-DeliveryFu...
AWS::Lambda::Function
Acct#: 144824907653
sam-events-advanced-OrderFunct...
AWS::Lambda::Function
Acct#: 144824907653
sam-events-advanced-CustomerFu...
AWS::Lambda::Function
Acct#: 144824907653
sam-events-advanced-CustomerFu...
AWS::Lambda
sam-events-advanced-PaymentFun...
AWS::Lambda::Function
Acct#: 144824907653
sam-events-advanced-PaymentFun...
AWS::Lambda

API Destinations

Connection

- Specifies the authorization type:
  - Basic (Username/Password)
  - OAuth
  - API Key
- Can be reused across API destinations

API destinations

- Utilize a connection for authorization
- HTTPS endpoint
- HTTP method
- Rate limit (per second)
- Must finish withing 5 seconds

API Destinations

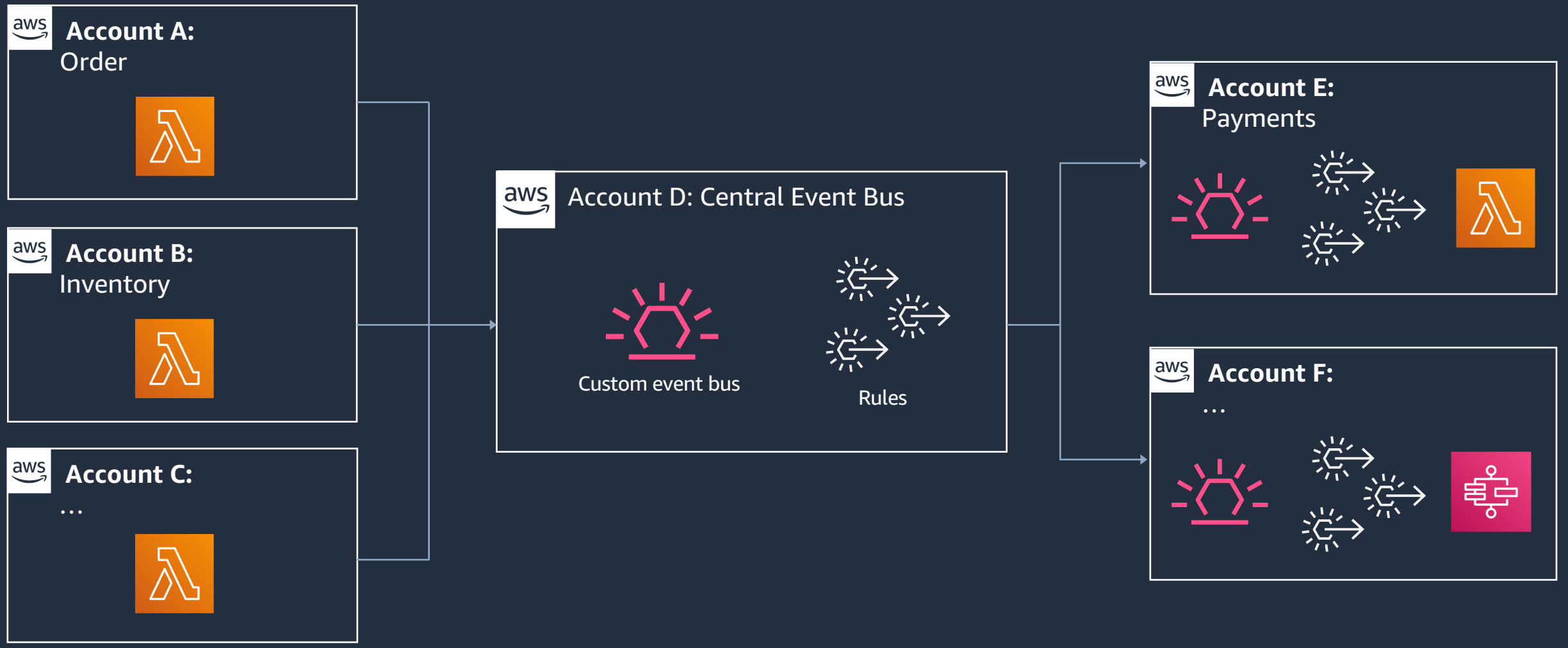
```
DeliveryConnection:
  Type: AWS::Events::Connection
  Properties:
    AuthorizationType: API_KEY
    Description: 'Connection with an API key’
    AuthParameters:
      ApiKeyAuthParameters:
        ApiKeyName: ApiKeyName
        ApiKeyValue: DeliveryServiceApiKeyValue

DeliveryApiDestination:
  Type: AWS::Events::ApiDestination
  Properties:
    Name: 'DeliveryApiDestination’
    ConnectionArn: !GetAtt DeliveryConnection.Arn
    InvocationEndpoint: !Ref DeliveryServiceURL
    HttpMethod: POST
    InvocationRateLimitPerSecond: 10
```

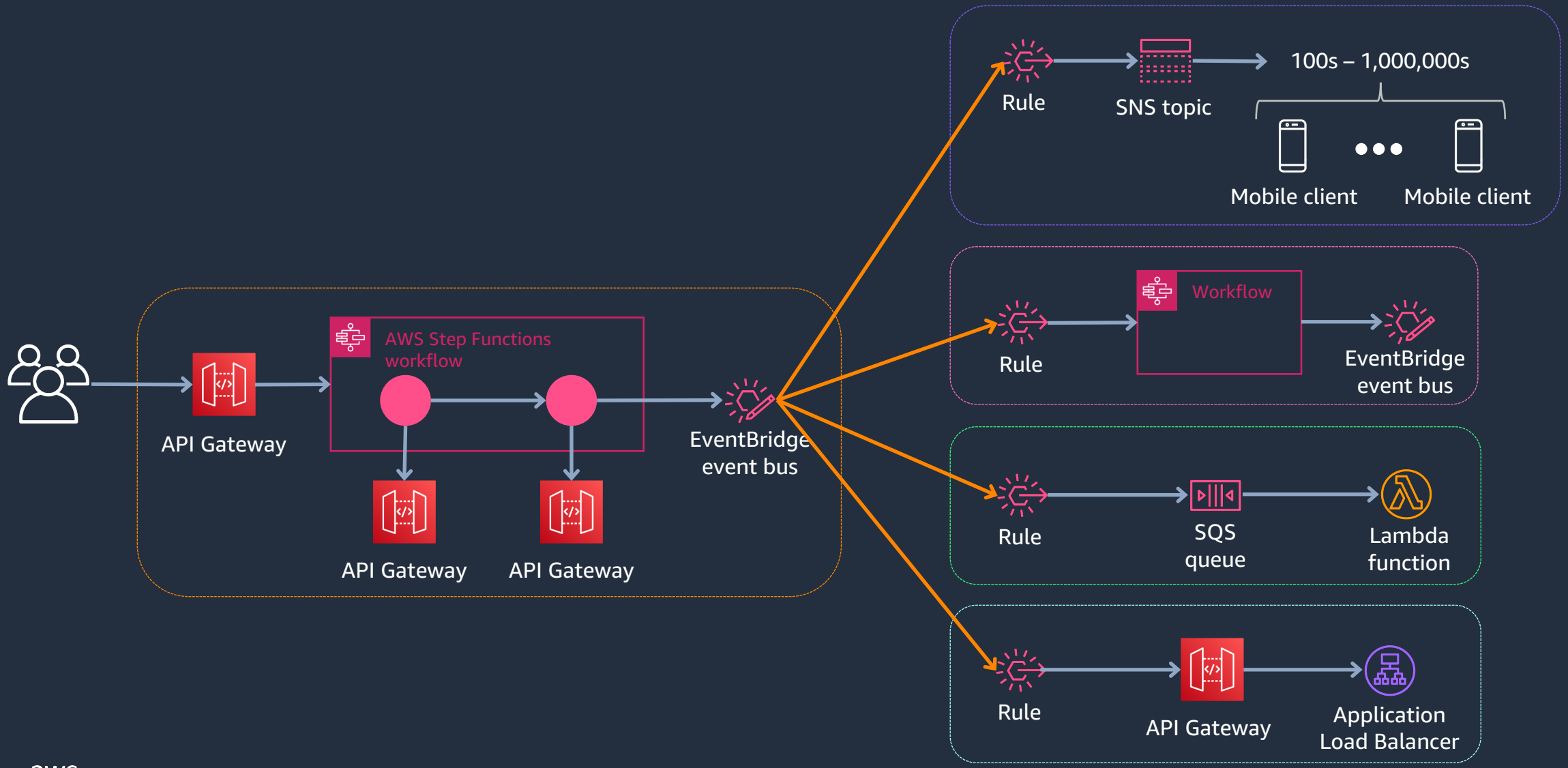
API Destinations + DLQ

```
DeliveryEventRule:
  Type: AWS::Events::Rule
  Properties:
    Description: "Delivery Service EventRule"
    EventBusName: !Ref AppEventBus
    EventPattern:
      detail-type:
        - DeliveryMarkedAsStarted
    Targets:
      - Arn: !GetAtt DeliveryApiDestination.Arn
        RoleArn: !GetAtt DeliveryEBTargetRole.Arn
        Id: "DeliveryApiDestination"
        DeadLetterConfig:
          Arn: !GetAtt DeliveryServiceDLQueue.Arn
          RetryPolicy:
            MaximumEventAgeInSeconds: 60
            MaximumRetryAttempts: 4
```

Multi-account architectures

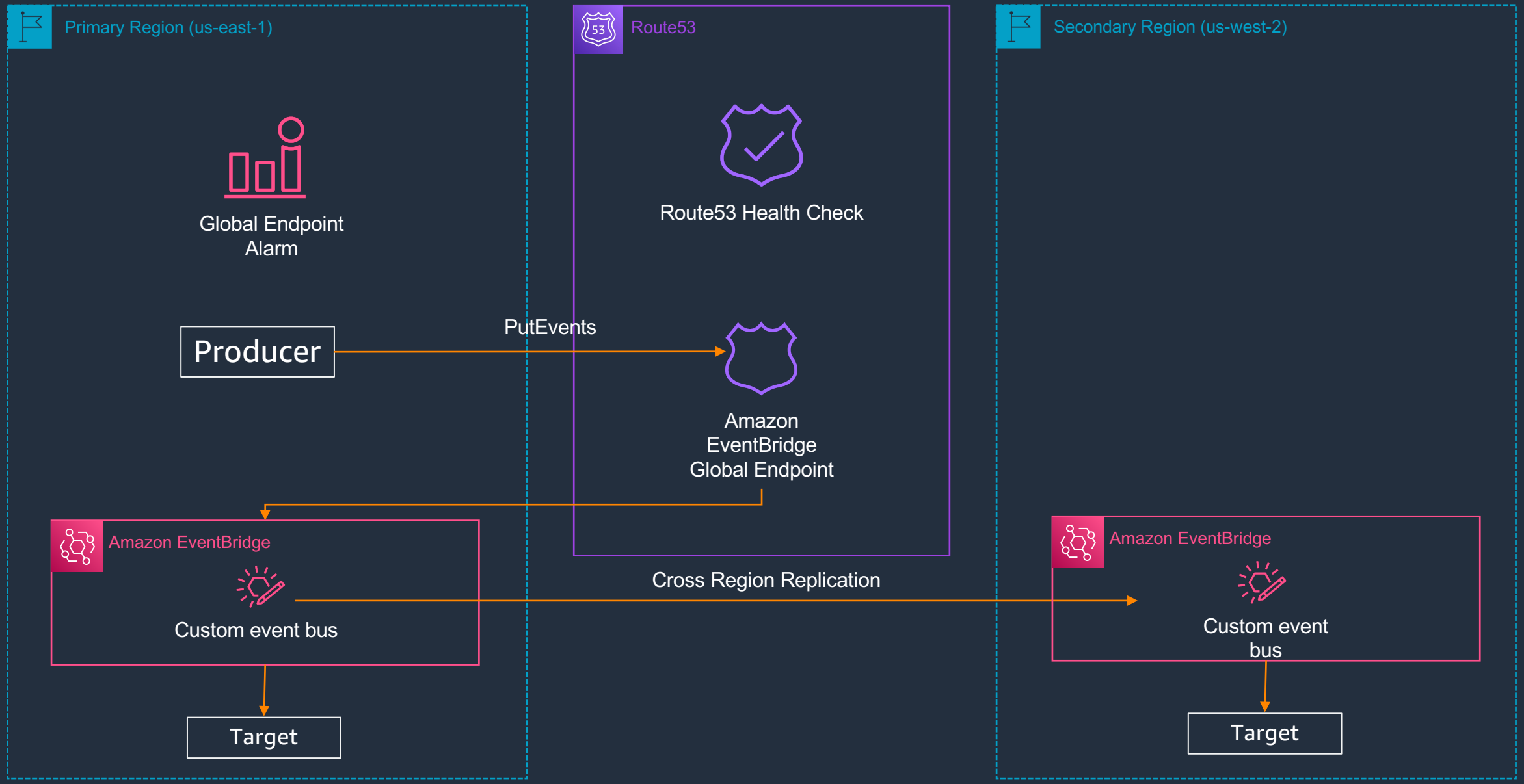


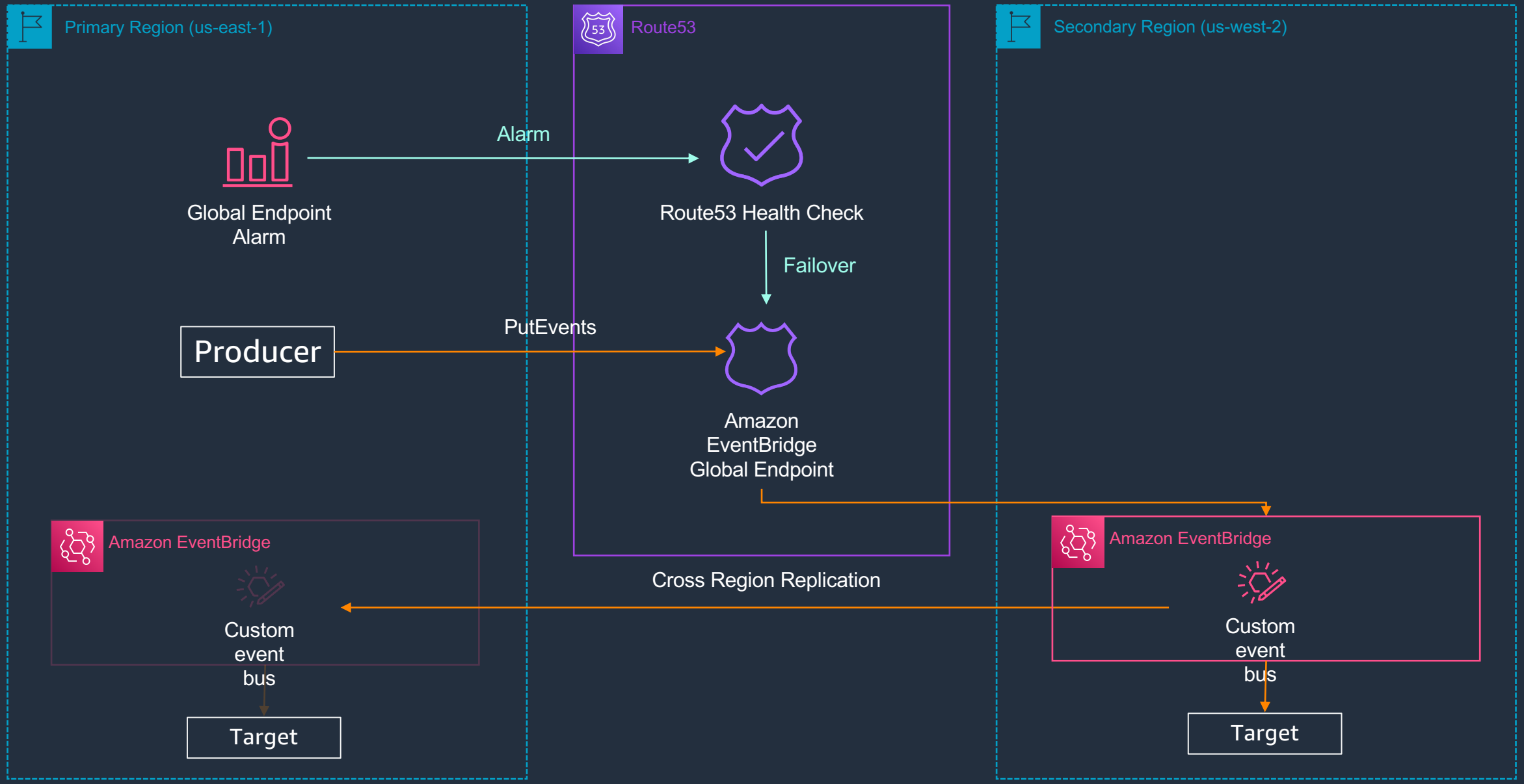
Better together: Orchestration + Choreography



Global Endpoints

- Managed replication
- Automated failover and recovery
- Simplify multi-region HA
- Simplify disaster recovery



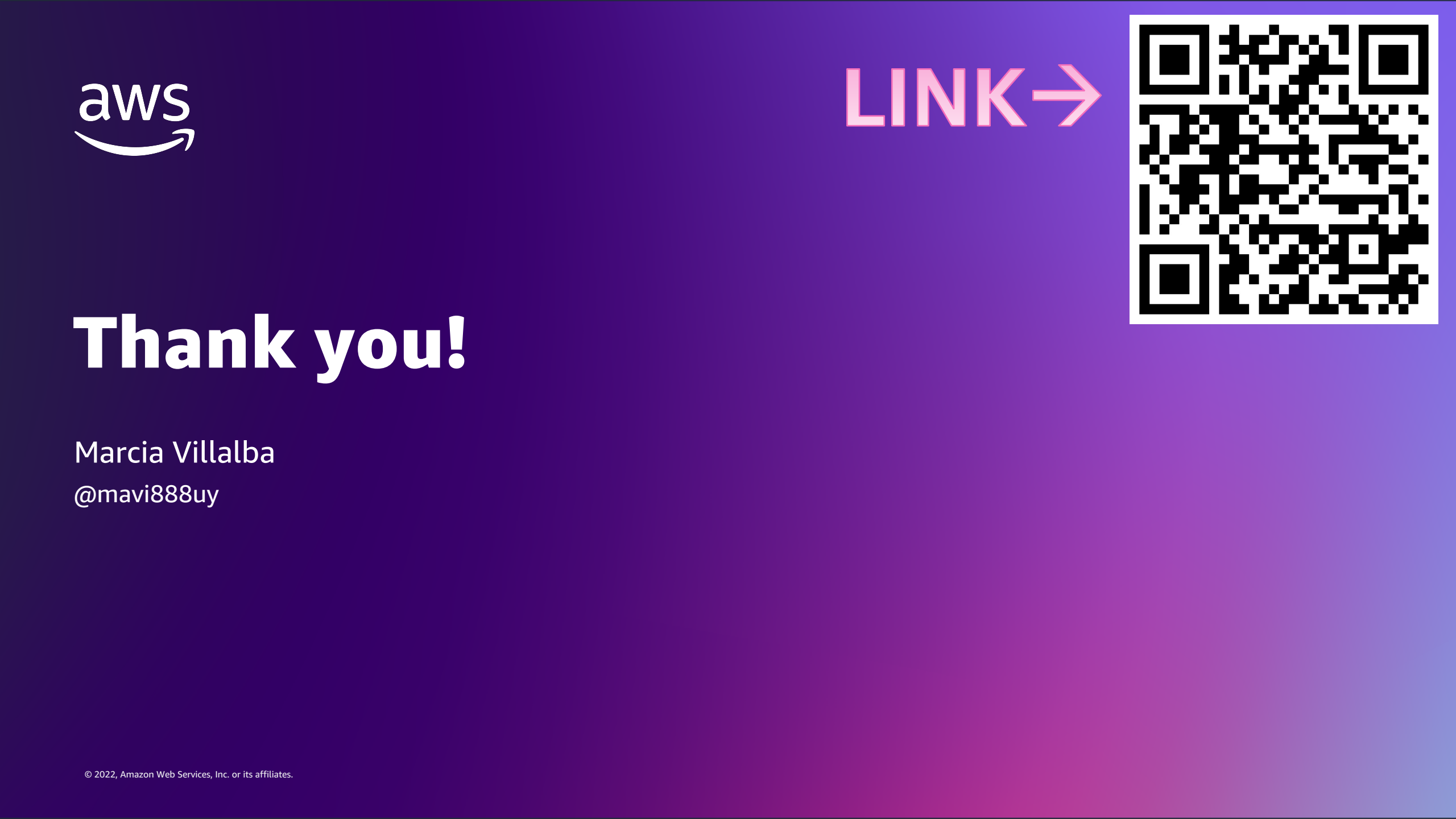


To summarize…

- Event buses helps you to decouple your applications
- Design your use cases using tables – one for your events and one for each use case
- Design the events
- Keep a central log for your events and don’t forget observability
- Keep learning the new Amazon EventBridge features
- And don’t forget you can mix and match with AWS Step Functions, AWS Kinesis, SQS, SNS…

More info about this...

https://blog.marcia.dev/event-driven-applications



Thank you!

Marcia Villalba
@mavi888uy

GOTO
Guide

Download on the App Store

GET IT ON Google Play

LINK→

Let us help you

Ask questions through **the app**

★★★★★

remember to rate the session

THANK YOU

goto;

GOTO
Guide

LINK→

★★★★★

Remember to **rate the session**

Download on the App Store

GET IT ON Google Play

THANK YOU

aws