

BOSTON MEETUP

Amazon EventBridge: Everything serverless developers need to know

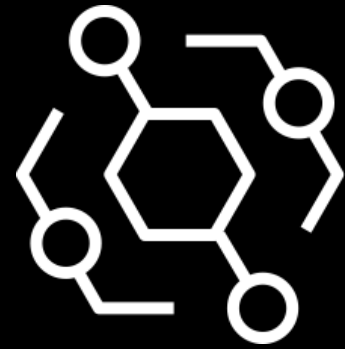
James Beswick
Senior Developer Advocate, AWS Serverless
Amazon Web Services



About me



- **James Beswick**
 - Email: jbeswick@amazon.com
 - Twitter: [@jbesw](https://twitter.com/jbesw)
- Senior Developer Advocate – AWS Serverless
- Serverless geek
- Software developer and PM activist
- Previously:
 - Multiple start-up tech guy
 - Rackspace, USAA, Morgan Stanley, J P Morgan...
 - AWS customer since 2012



Amazon EventBridge

A serverless event bus service
for SaaS and AWS services



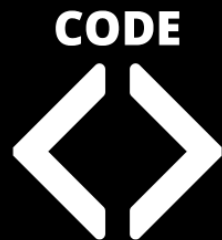
Topics for today



Event-driven architectures

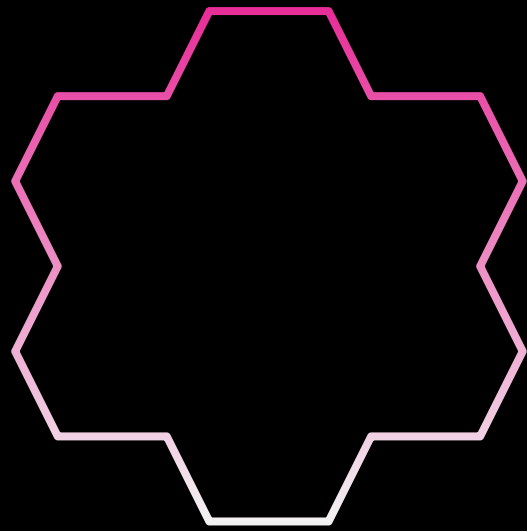


Amazon EventBridge

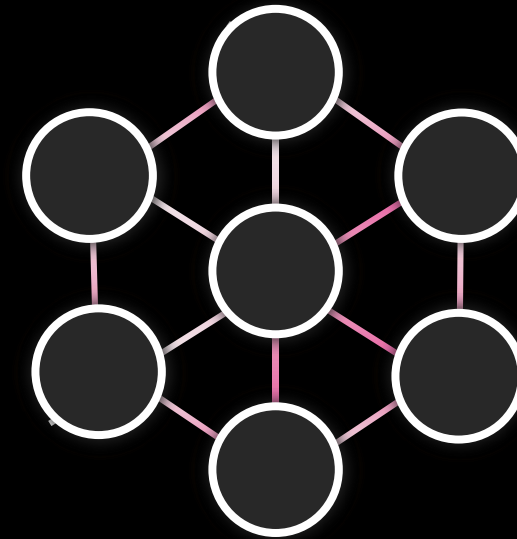


Live demo

Monoliths and microservices



Monolith
Does everything



Microservices
Does one thing

The end goal



Customer value



Reliability



Resilience



Scalability



And do it faster

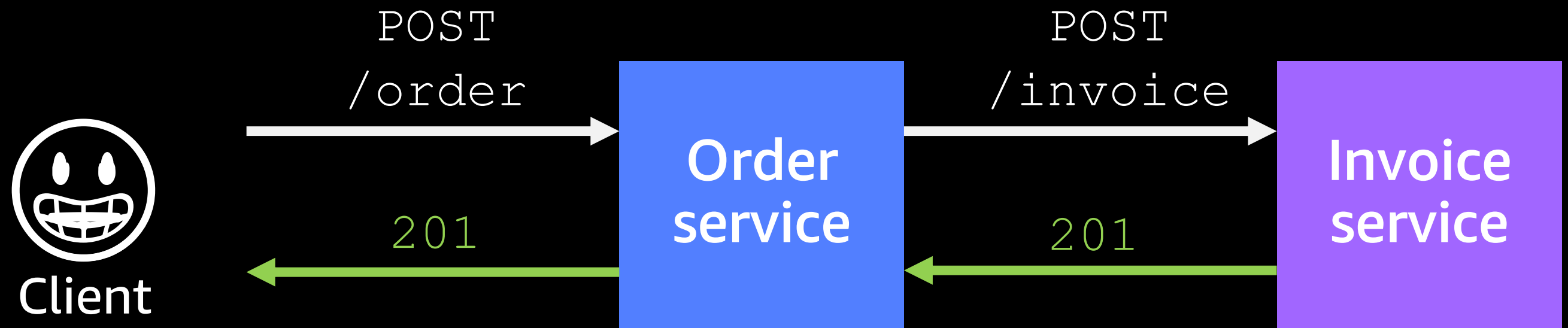
Challenges with distributed systems



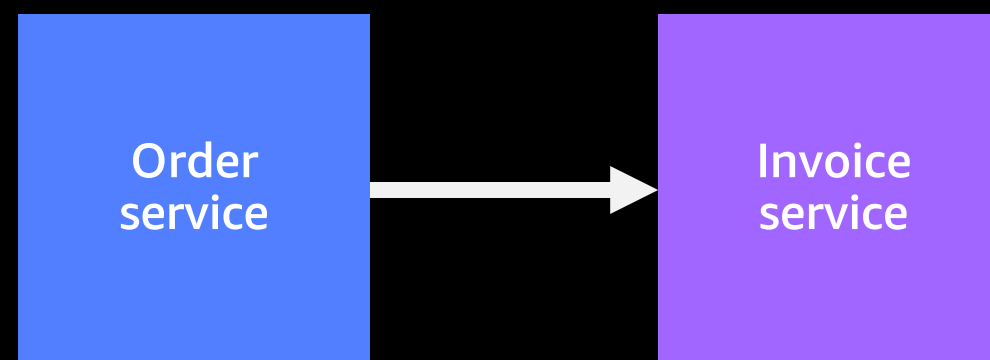
Coupling



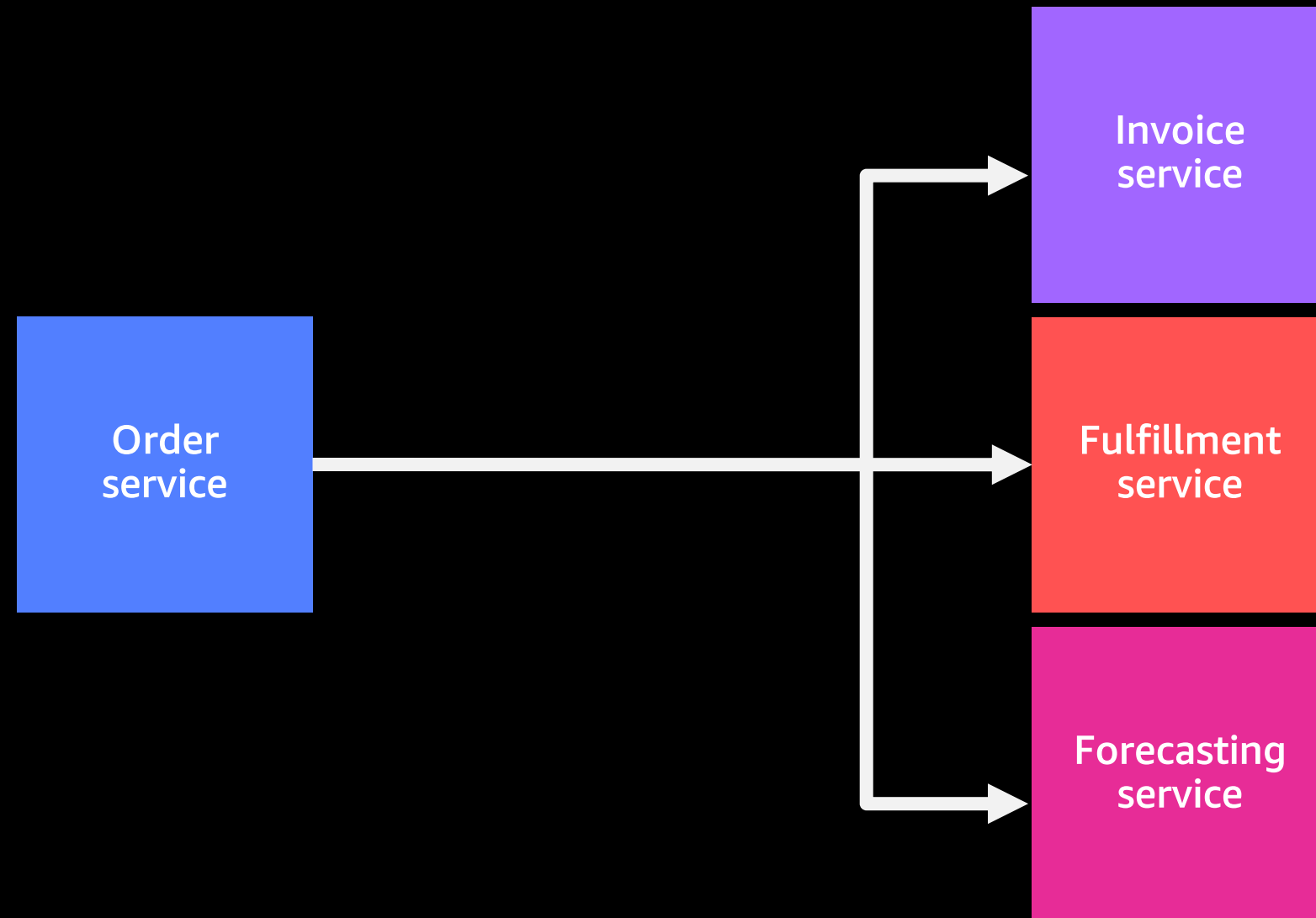
Synchronous APIs



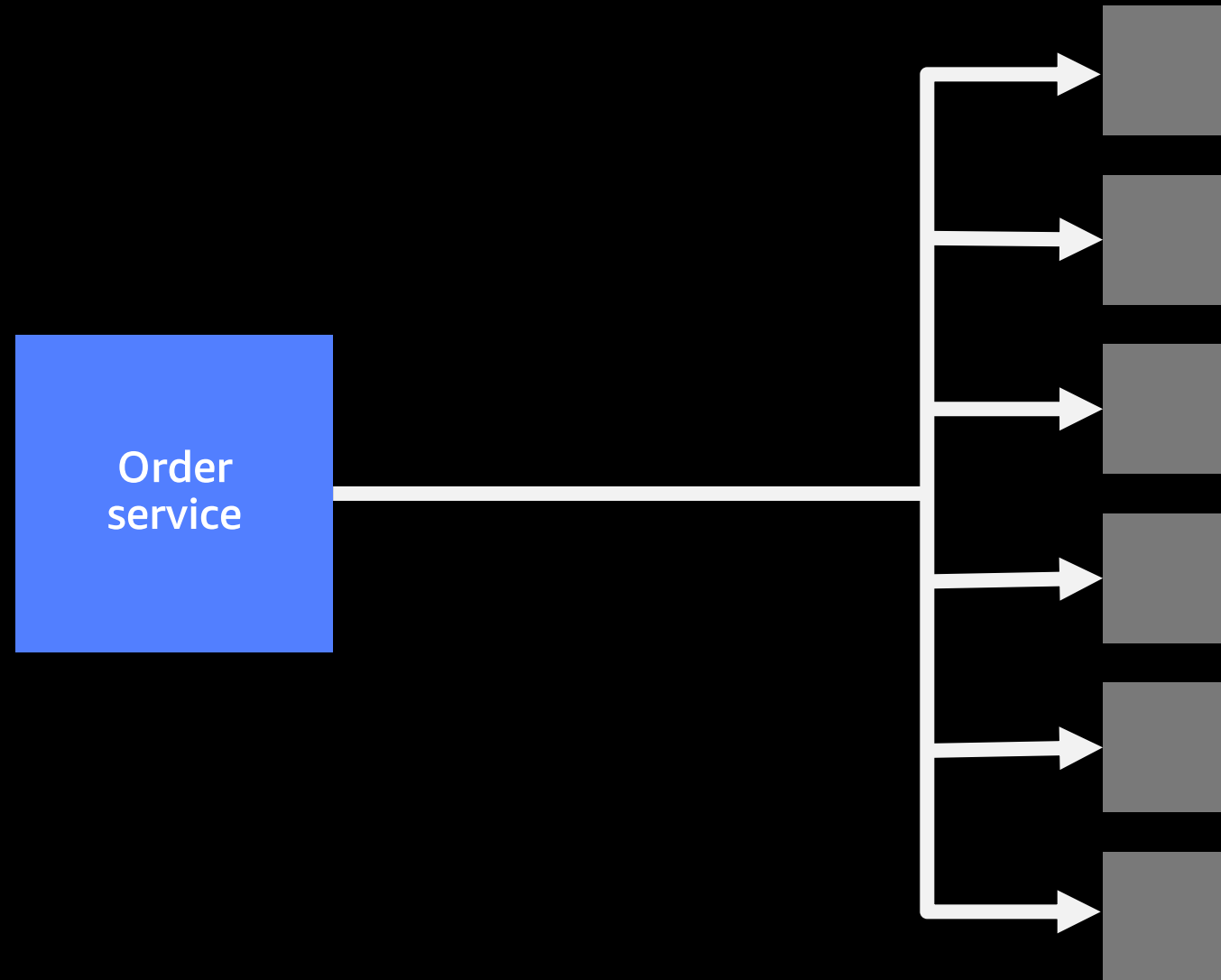
Architecture v1



Architecture v2



Long-term architecture



Events to the rescue

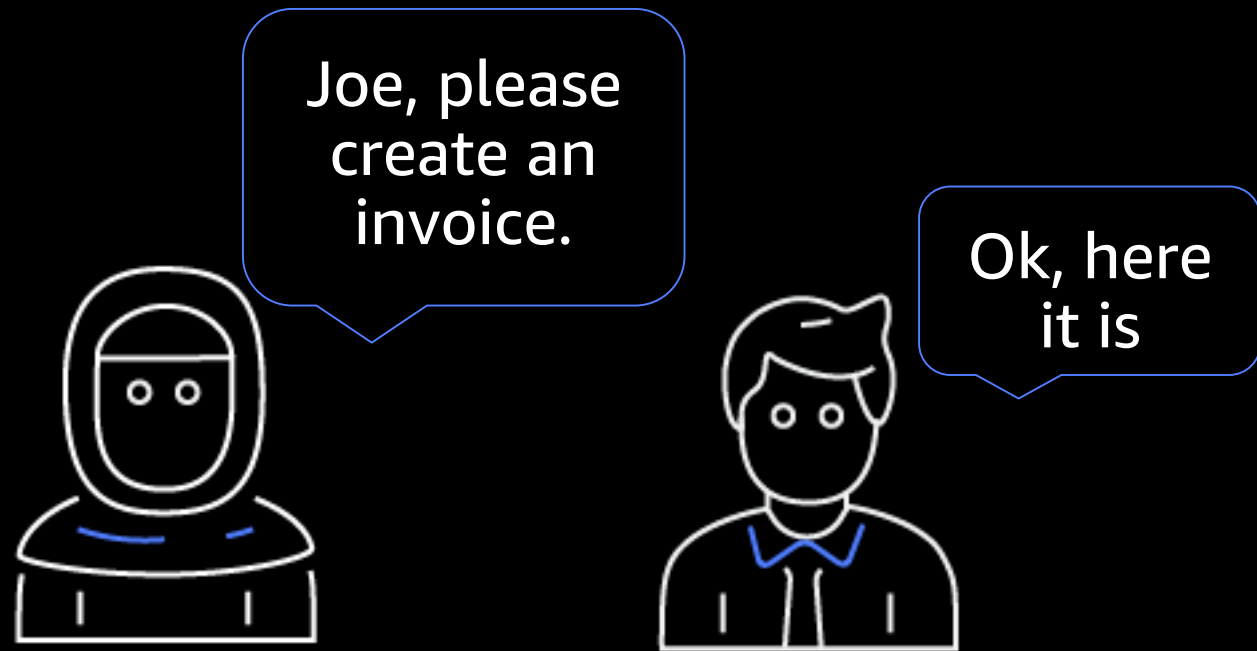


event

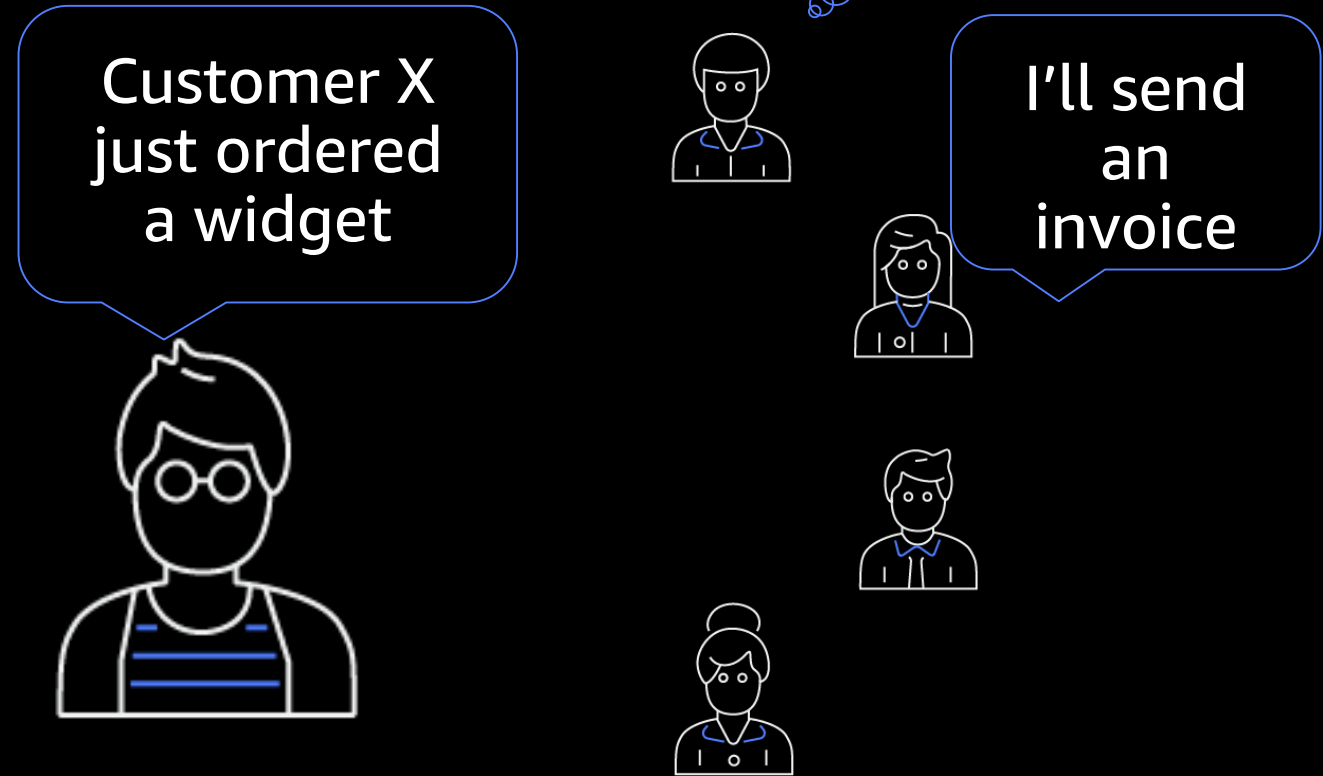
[i-'vent] **noun**

A signal that a system's state has changed.

Events are observable, not directed

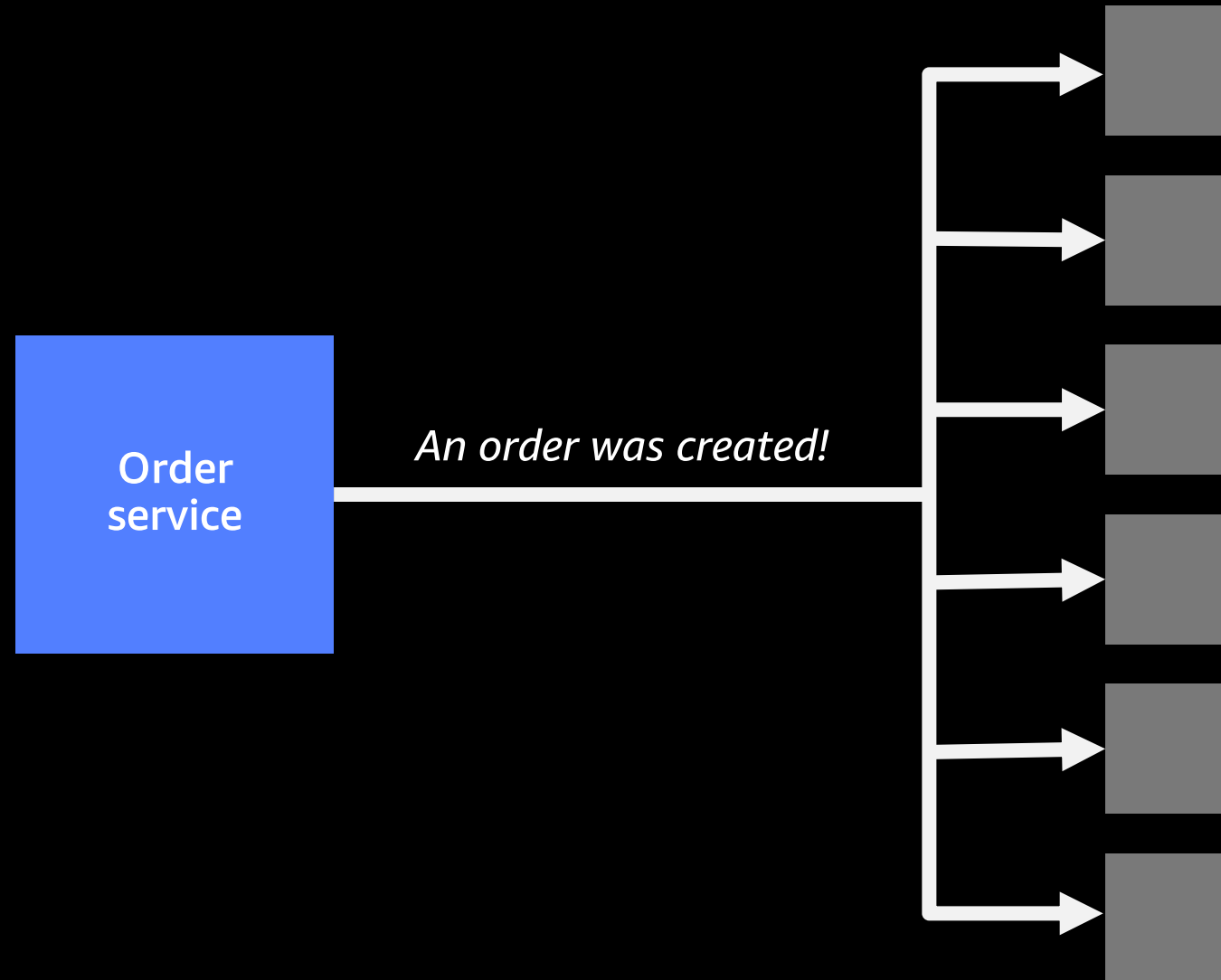


Directed
Commands

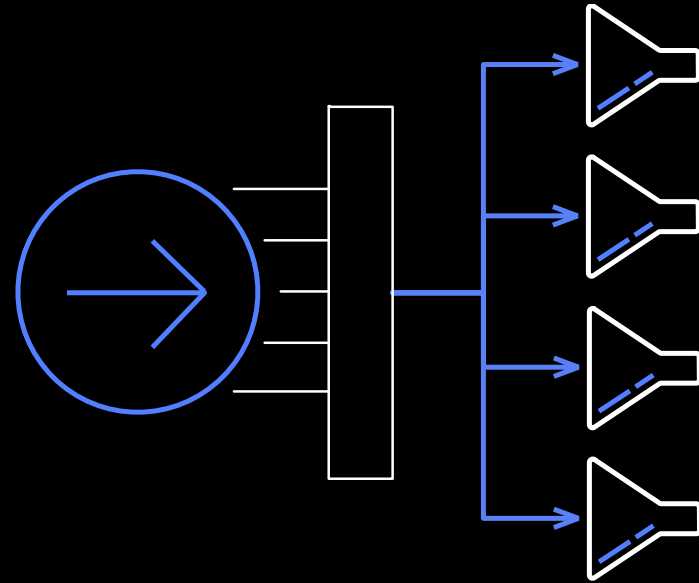


Observable
Events

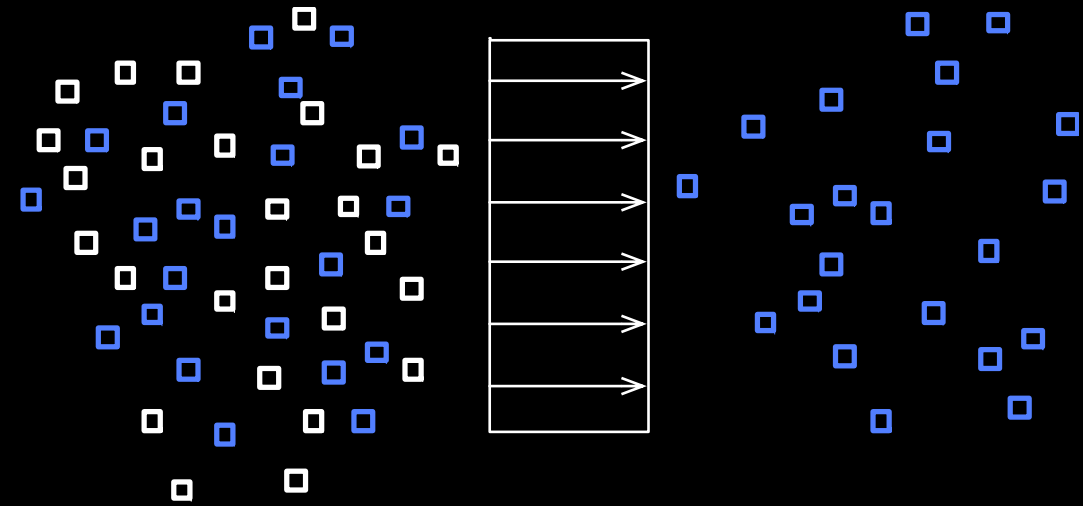
What do we do with these arrows?



Event routers



Abstracts producers
and consumers



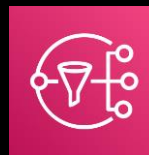
Selects and filters
events

Event routers in AWS

AWS Service

Characteristics

Topics



Amazon Simple
Notification Service

Lambda/SQS/HTTP targets
Nearly unlimited subscriptions
Filter on event metadata

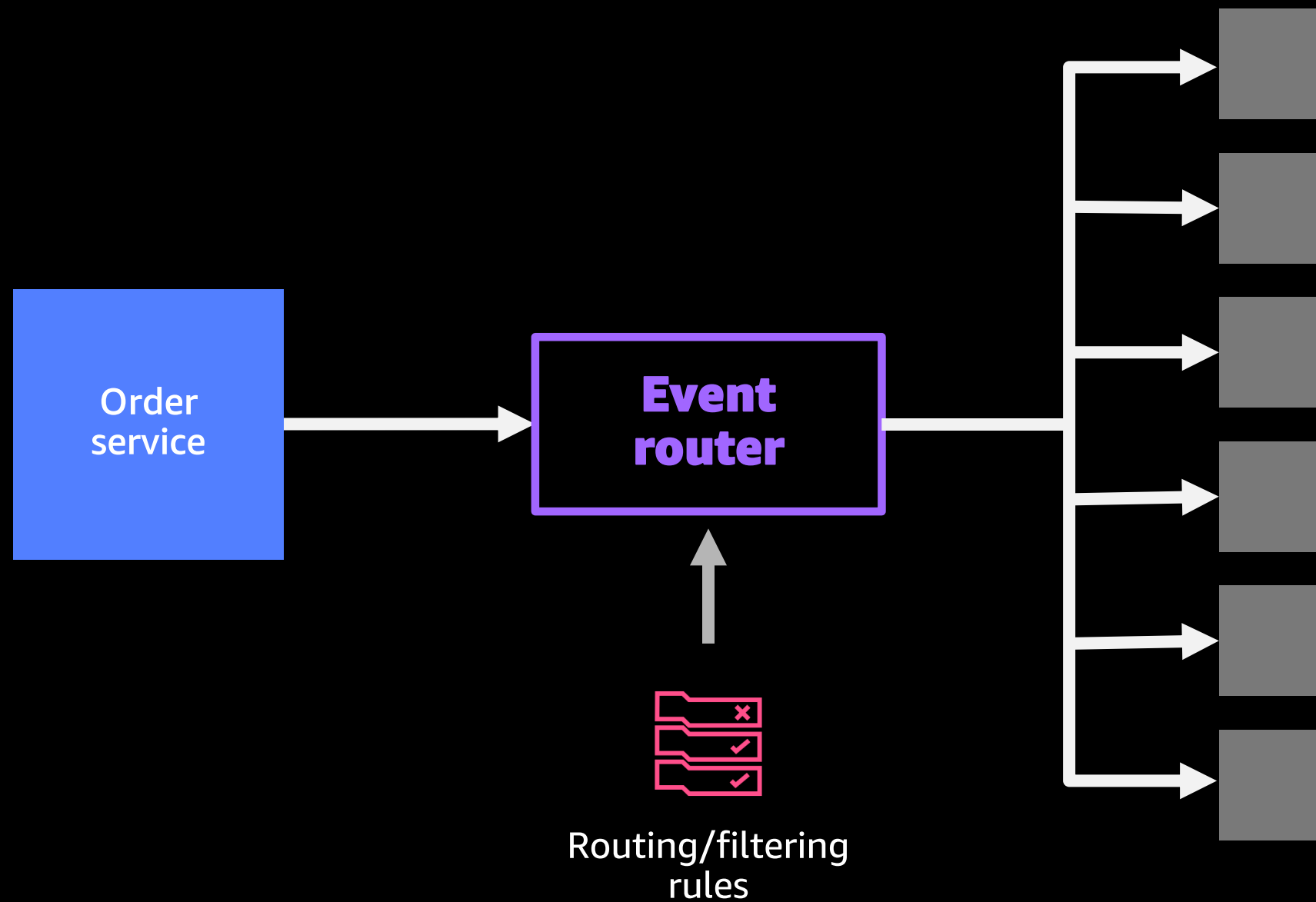
Event buses



Amazon EventBridge

19 AWS targets
Native SaaS event sources
Metadata and payload routing

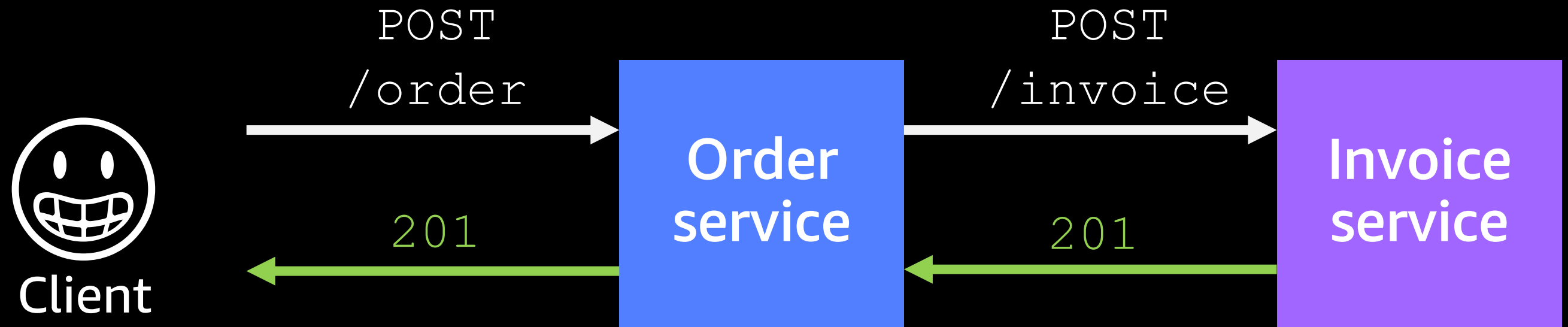
Event routers for fan out



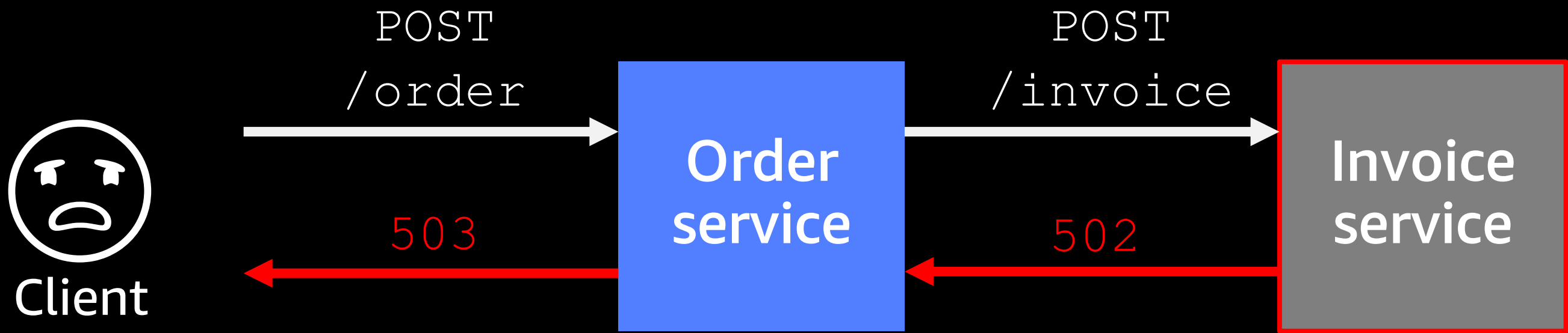
Availability



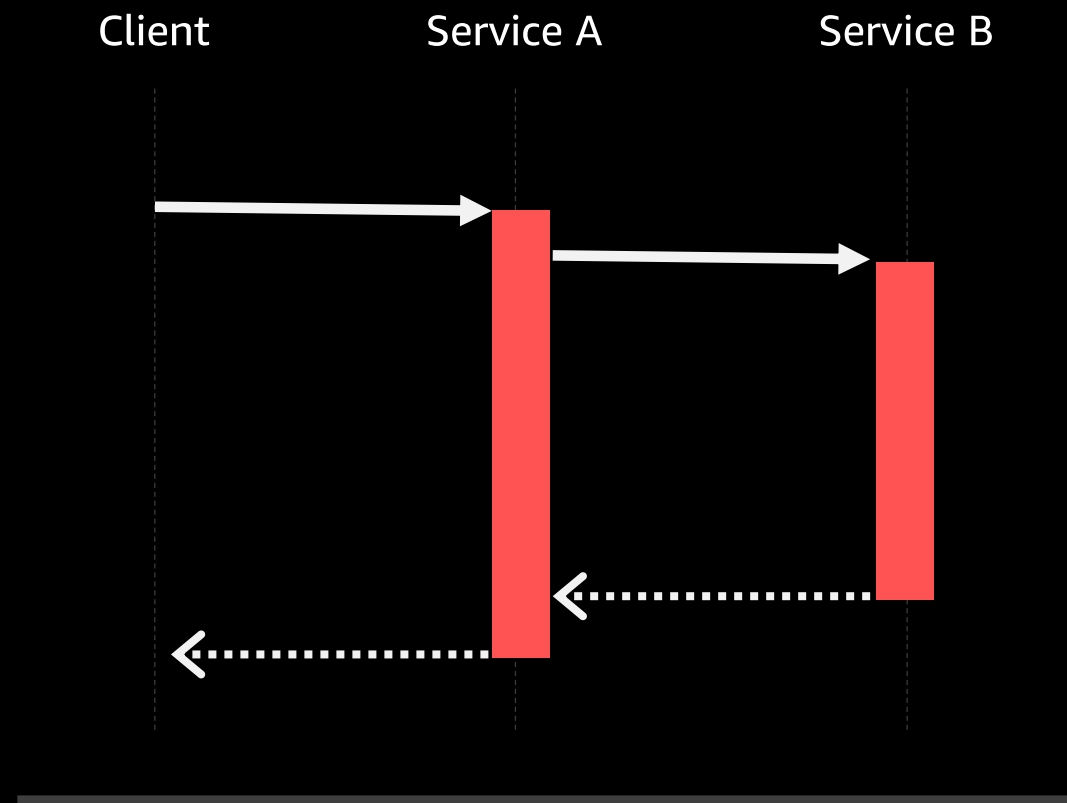
Synchronous APIs



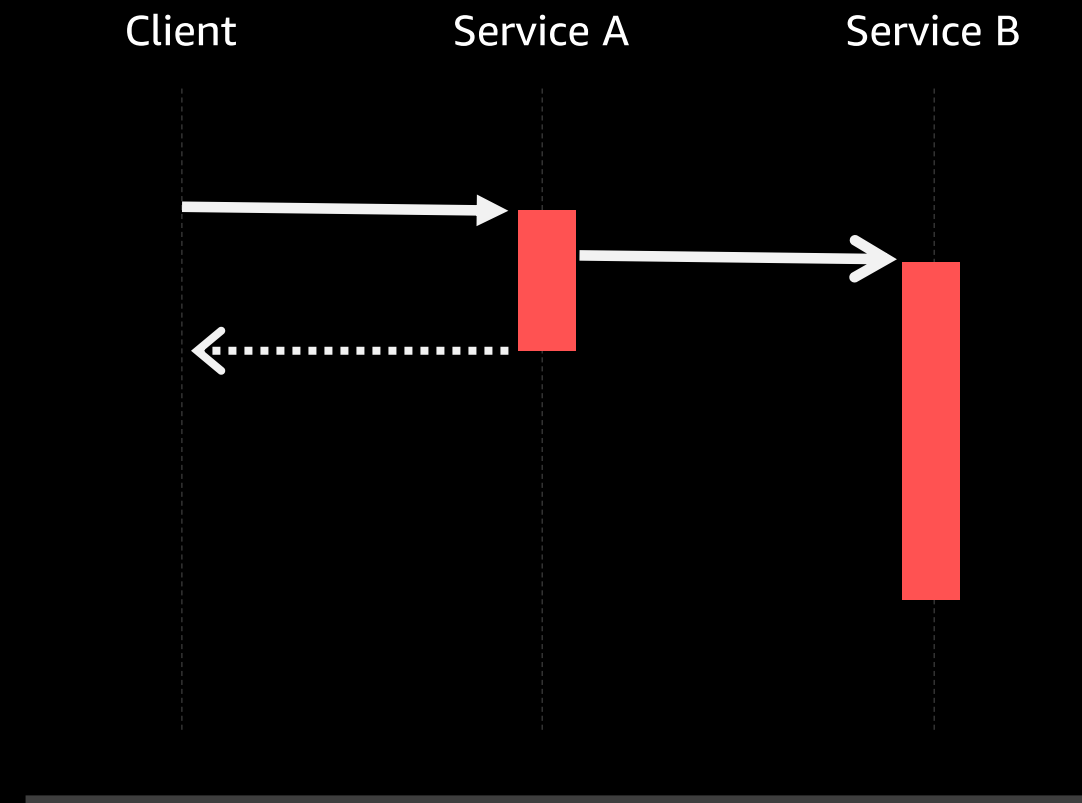
Downstream failures



Events are asynchronous

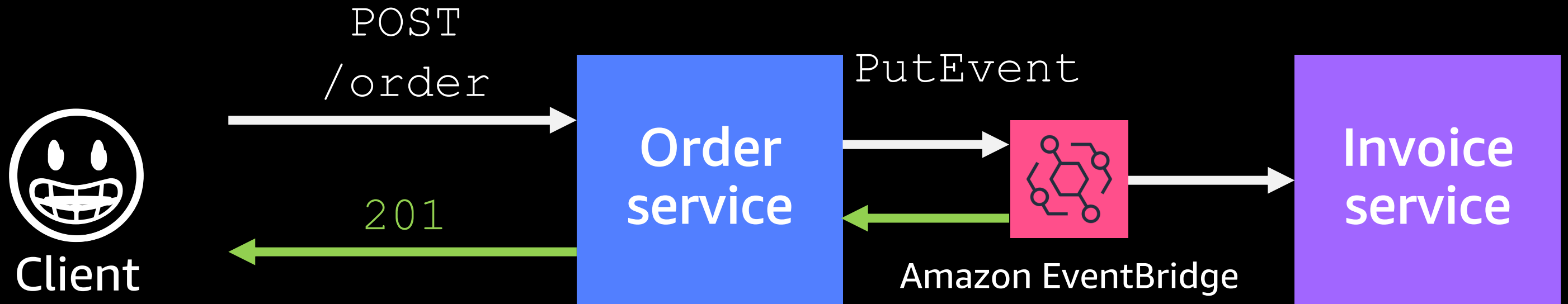


Synchronous
Commands

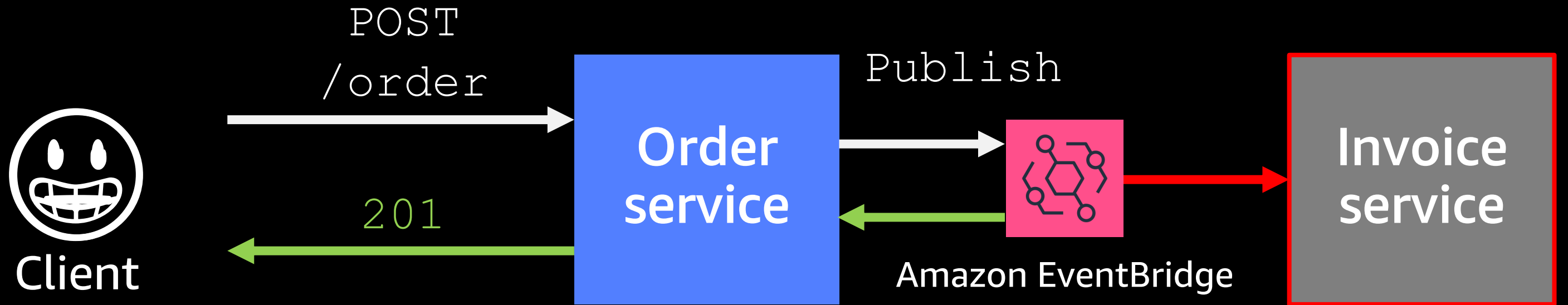


Asynchronous
Events

Asynchronous events

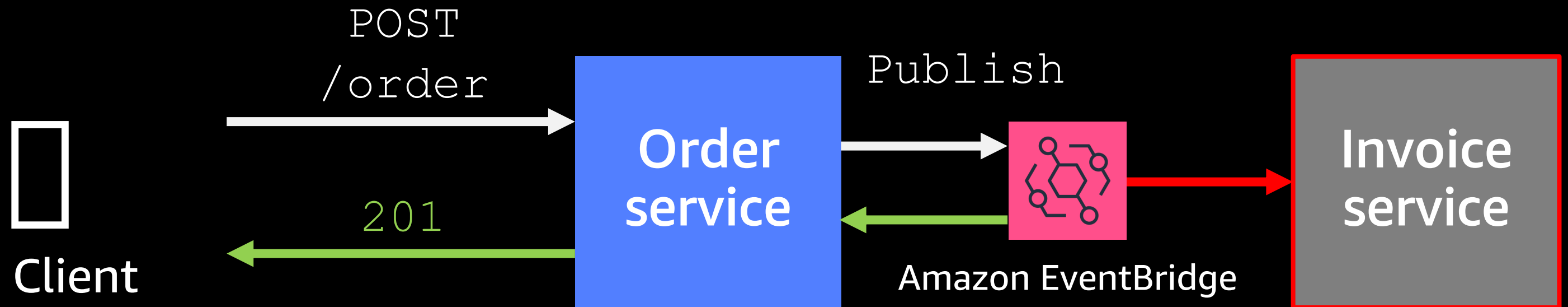


Asynchronous events

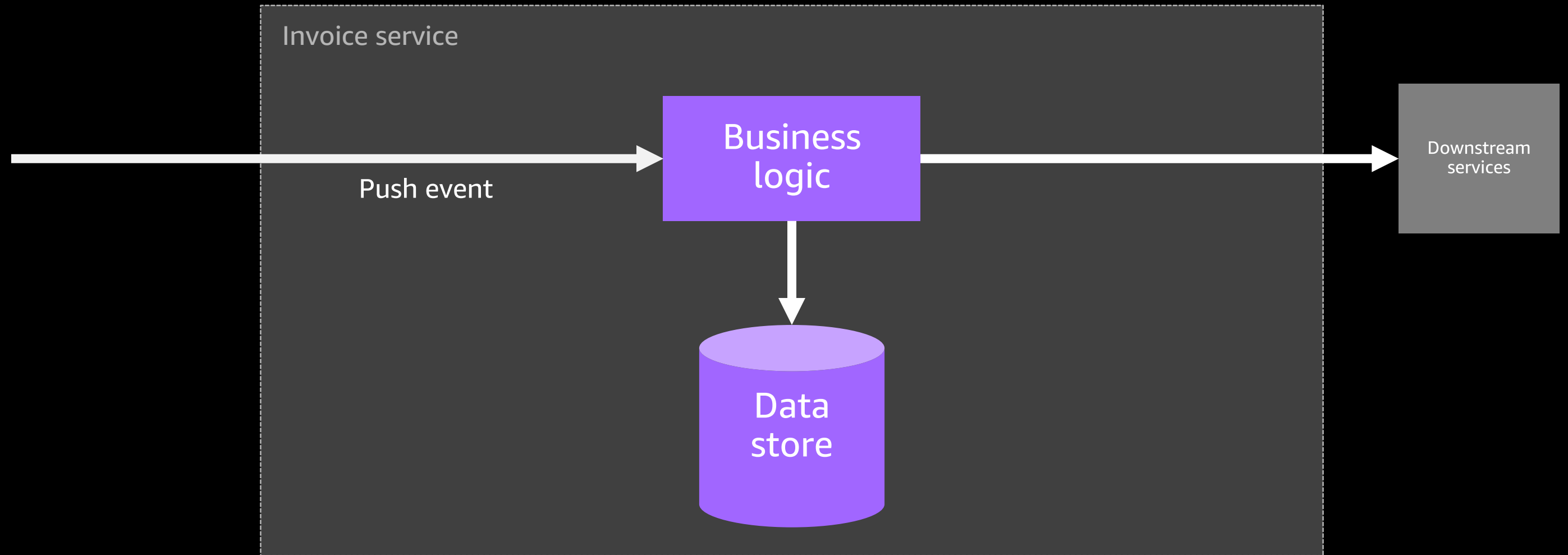


But wait, now the customer
won't get their invoice!

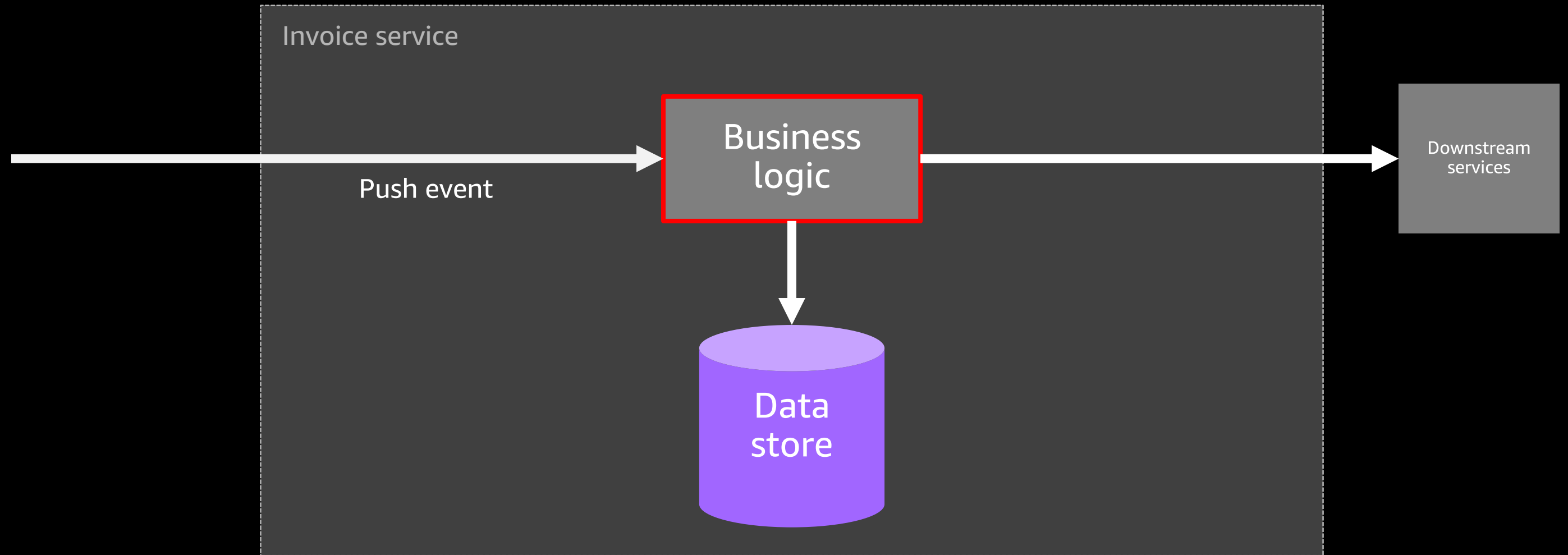
Event-driven resiliency



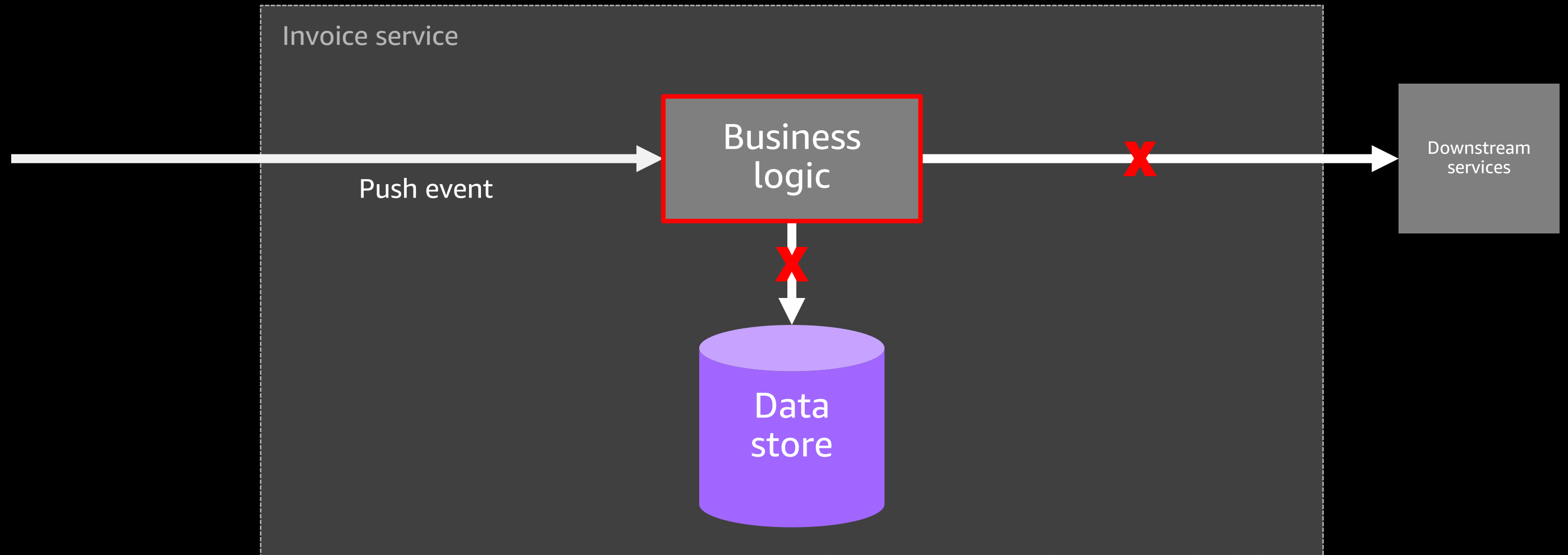
Inside the invoice service



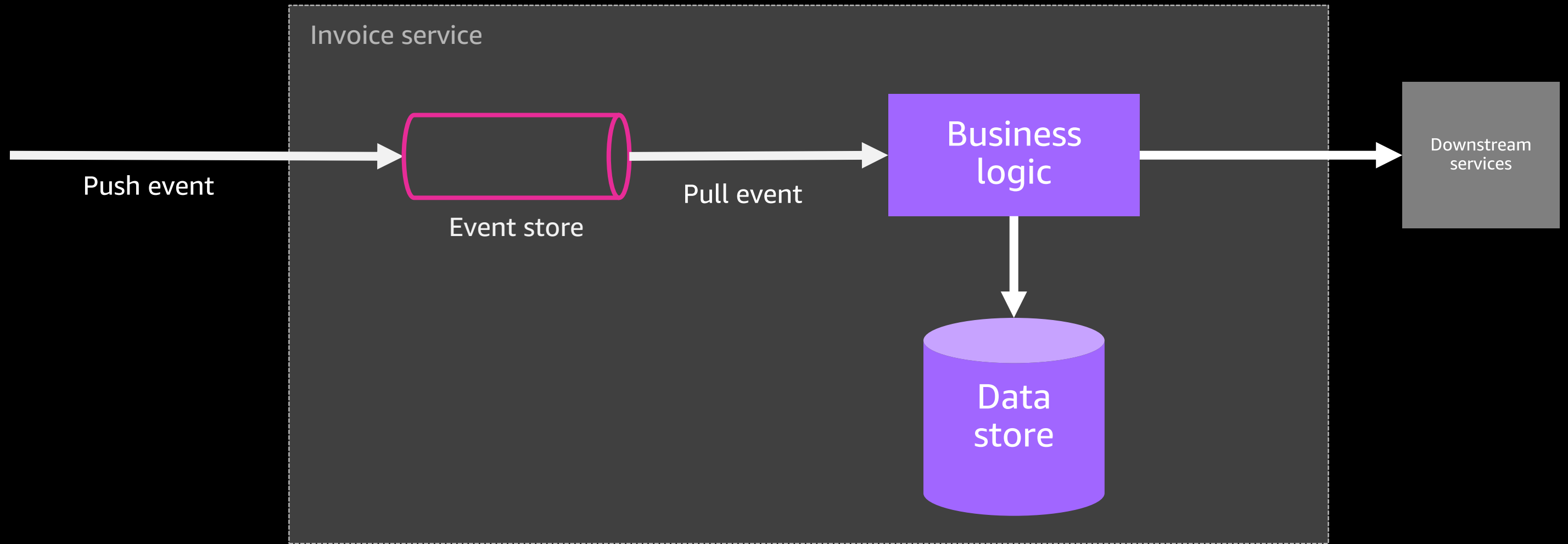
Inside the invoice service



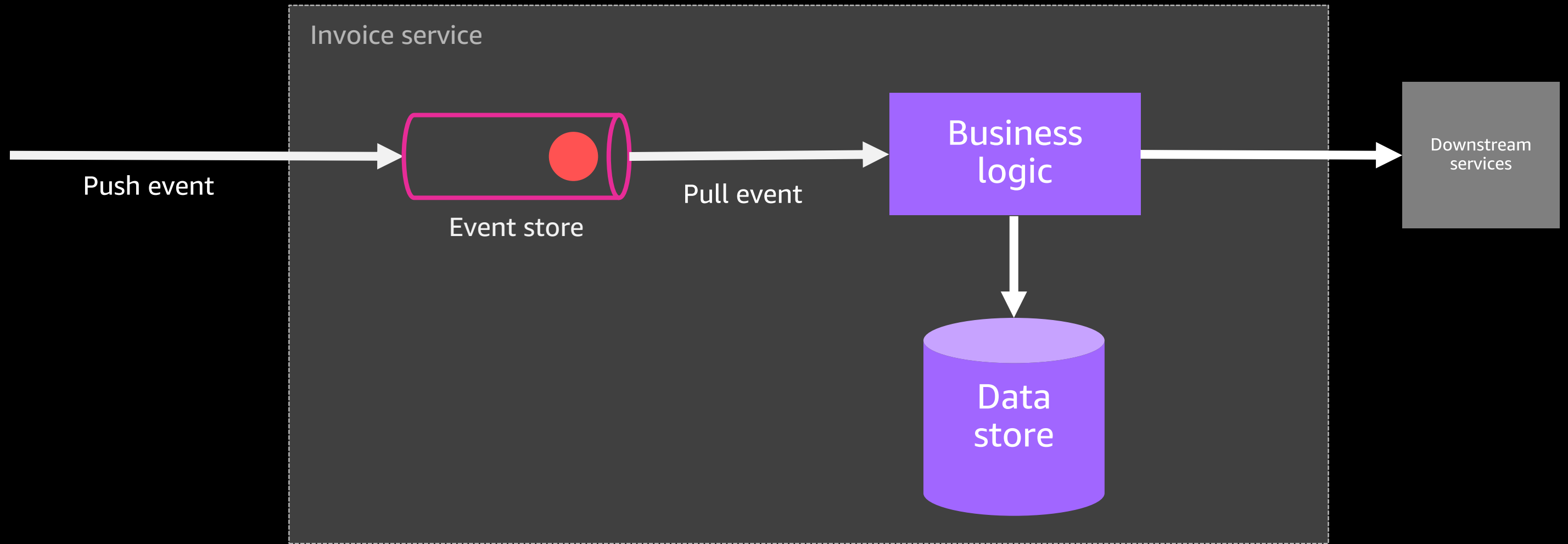
Inside the invoice service



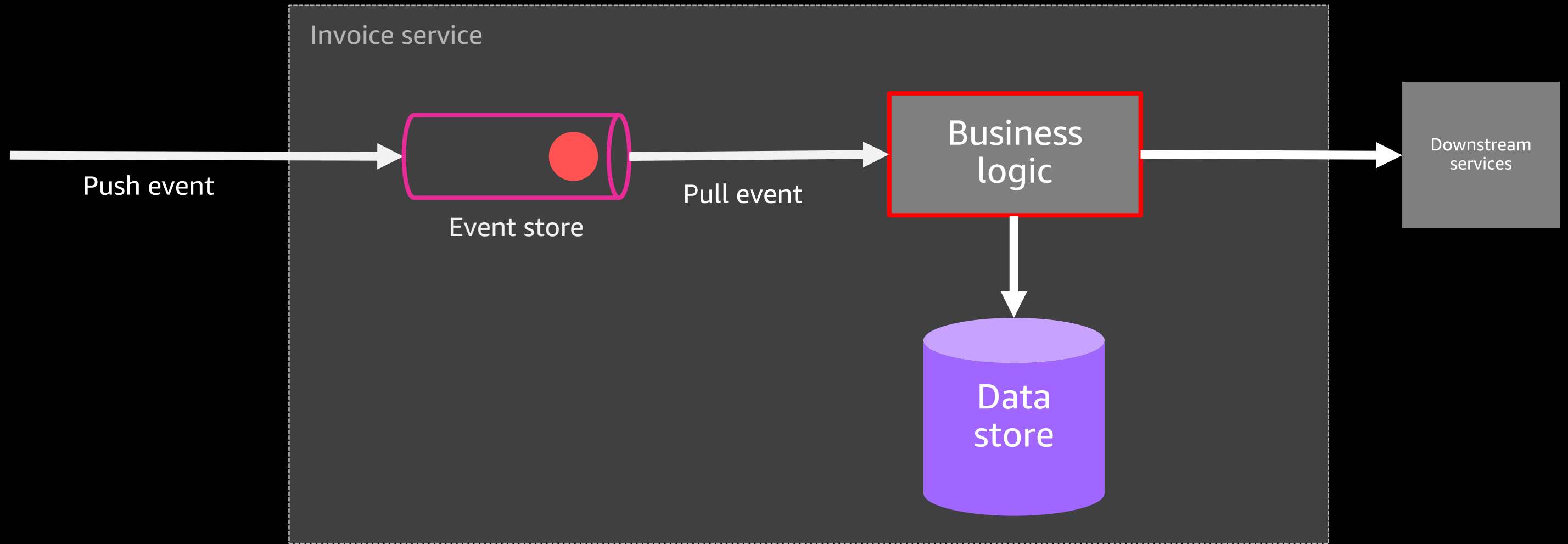
Inside the invoice service



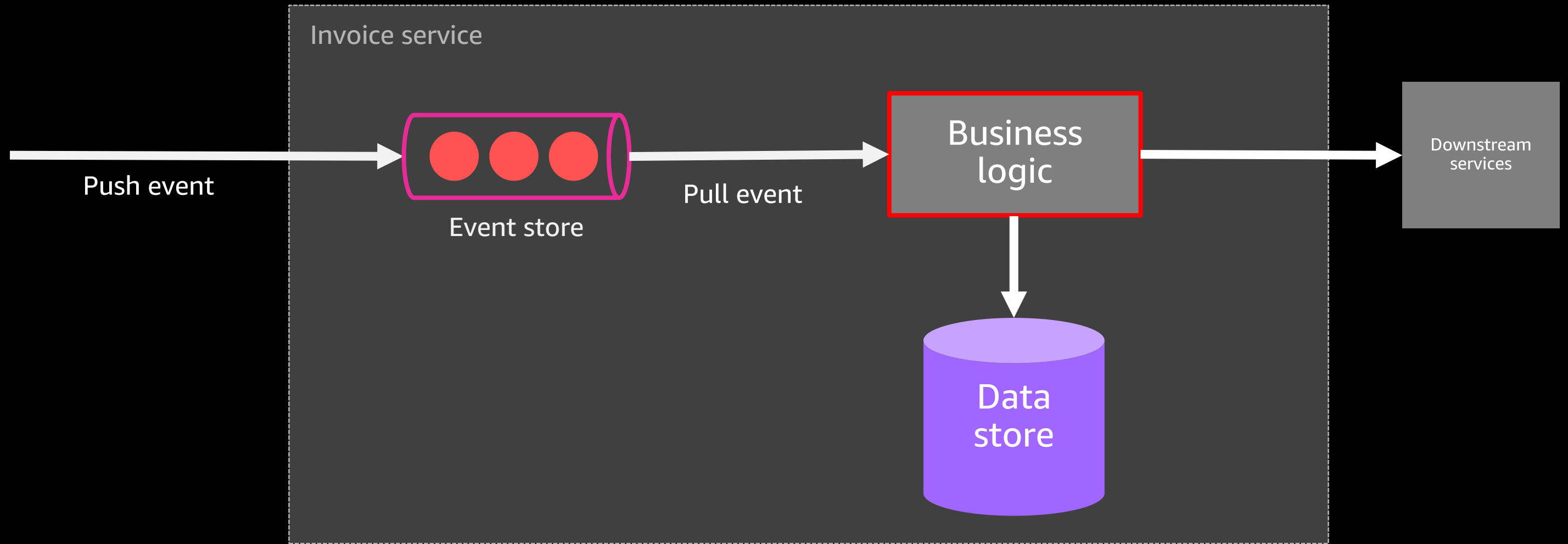
Inside the invoice service



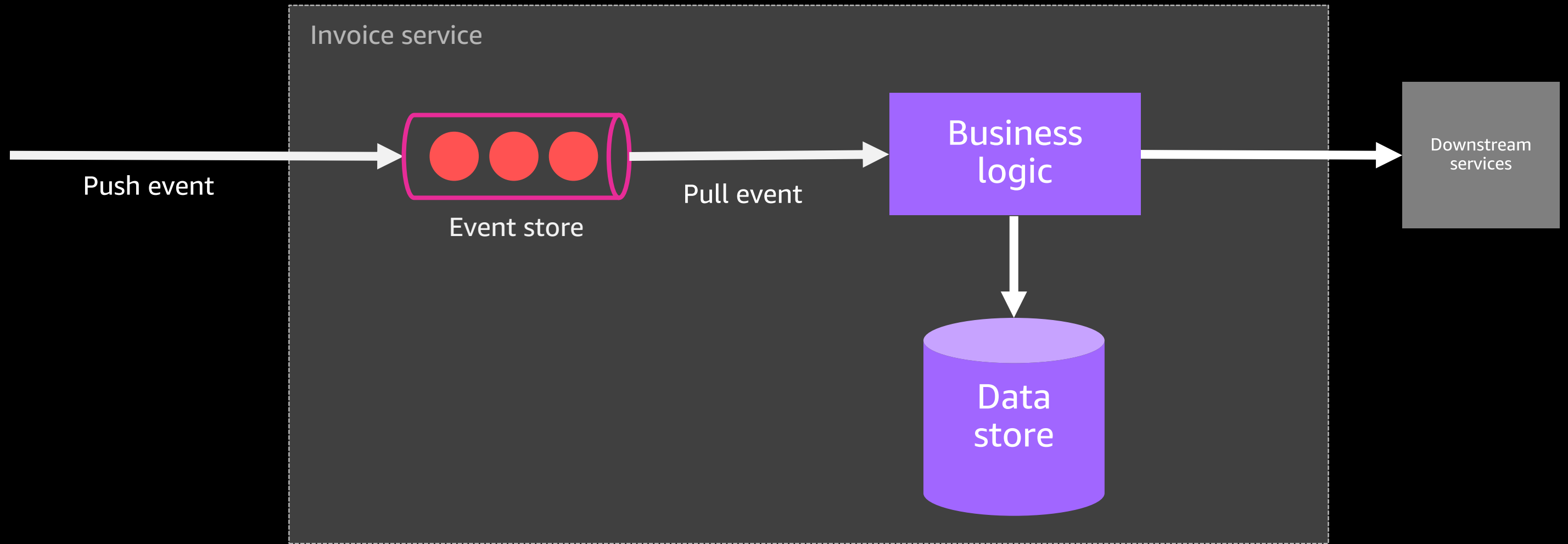
Inside the invoice service



Inside the invoice service



Inside the invoice service



Event stores in AWS

AWS Service

Characteristics

Queues



Amazon SQS

Single consumer
Optional FIFO
Fully managed scaling

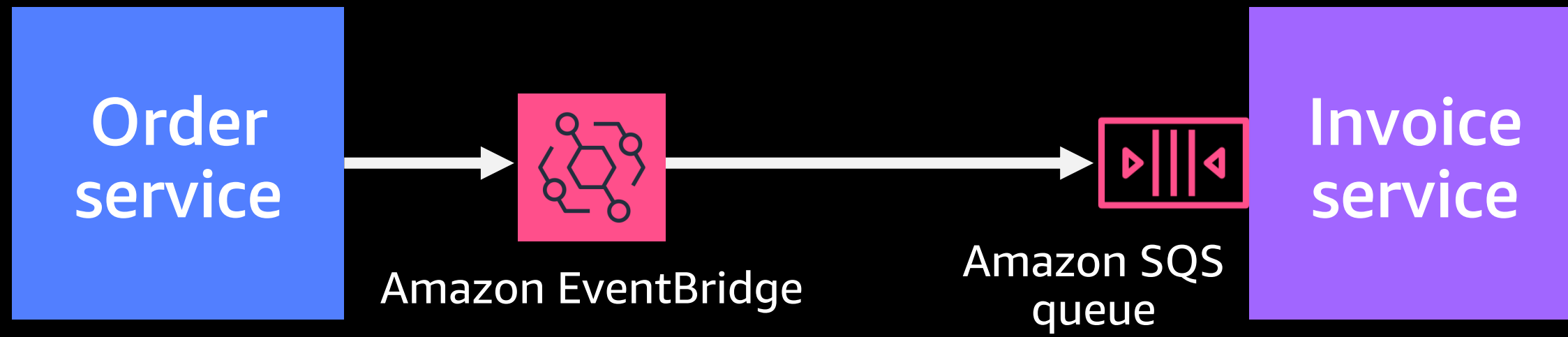
Streams



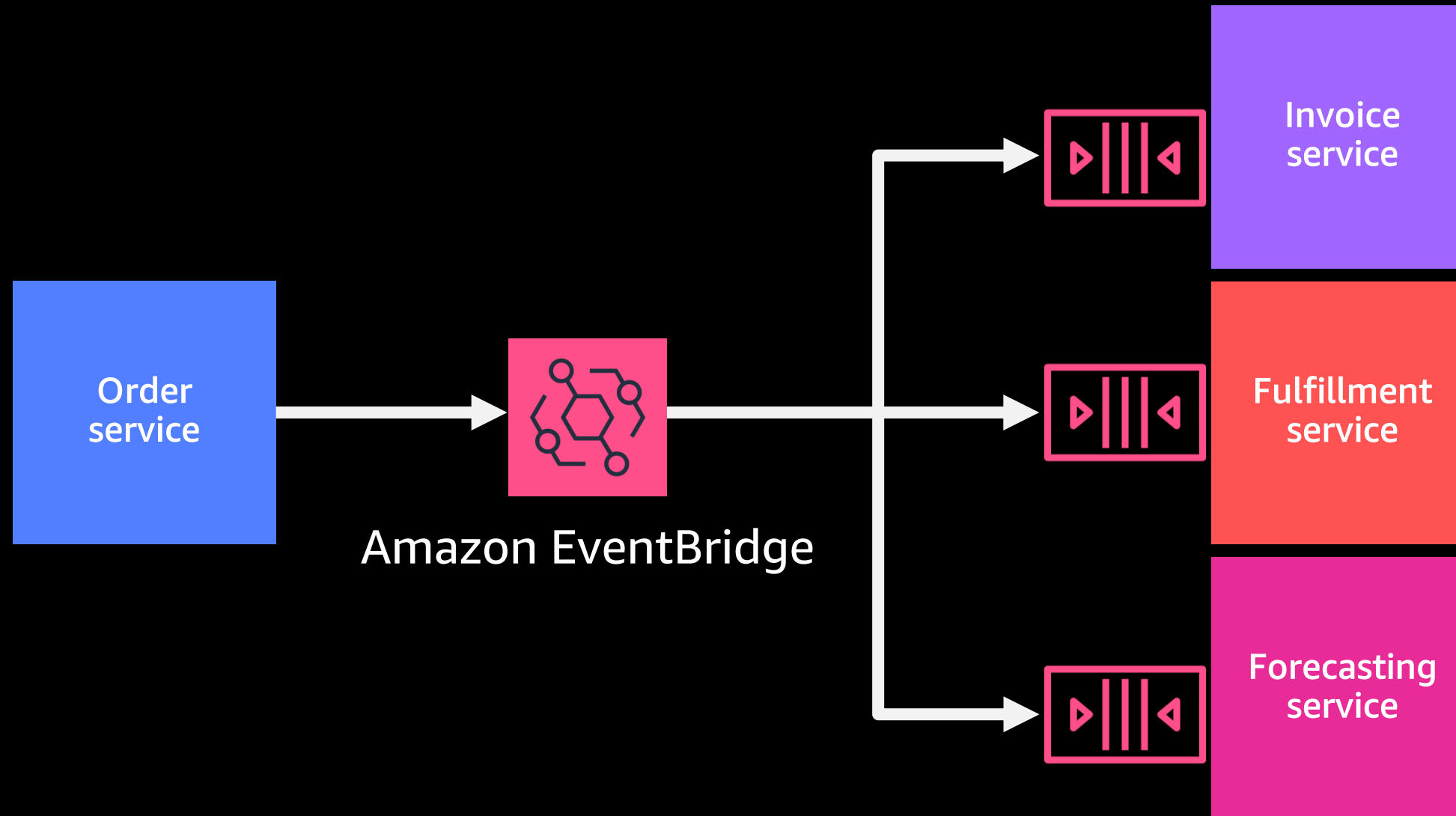
Amazon Kinesis Data
Streams

Multiple consumers
Well-ordered per shard
Shard-based scaling

Final architecture



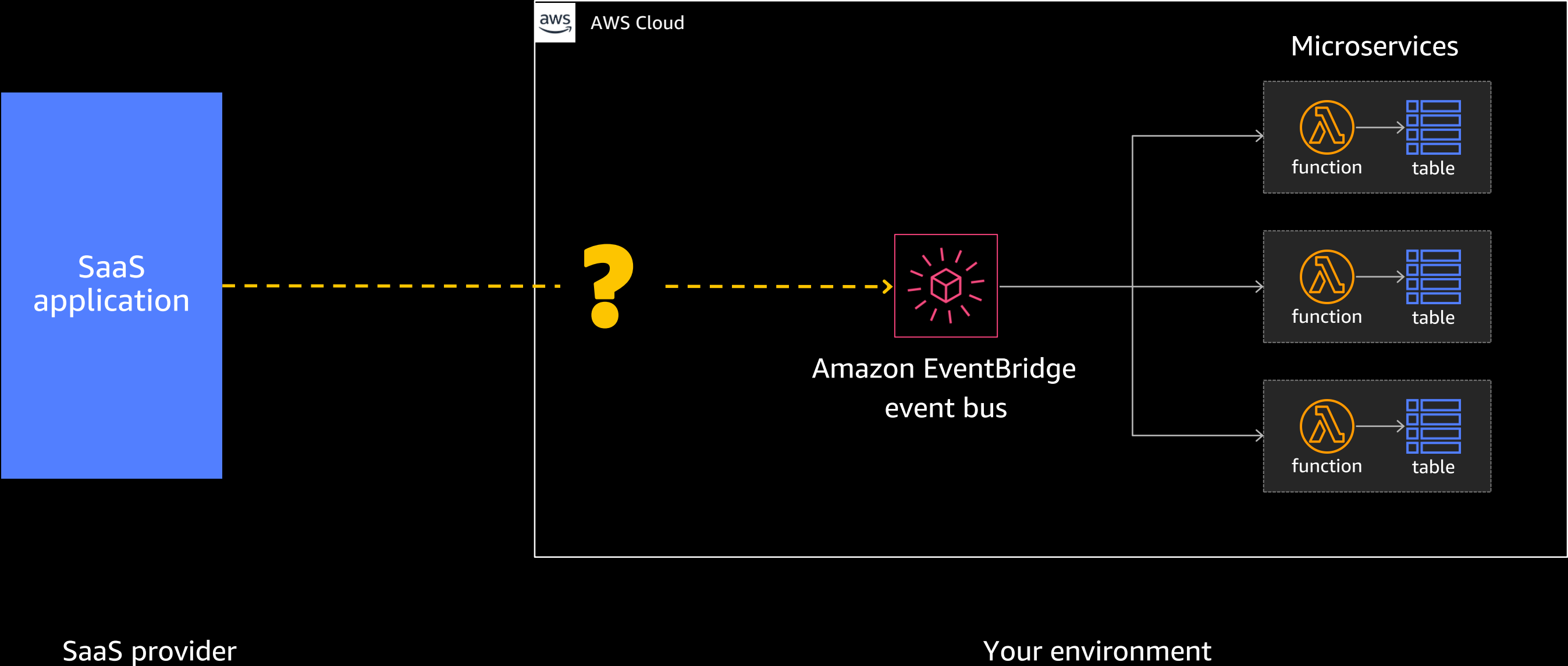
Event routing and storing



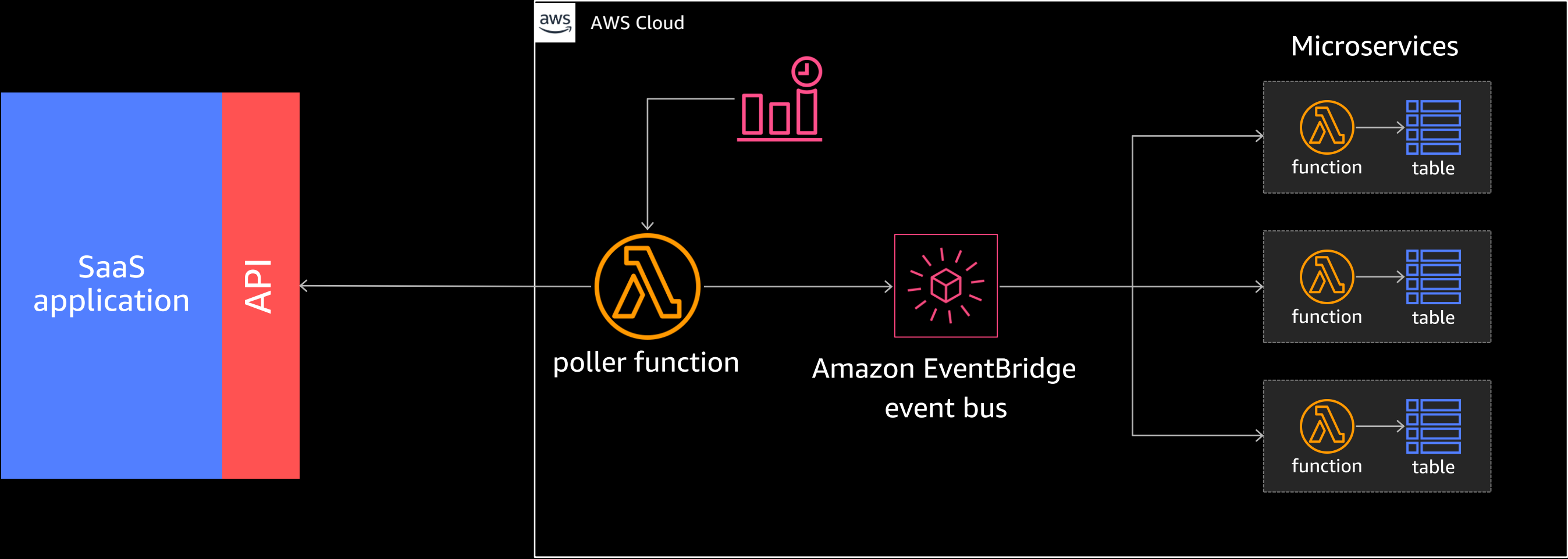
Integrating with SaaS applications



Bridging the gap



Polling

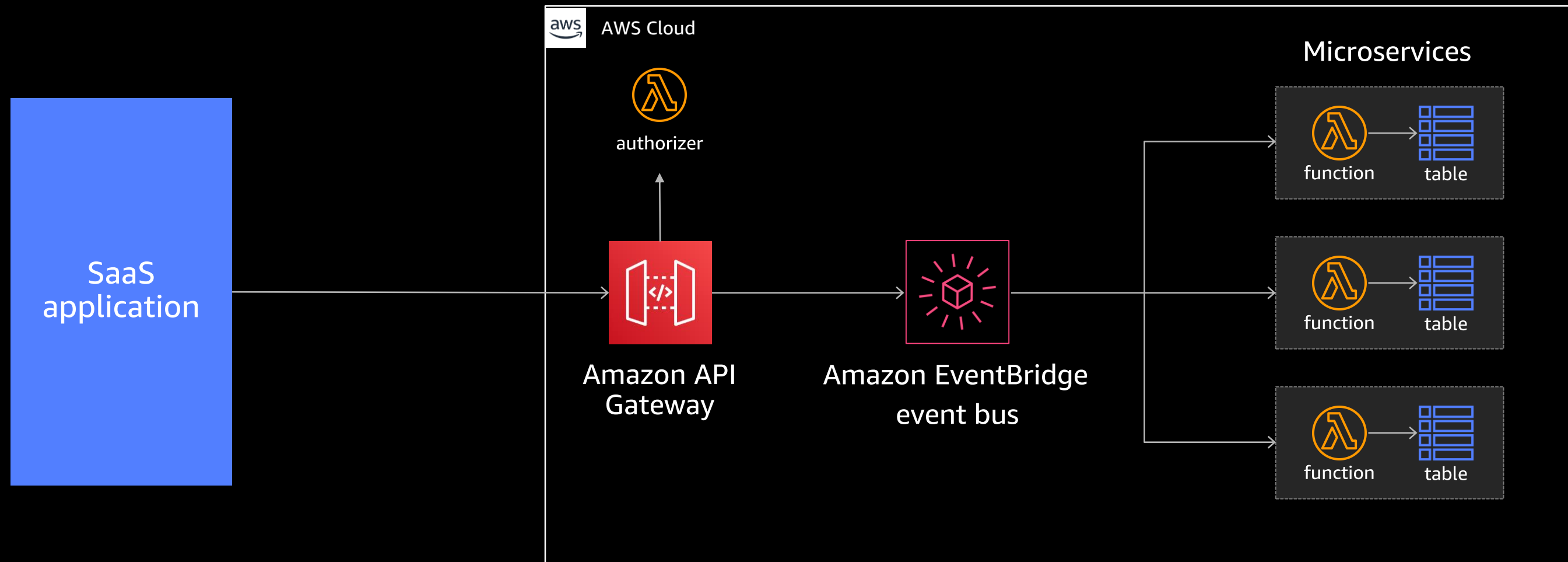


SaaS provider

Your environment



SaaS web hooks



SaaS provider

My environment



Is there a better option?





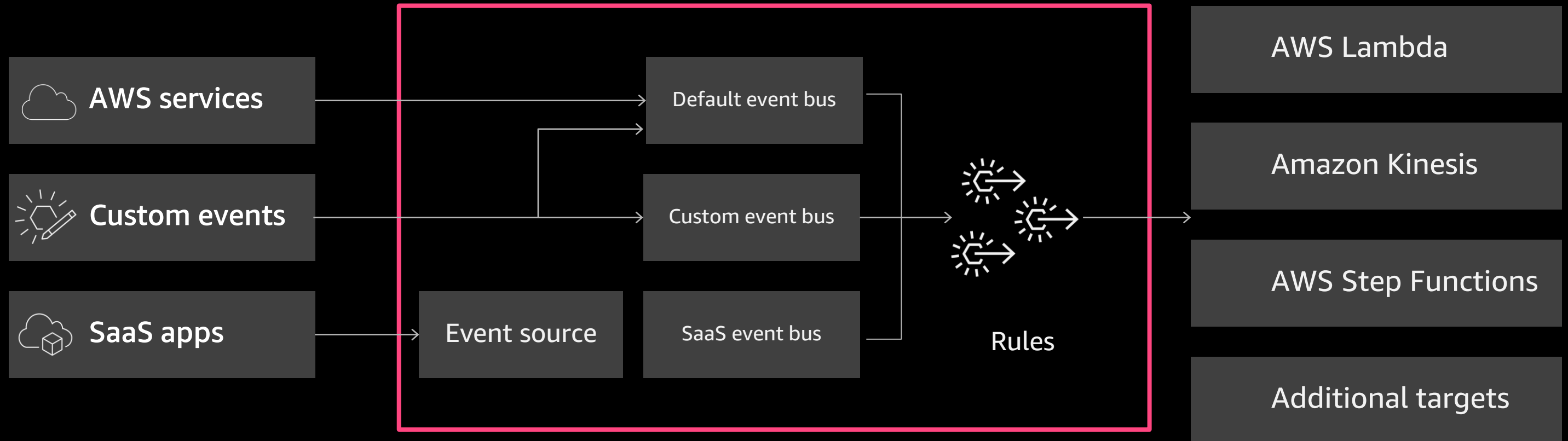
Amazon EventBridge

A serverless event bus service for SaaS and
AWS services

- Fully managed, pay-as-you-go
- Native integration with SaaS providers
- 90+ AWS services as sources
- 17 AWS services as targets
- \$1 per million events put into the bus
- No additional cost for delivery
- Easily build event-driven architectures



Amazon EventBridge



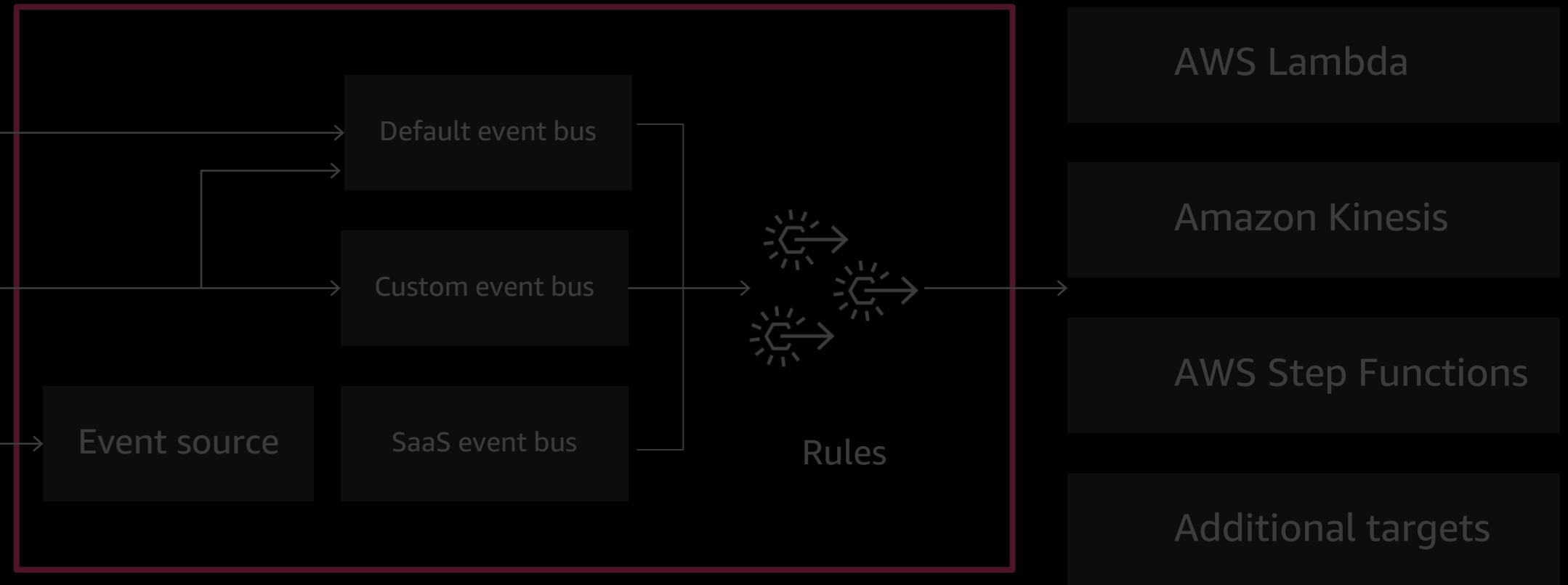
Amazon EventBridge

Event sources

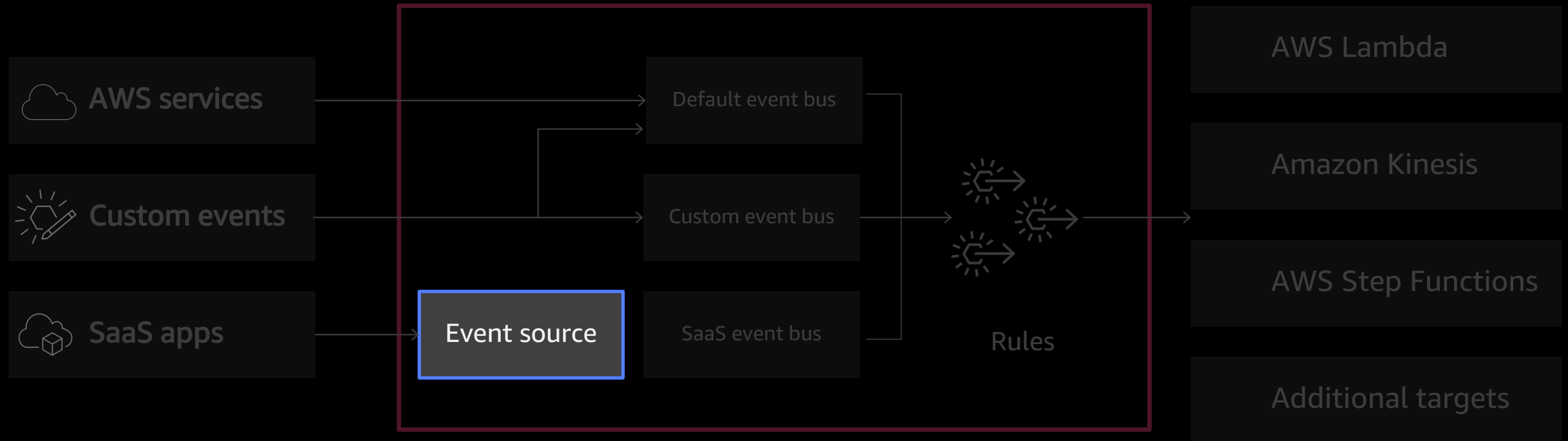
 AWS services

 Custom events

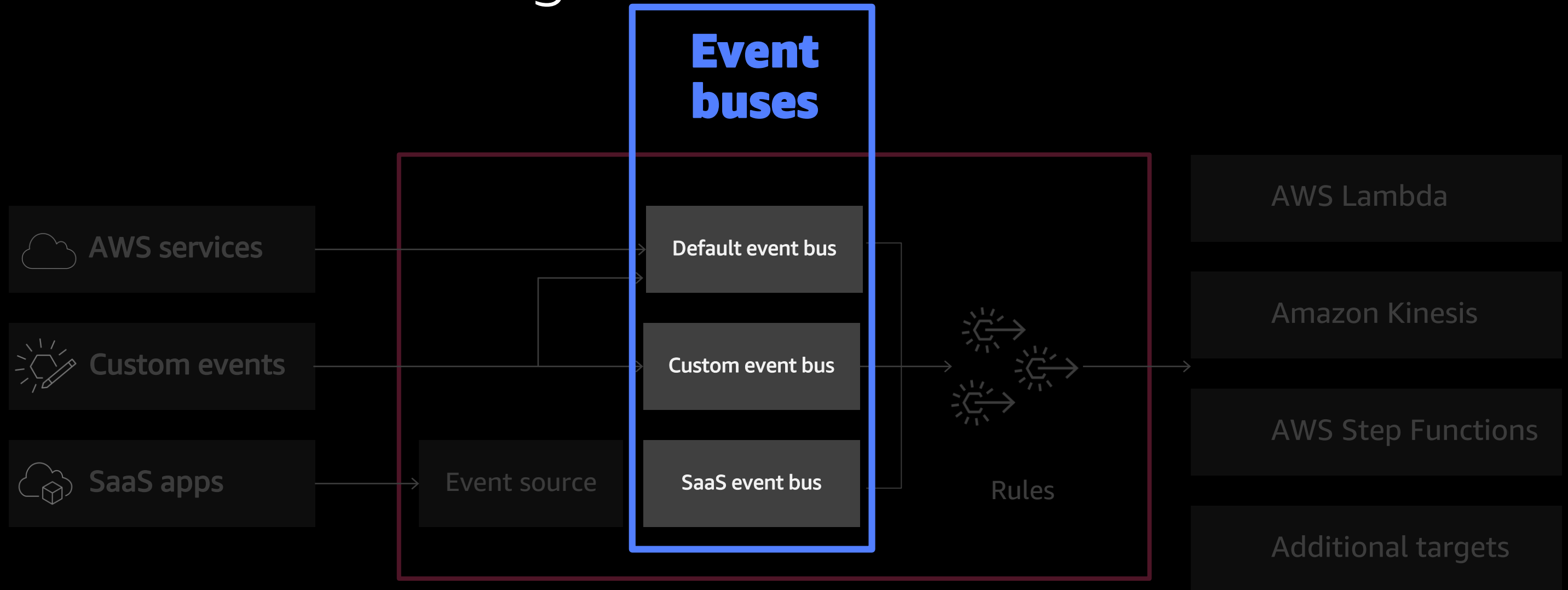
 SaaS apps



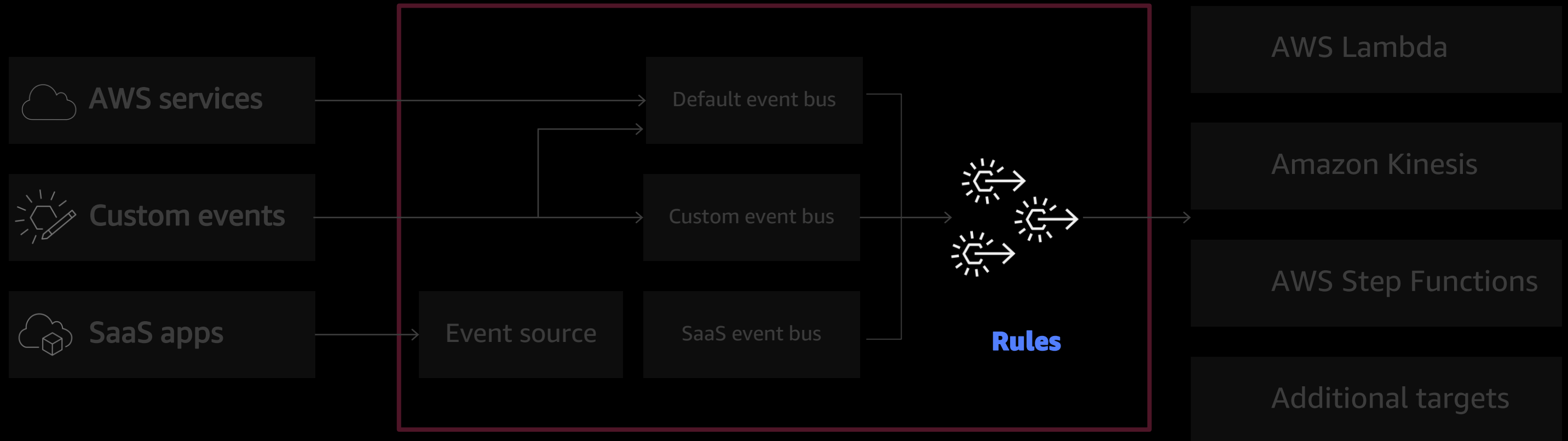
Amazon EventBridge



Amazon EventBridge



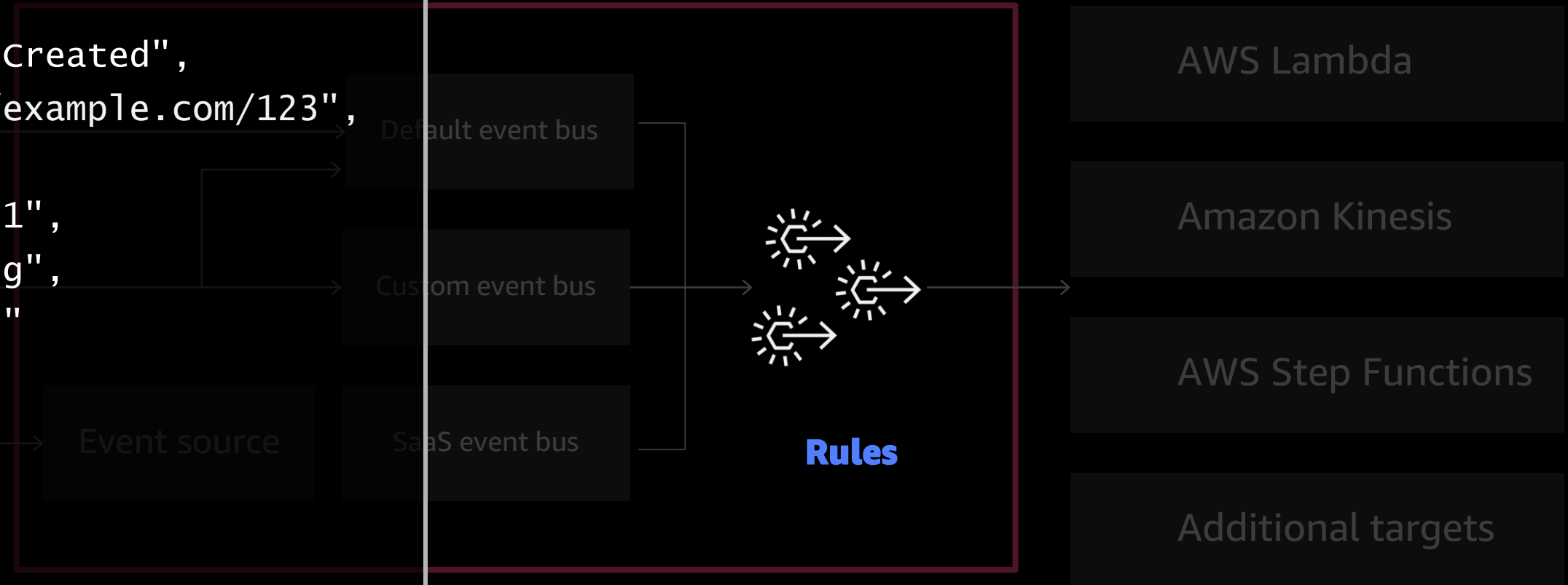
Amazon EventBridge



Amazon EventBridge

Example event:


```
{  
  "detail-type": "Ticket Created",  
  "source": "aws.partner/example.com/123",  
  "detail": {  
    "ticketId": "987654321",  
    "department": "billing",  
    "creator": "user12345"  
  },  
  ...  
}
```



Amazon EventBridge

Example event:

```
{  
  "detail-type": "Ticket Created",  
  "source": "aws.partner/example.com/123",  
  "detail": {  
    "ticketId": "987654321",  
    "department": "billing",  
    "creator": "user12345"  
  },  
  ...  
}
```



The diagram illustrates the event flow process. It starts with 'SaaS apps' (represented by a cloud icon) sending events to an 'Event source'. From the 'Event source', events pass through a 'SaaS event bus', then a 'Custom event bus', and finally reach the 'Default event bus'. The 'Event source' and 'SaaS event bus' are highlighted with a red box.

Example rule:

```
{  
  "source": ["aws.partner/example.com/123"]  
}
```



The diagram shows a 'Rules' box containing three gear icons. An arrow from the 'Event source' in the previous diagram points to the 'Rules' box. From the 'Rules' box, arrows point to four target categories: 'Amazon Kinesis', 'AWS Step Functions', and 'Additional targets'. The 'Rules' box is highlighted with a red box.

Amazon EventBridge

Example event:

```
{  
  "detail-type": "Ticket Created",  
  "source": "aws.partner/example.com/123",  
  "detail": {  
    "ticketId": "987654321",  
    "department": "billing",  
    "creator": "user12345"  
  }  
  ...  
}
```

The diagram illustrates the event flow process. On the left, 'SaaS apps' (represented by a cloud icon) send events to an 'Event source'. From the 'Event source', the event is sent to a 'SaaS event bus'. From there, it can be sent to a 'Custom event bus' or a 'Default event bus'. The 'Custom event bus' is associated with 'Custom events' (represented by a gear icon), and the 'Default event bus' is associated with 'AWS services' (represented by a gear icon).

Example rule:

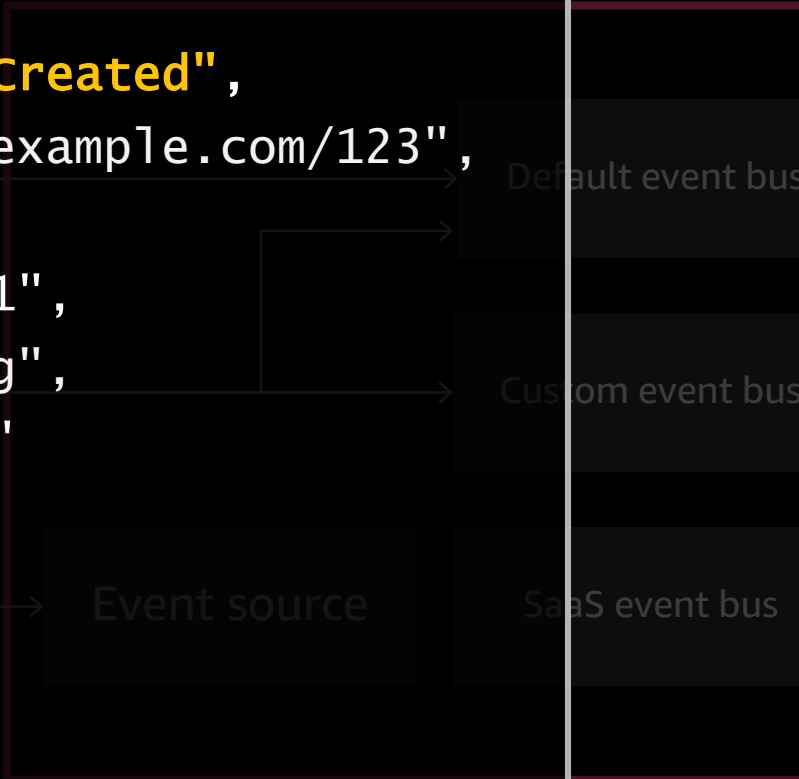
```
{  
  "detail": {  
    "department": ["billing", "fulfillment"]  
  }  
}
```

The diagram shows the rule configuration and its targets. The rule is defined with a 'detail' object containing a 'department' array with values 'billing' and 'fulfillment'. The rule is represented by a gear icon with an arrow. The rule triggers events to four targets: 'AWS Lambda', 'Amazon Kinesis', 'AWS Step Functions', and 'Additional targets'.

Amazon EventBridge

Example event:

```
{  
  "detail-type": "Ticket Created",  
  "source": "aws.partner/example.com/123",  
  "detail": {  
    "ticketId": "987654321",  
    "department": "billing",  
    "creator": "user12345"  
  },  
  ...  
}
```



The diagram illustrates the event flow process. It starts with 'SaaS apps' (represented by a cloud icon) sending events to an 'Event source'. From the 'Event source', events pass through a 'SaaS event bus', then a 'Custom event bus', and finally reach the 'Default event bus'. The text 'AWS services' and 'Custom events' is overlaid on the diagram.

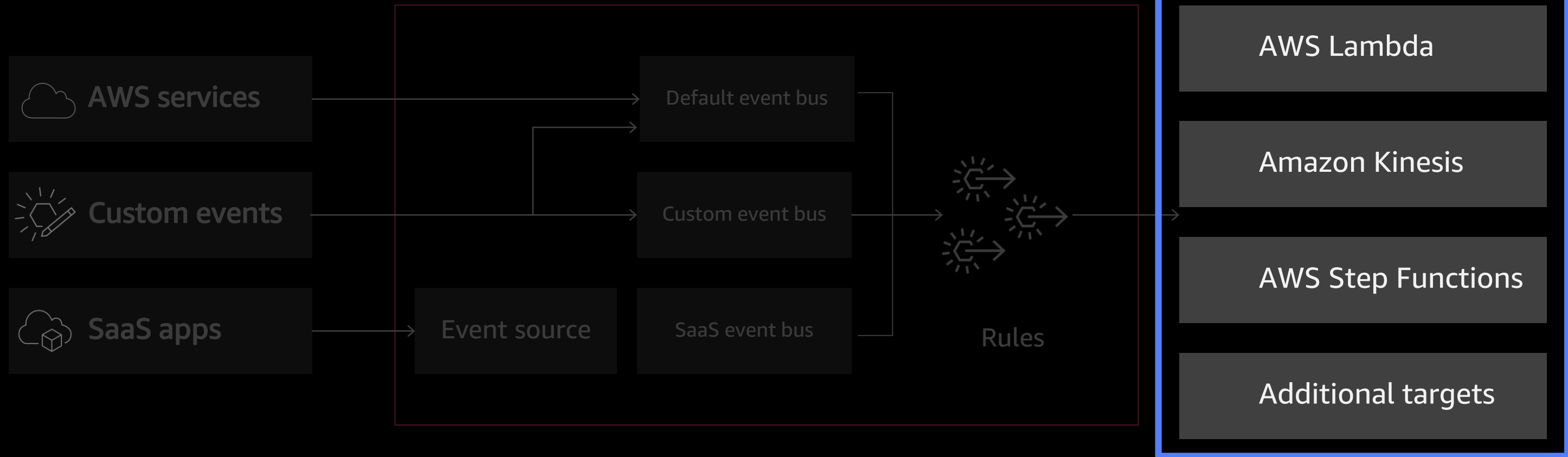
Example rule:

```
{  
  "detail-type": ["Ticket Resolved"]  
}
```



The diagram shows a 'Rules' box containing three gear icons. An arrow points from the 'Rules' box to a vertical stack of target boxes: 'Amazon Kinesis', 'AWS Step Functions', and 'Additional targets'. The text 'AWS services' is overlaid on the diagram.

Amazon EventBridge



EventBridge Integration Partners

onelogin

pagerduty

SAVIYNT



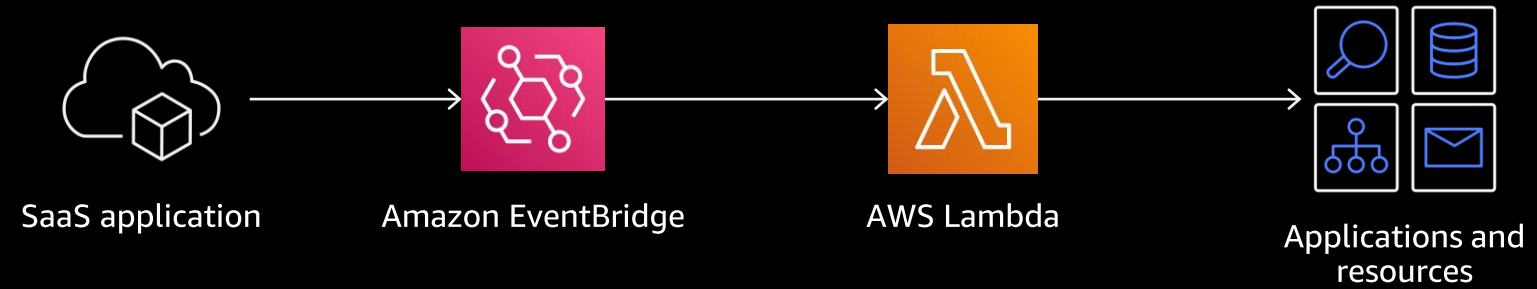
whispir



SignalFx

Common use cases

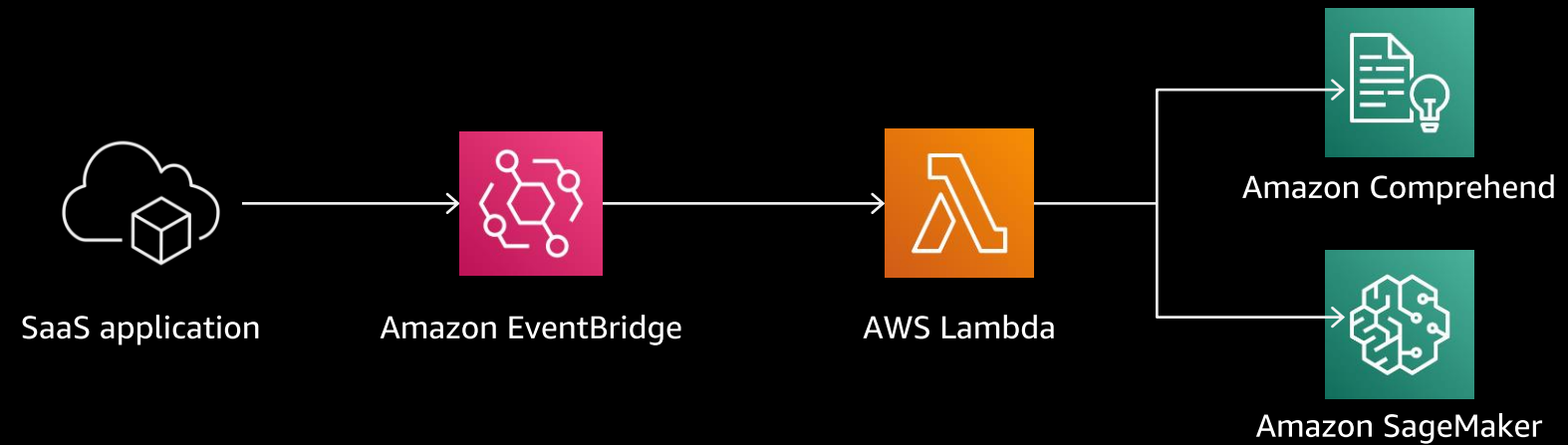
Take action



Run workflows



Apply intelligence

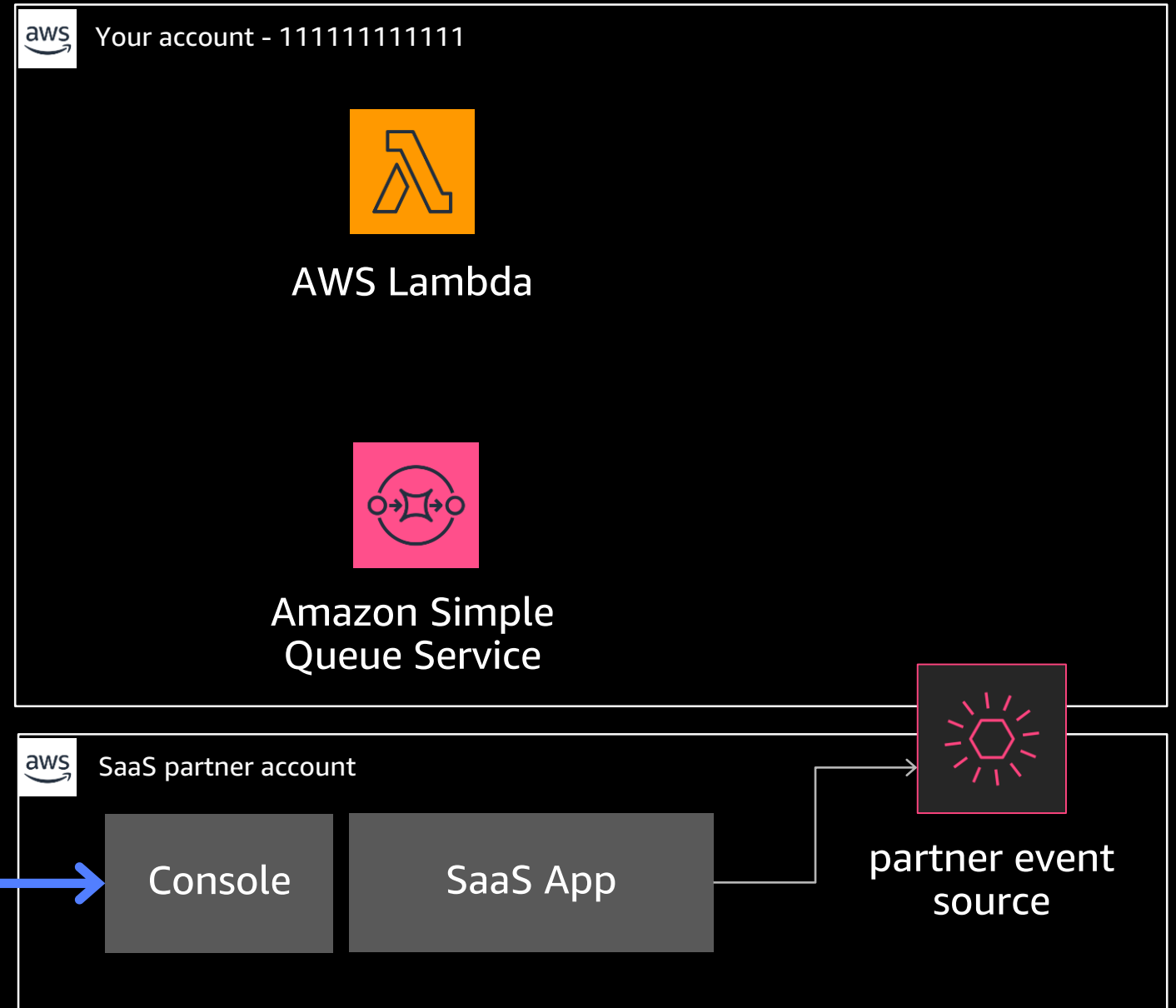


Onboarding a SaaS event source with EventBridge

1. Register your AWS account ID with SaaS partner

AWS account ID

AWS Region



Onboarding a SaaS event source with EventBridge

2. Associate the event source with an event bus

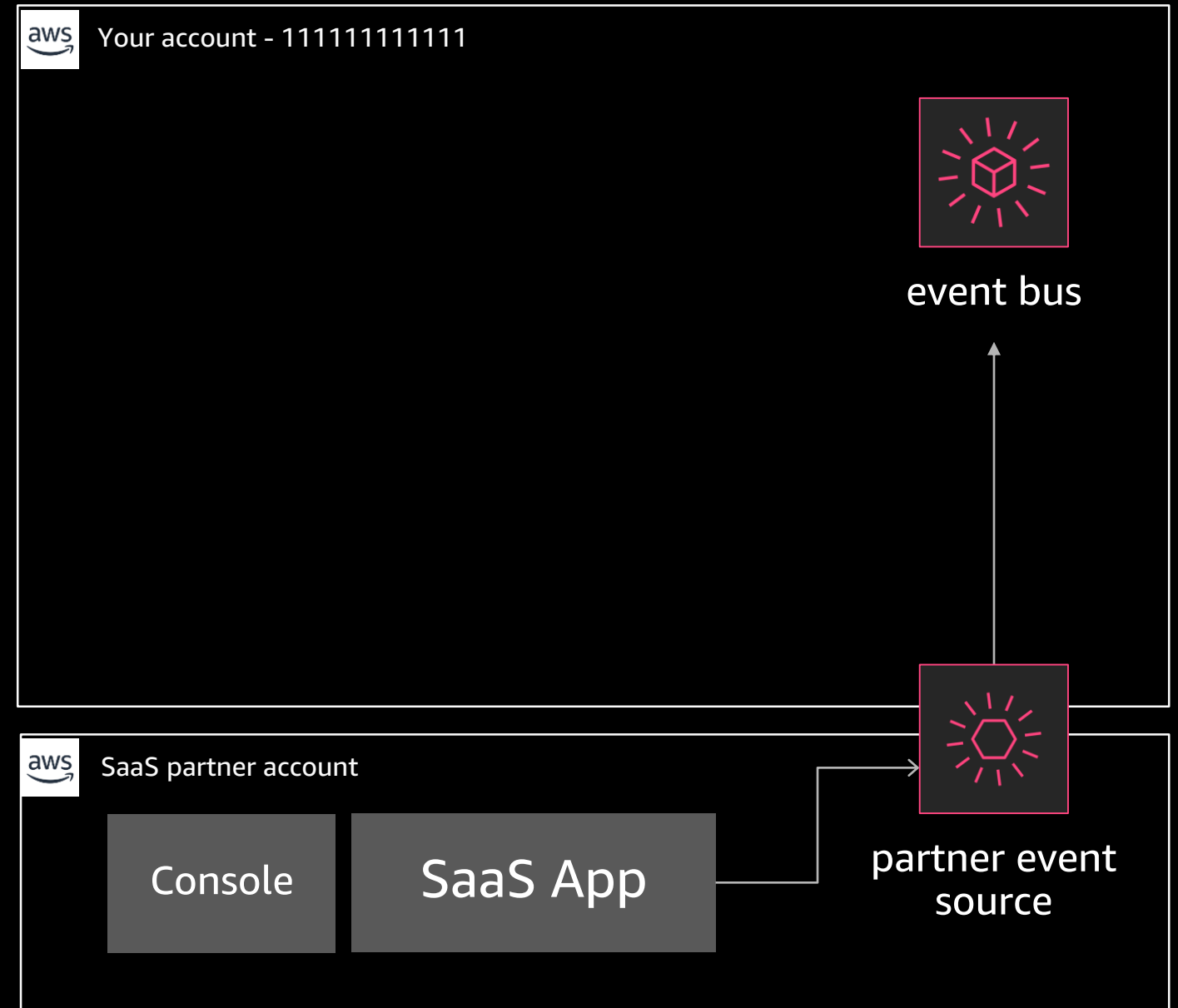
Partner event sources

Partner event sources (1) Associate with event bus

Search partner event source

< 1 >

Name	Status	Event Bus
aws.partner/example.com/123/example	Pending	No event bus associated



Onboarding a SaaS event source with EventBridge

3. Configure rules and targets

Create rule

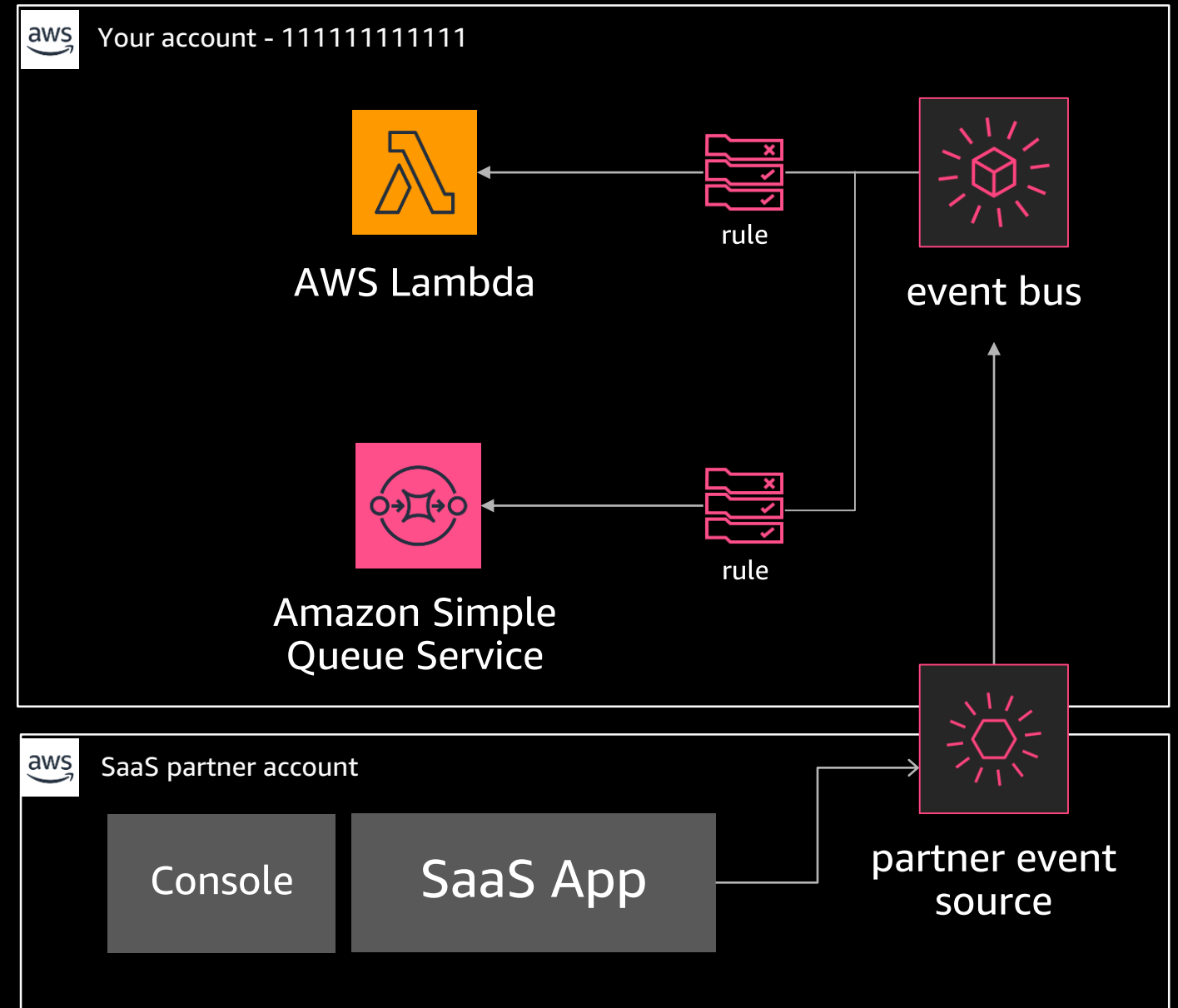
A rule watches for certain events and then routes them to AWS targets that you choose. You can create a rule that performs an AWS action automatically when another AWS action happens, or a rule that performs an AWS action regularly on a set schedule.

Name and description

Name

Maximum of 64 characters consisting of lower/upper case letters, -, _, .

Description - *optional*



Demo



Thank you!

James Beswick
Developer Advocate, AWS Serverless

jbeswick@amazon.com
[@jbesw](https://twitter.com/jbesw)

