AWS re:Invent

API306

Building event-driven architectures

Sam Dengler
Principal Solutions Architect
Serverless
Amazon Web Services

Stephen Liedig
Senior Solutions Architect
Serverless
Amazon Web Services

AWS re:Invent

aws

Agenda

Event-driven design

Event-driven processing

Workshop

Related breakout sessions

API304: Scalable serverless event-driven applications using Amazon SQS & Lambda

API315: Application integration patterns for microservices

Before we jump into the tech…

"When you start modeling events, it forces you to think about the *behavior* of the system. As opposed to thinking about the *structure* of the system."

Greg Young

A Decade of DDD, CQRS, Event Sourcing, 2016

Event-driven design

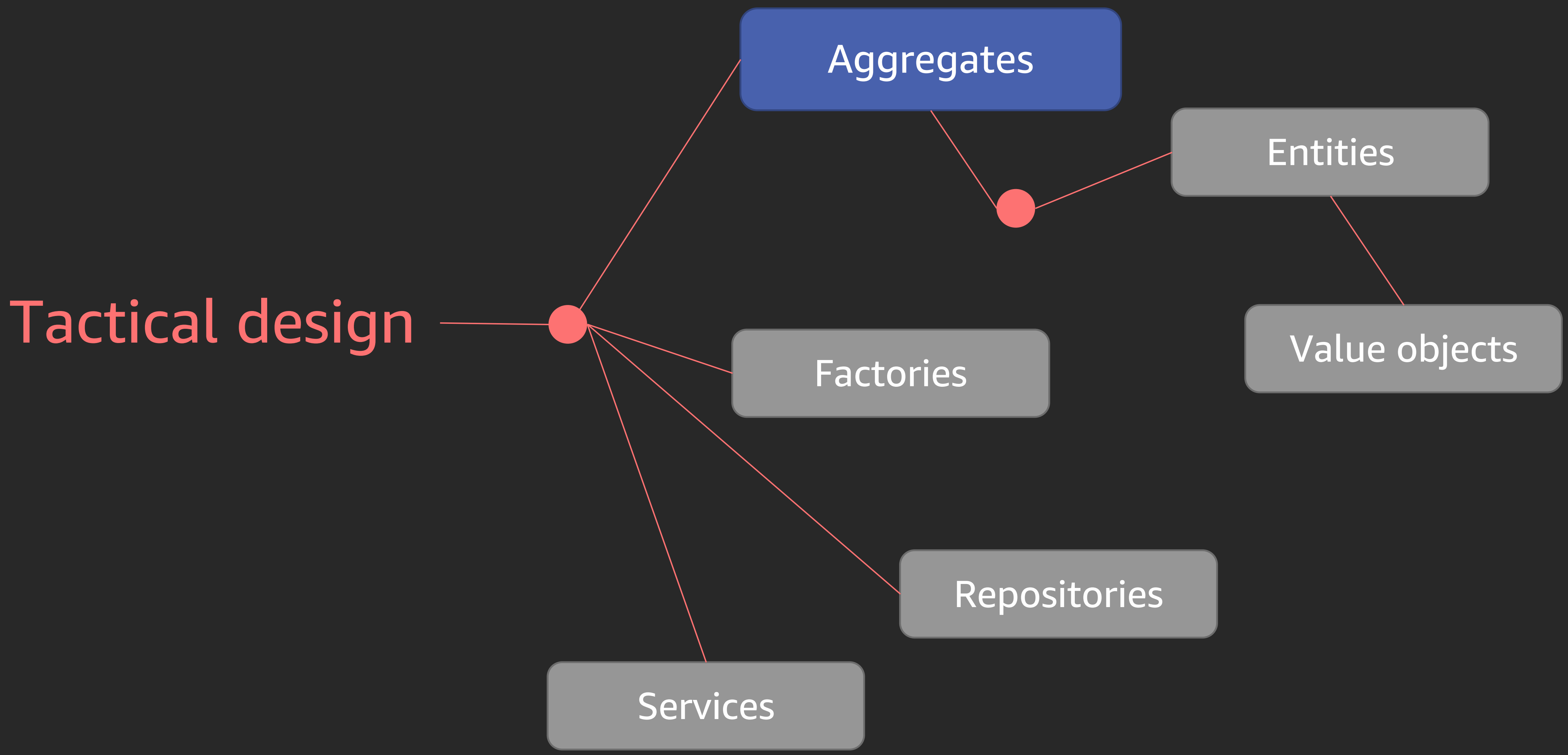
Domain-driven design

Provides a broad framework for making design decisions and developing a technical vocabulary for discussing domain design

Ubiquitous language—modeling the language of the business

Provides guidance about tactical design—model domains with entities, value objects, repositories and services, strategic design…

Elements of tactical design

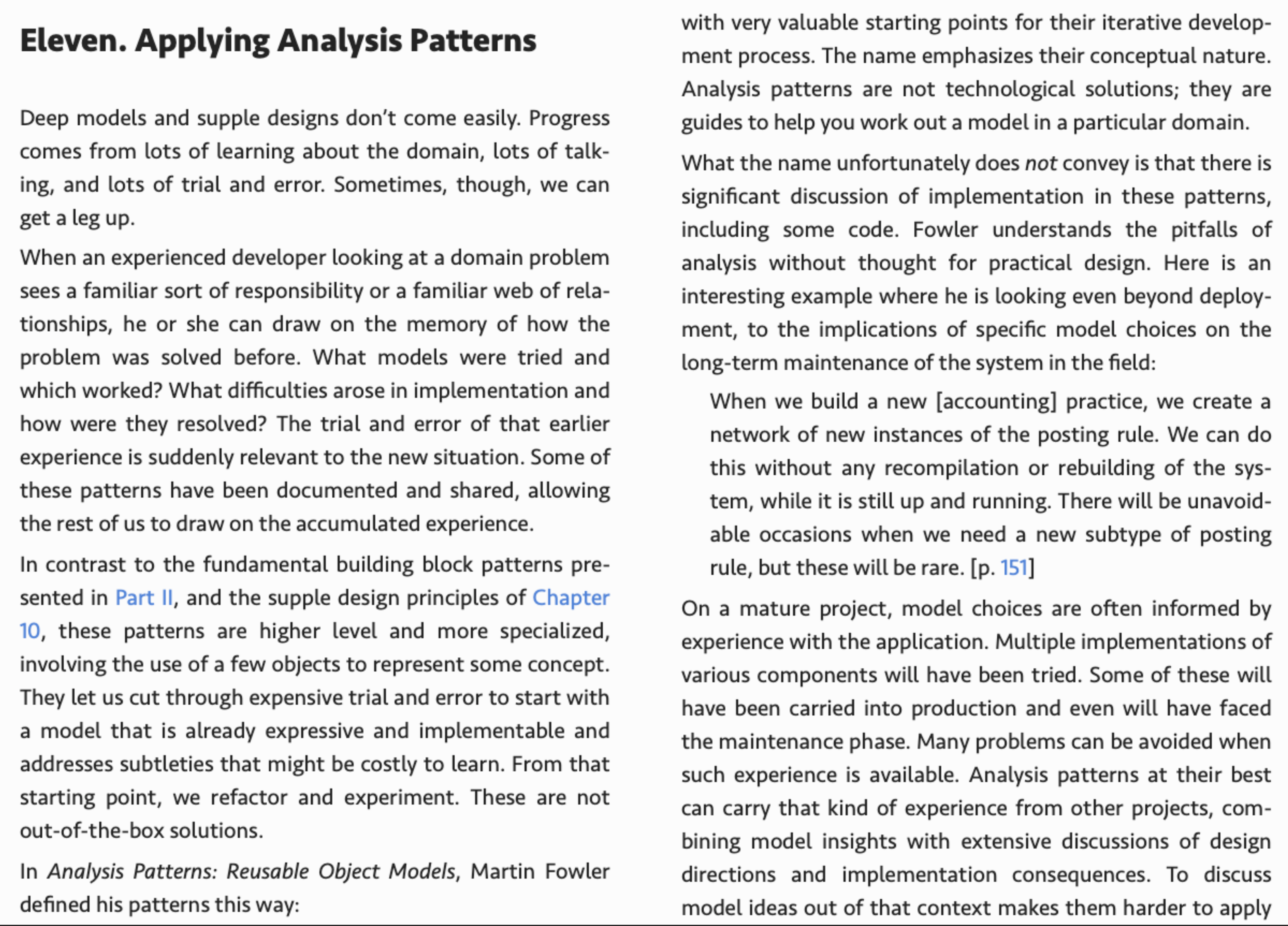


Strategic design

Start reading here…

“Strategic design principles must guide design decisions to reduce the interdependence of parts and improve clarity without losing critical interoperability and synergy. They must focus the model to capture the conceptual core of the system, the “vision” of the system.”

Excerpt From: Eric Evans. “Domain-Driven Design: Tackling Complexity in the Heart of Software.”



Eleven. Applying Analysis Patterns

Deep models and supple designs don’t come easily. Progress comes from lots of learning about the domain, lots of talking, and lots of trial and error. Sometimes, though, we can get a leg up.

When an experienced developer looking at a domain problem sees a familiar sort of responsibility or a familiar web of relationships, he or she can draw on the memory of how the problem was solved before. What models were tried and which worked? What difficulties arose in implementation and how were they resolved? The trial and error of that earlier experience is suddenly relevant to the new situation. Some of these patterns have been documented and shared, allowing the rest of us to draw on the accumulated experience.

In contrast to the fundamental building block patterns presented in Part II, and the supple design principles of Chapter 10, these patterns are higher level and more specialized, involving the use of a few objects to represent some concept. They let us cut through expensive trial and error to start with a model that is already expressive and implementable and addresses subtleties that might be costly to learn. From that starting point, we refactor and experiment. These are not out-of-the-box solutions.

In *Analysis Patterns: Reusable Object Models*, Martin Fowler defined his patterns this way:

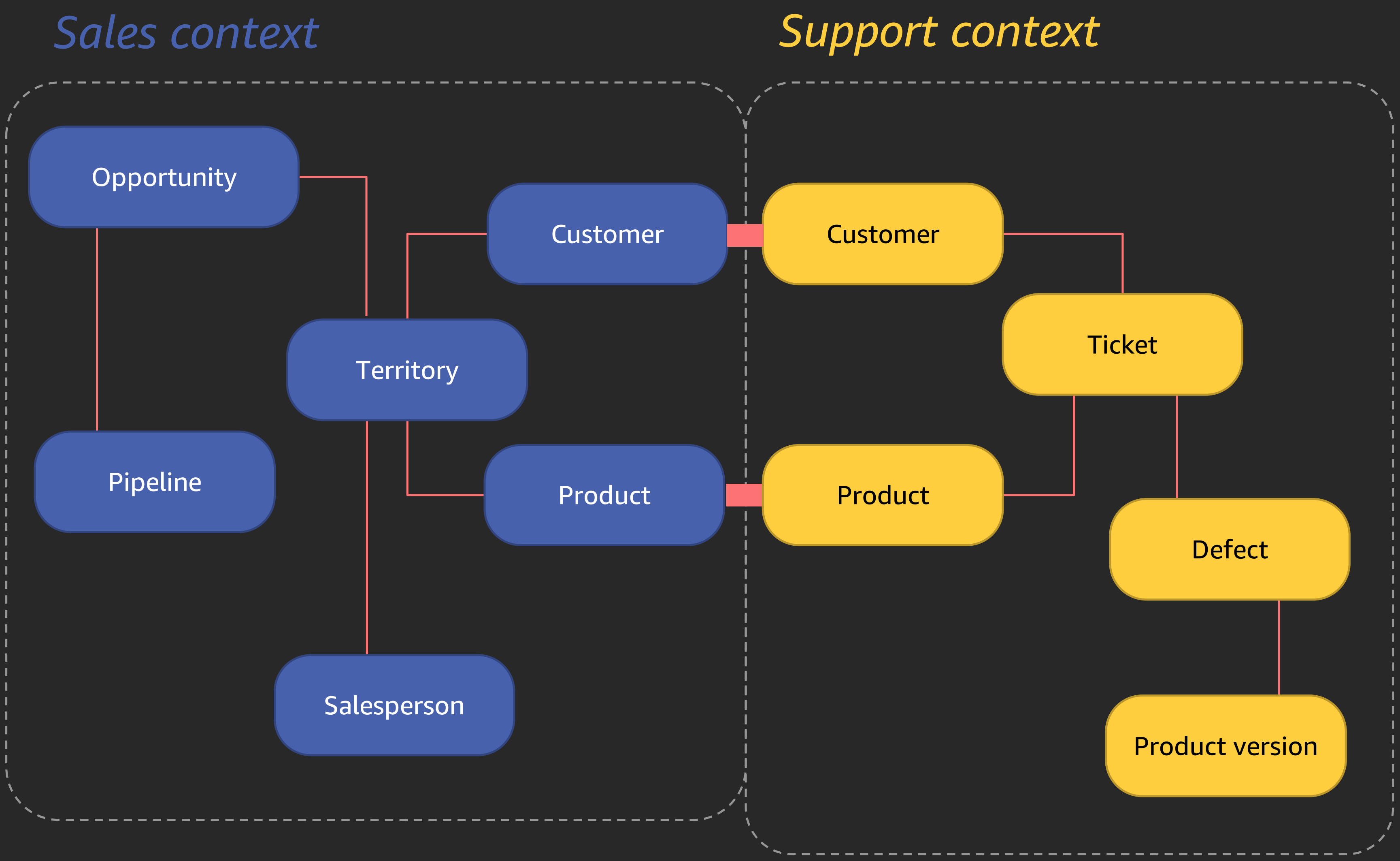
with very valuable starting points for their iterative development process. The name emphasizes their conceptual nature. Analysis patterns are not technological solutions; they are guides to help you work out a model in a particular domain.

What the name unfortunately does *not* convey is that there is significant discussion of implementation in these patterns, including some code. Fowler understands the pitfalls of analysis without thought for practical design. Here is an interesting example where he is looking even beyond deployment, to the implications of specific model choices on the long-term maintenance of the system in the field:

> When we build a new [accounting] practice, we create a network of new instances of the posting rule. We can do this without any recompilation or rebuilding of the system, while it is still up and running. There will be unavoidable occasions when we need a new subtype of posting rule, but these will be rare. [p. 151]

On a mature project, model choices are often informed by experience with the application. Multiple implementations of various components will have been tried. Some of these will have been carried into production and even will have faced the maintenance phase. Many problems can be avoided when such experience is available. Analysis patterns at their best can carry that kind of experience from other projects, combining model insights with extensive discussions of design directions and implementation consequences. To discuss model ideas out of that context makes them harder to apply

Domain-driven design



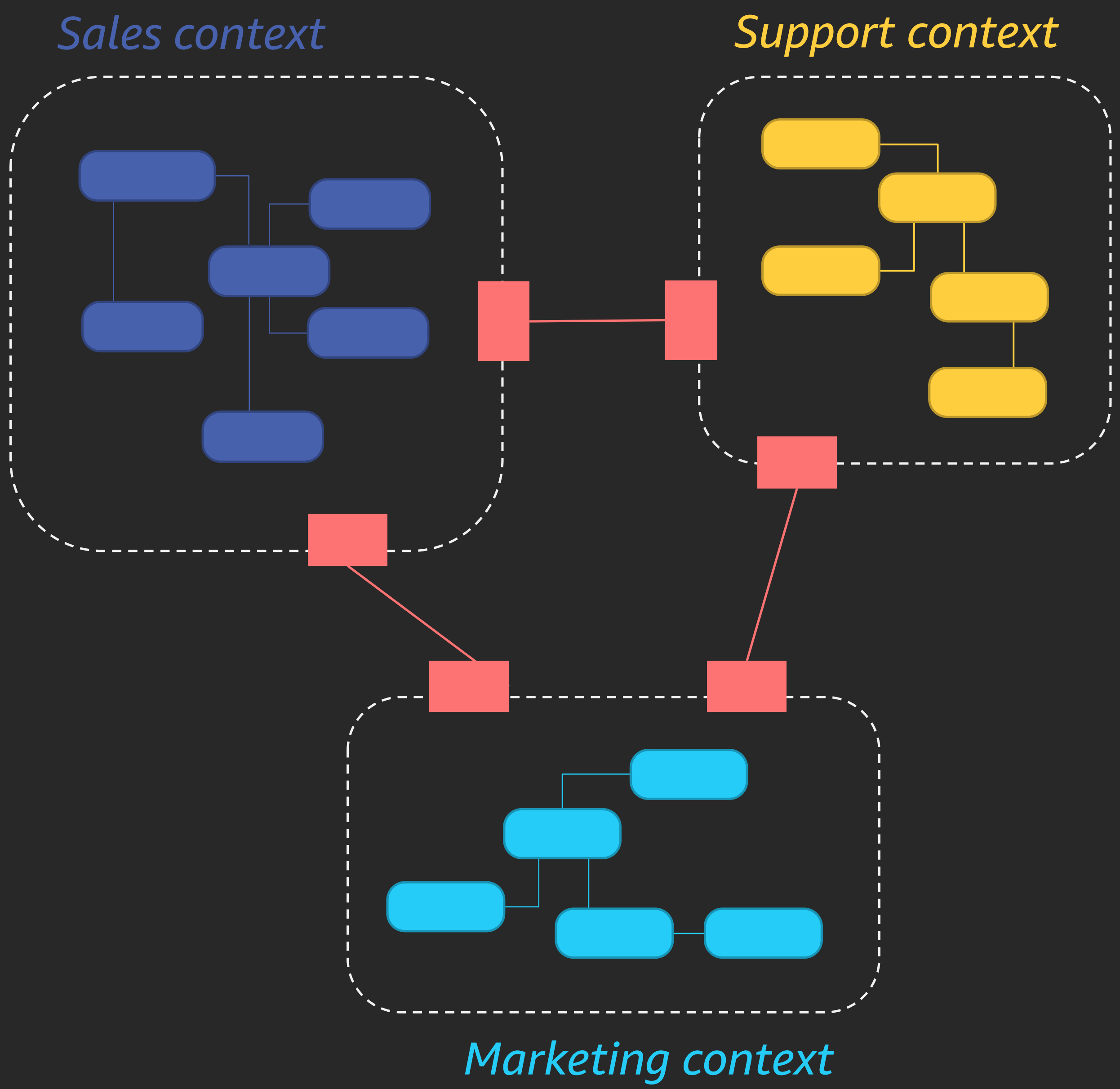
Bounded contexts are an essential modeling tool in DDD, microservices and event-driven architectures.

Identify explicit boundaries around our understanding of the ubiquitous language and the things people care about

Multiple models for a business concept

https://www.martinfowler.com/bliki/BoundedContext.html

Context maps



Bounded contexts alone don't provide a global view of your domain

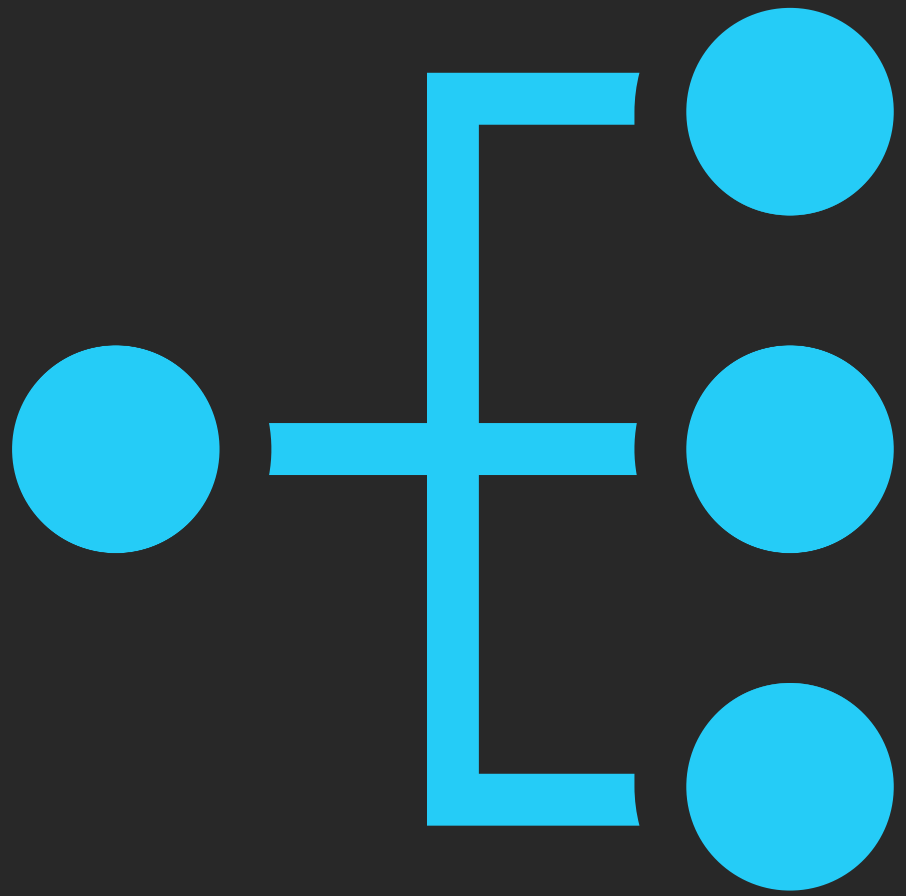
Context maps deal with mapping different but related ubiquitous languages, by integrating their bounded contexts

Evans and Vernon describe 7 patterns for integrating bounded contexts

- Shared kernel
- Customer/Supplier
- Conformist
- Anticorruption layer
- Separate ways
- Open/Host service
- Published language

Publish/Subscribe

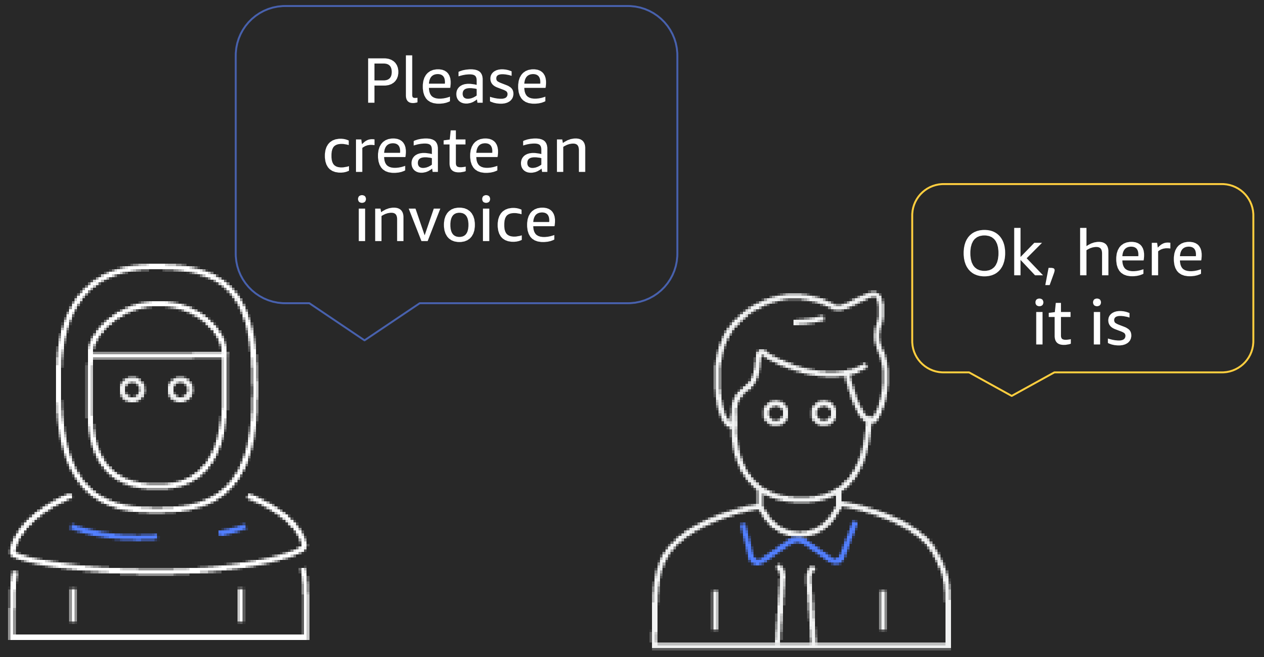
- One logical publisher for a given domain event; fully enforces a consistent boundary
- Addresses multiple types of coupling
- Each subscriber can react to domain events in their own bounded context



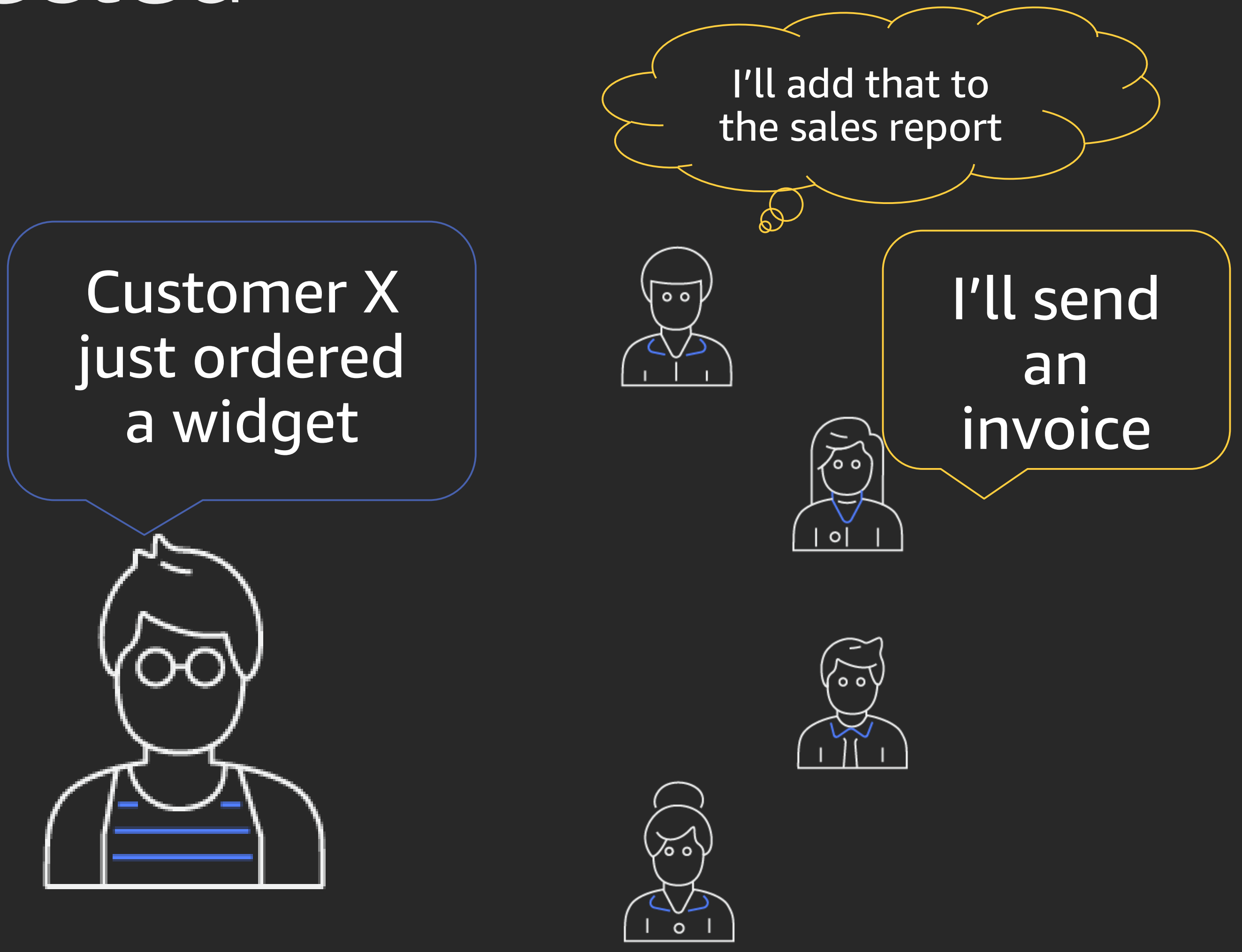
What are events?

- Type of message representing that something has happened
- Immutable—cannot change the past
- Represented as verbs in the past tense, e.g. “customer_created”
- Lightweight, correlated by properties that are common across bounded contexts “customer_id”
- Source system has no expectations on how an event is processed

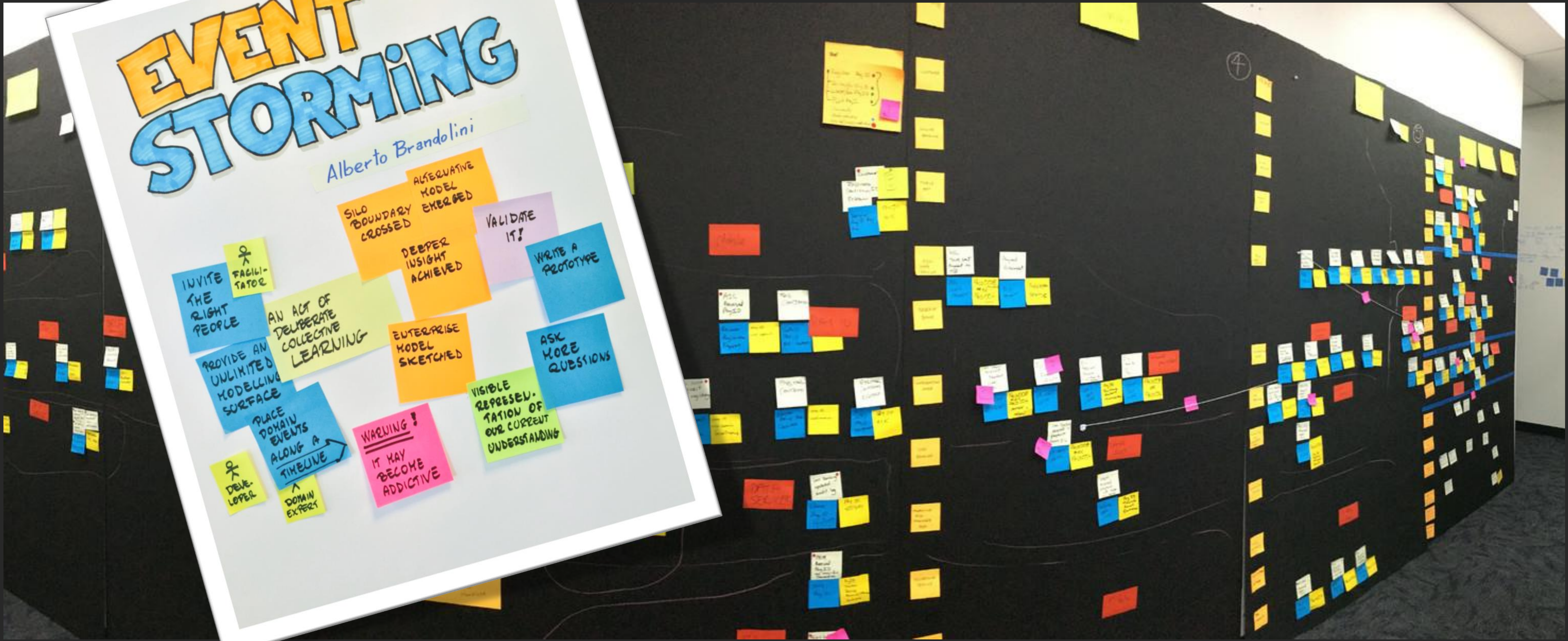
Events are observable, not directed



Directed commands


I'll add that to the sales report
Customer X just ordered a widget
I'll send an invoice

Observable events

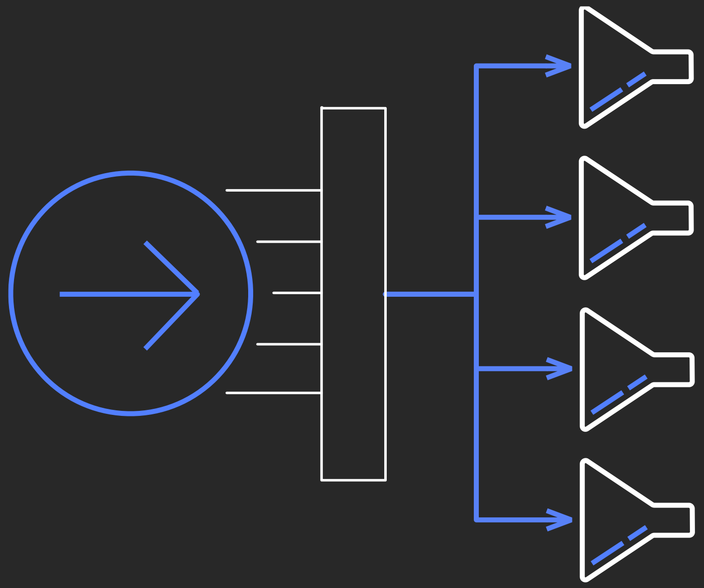


EVENT STORMING
Alberto Brandolini
INVITE THE RIGHT PEOPLE
FACILITATOR
AN ACT OF DELIBERATE COLLECTIVE LEARNING
SILO BOUNDARY CROSSED
ALTERNATIVE MODEL EMERGED
VALIDATE IT!
DEEPER INSIGHT ACHIEVED
WRITE A PROTOTYPE
PROVIDE AN UNLIMITED MODELLING SURFACE
ENTERPRISE MODEL SKETCHED
ASK MORE QUESTIONS
PLACE DOMAIN EVENTS ALONG A TIMELINE
VISIBLE REPRESENTATION OF OUR CURRENT UNDERSTANDING
WARNING! IT MAY BECOME ADDICTIVE
DEVELOPER
DOMAIN EXPERT

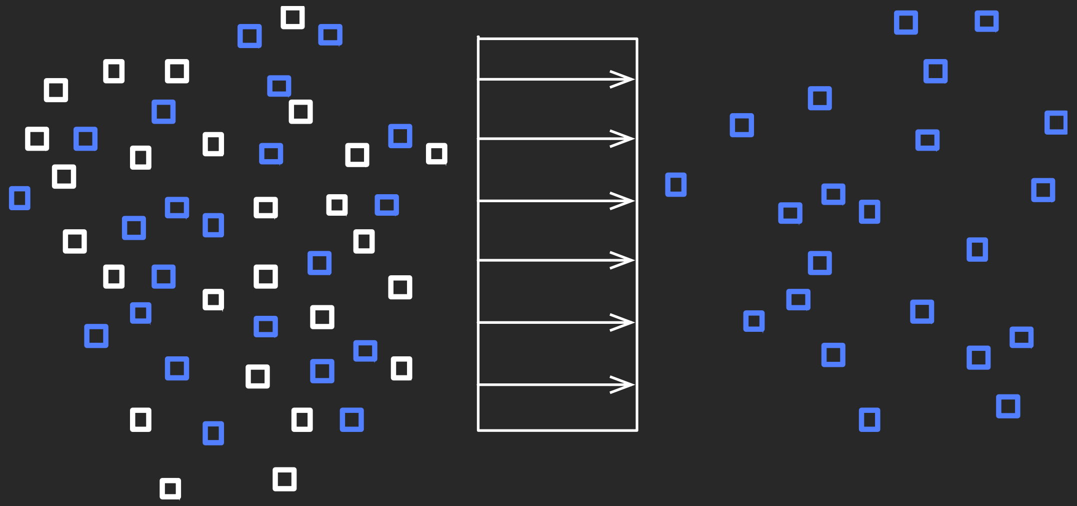
https://d33wubrfki0l68.cloudfront.net/36dc9d73a1362b252981cfc4bf1e6feb71ba12b1/415b6/images/eventstorm.jpeg

Event-driven processing

Event bus



Abstracts producers and consumers



Selects and filters events

Amazon Simple Notification Service

Fully managed pub/sub messaging for microservices, distributed systems, and serverless applications

Amazon SNS

What is it?

Simple, flexible, fully managed publish/subscribe messaging and mobile push notification service for high throughput, highly reliable message delivery

Use case

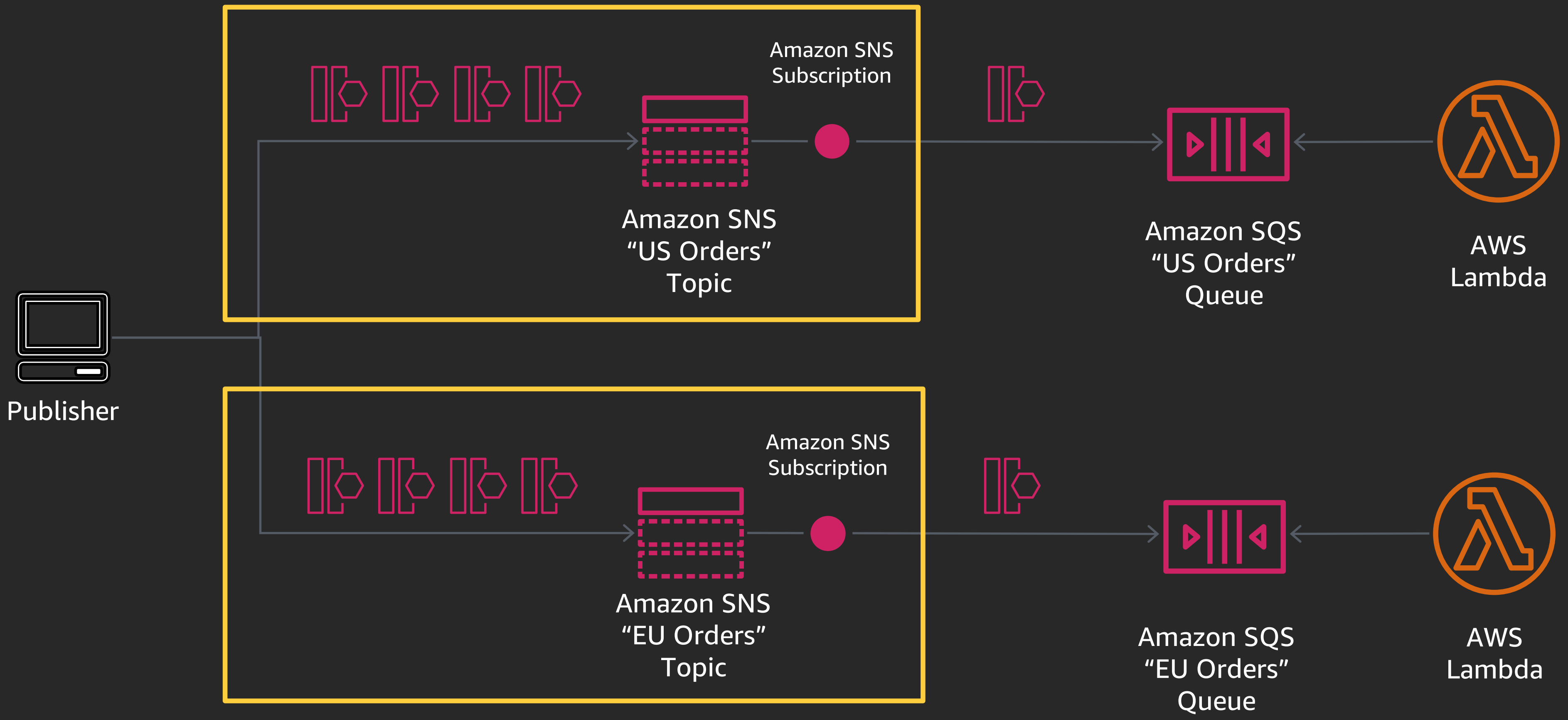
Push messages to a variety of endpoints and clients in distributed systems, microservices, and serverless applications, and enable event-driven architecture

Cool capabilities

Highly reliable delivery of any volume of messages to any number of recipients across multiple protocols

```
> aws sns create-topic --name my-topic
```

Mapping events to topics



Each message type is mapped to logical destination

Amazon SNS Message Filters

- Publishers do not need to route message
- Subscribers do not need to filter for message of interest
- Lowers cost

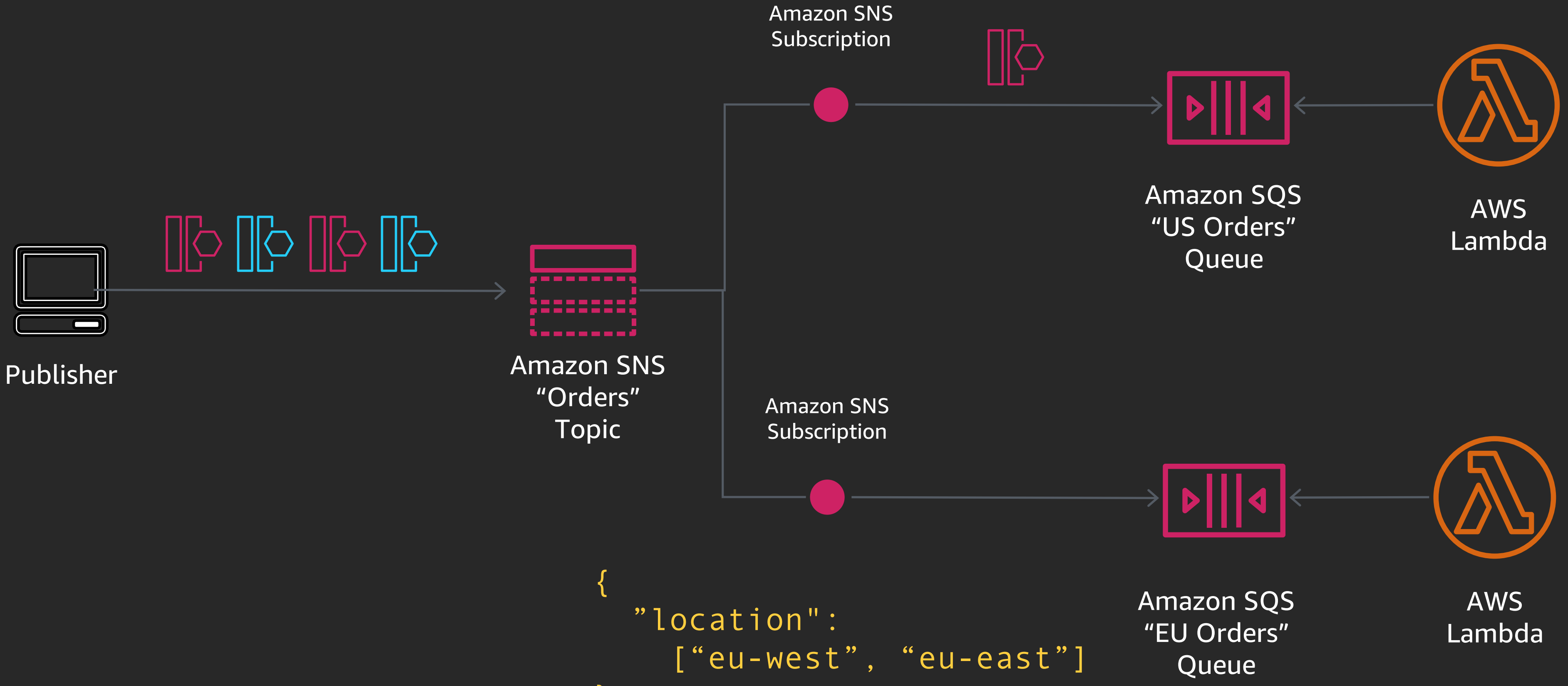
Message Attributes

```
{
  ”location": “eu-west”]
}
```

Filter Policy

```
{
  ”location":
    [”us-west”, “us-east”]
}
```

```
{
  ”location":
    [“eu-west”, “eu-east”]
}
```


Publisher
Amazon SNS “Orders” Topic
Amazon SNS Subscription
Amazon SNS Subscription
Amazon SQS “US Orders” Queue
Amazon SQS “EU Orders” Queue
AWS Lambda
AWS Lambda

Message-filtering operators

Exact matching on string values (whitelisting)

```
Subscription filter policy
  {"sport": ["rugby"]}

matches message attribute
  {"sport": "rugby"}
```

Prefix matching on string values

```
Subscription filter policy
  {"sport": [{"prefix": "bas"}]}

matches message attributes such as
  {"sport": "baseball"}
and
  {"sport": "basketball"}
```

Anything-but matching on string values (blacklisting)

```
Subscription filter policy
  {"sport": [{"anything-but": "rugby"}]}

matches message attributes such as
  {"sport": "baseball"} and {"sport": "basketball"} and {"sport": "football"}
but not
  {"sport": "rugby"}
```

Message-filtering operators

Exact matching on numeric values

```
Subscription filter policy
  {"balance": [{"numeric":
                ["=", 301.5]}]}

matches message attributes
  {"balance": 301.500}

and
  {"balance": 3.015e2}
```

Range matching on numeric values

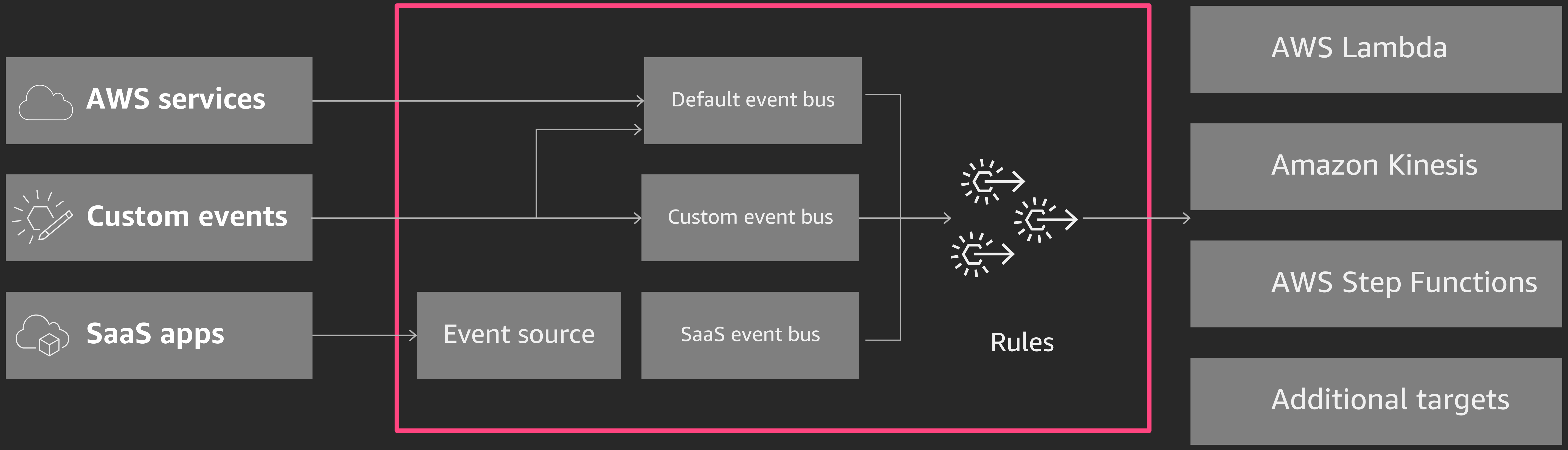
```
Subscription filter policy
 {"balance": [{"numeric":
               ["<", 0]}]}

matches negative numbers only,
and
{"balance": [{"numeric":
               [">", 0, "<=",
150]}]}
matches any positive number up
to 150.
```

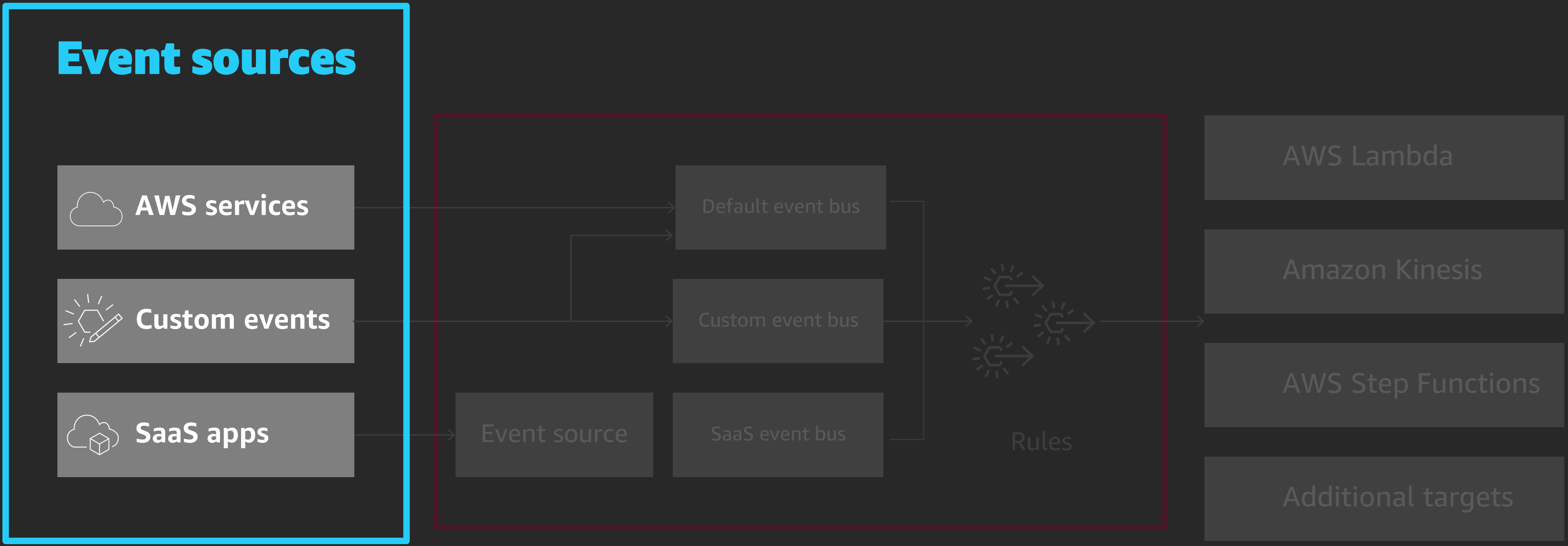
Amazon EventBridge

A serverless event bus service for AWS services, your own applications, and SaaS providers

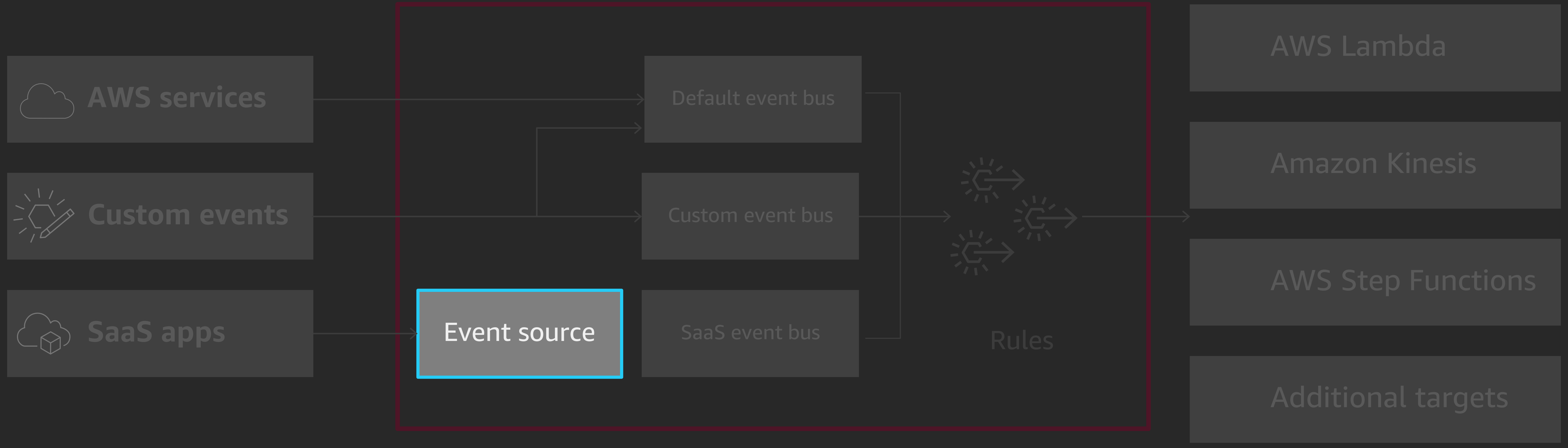
EventBridge



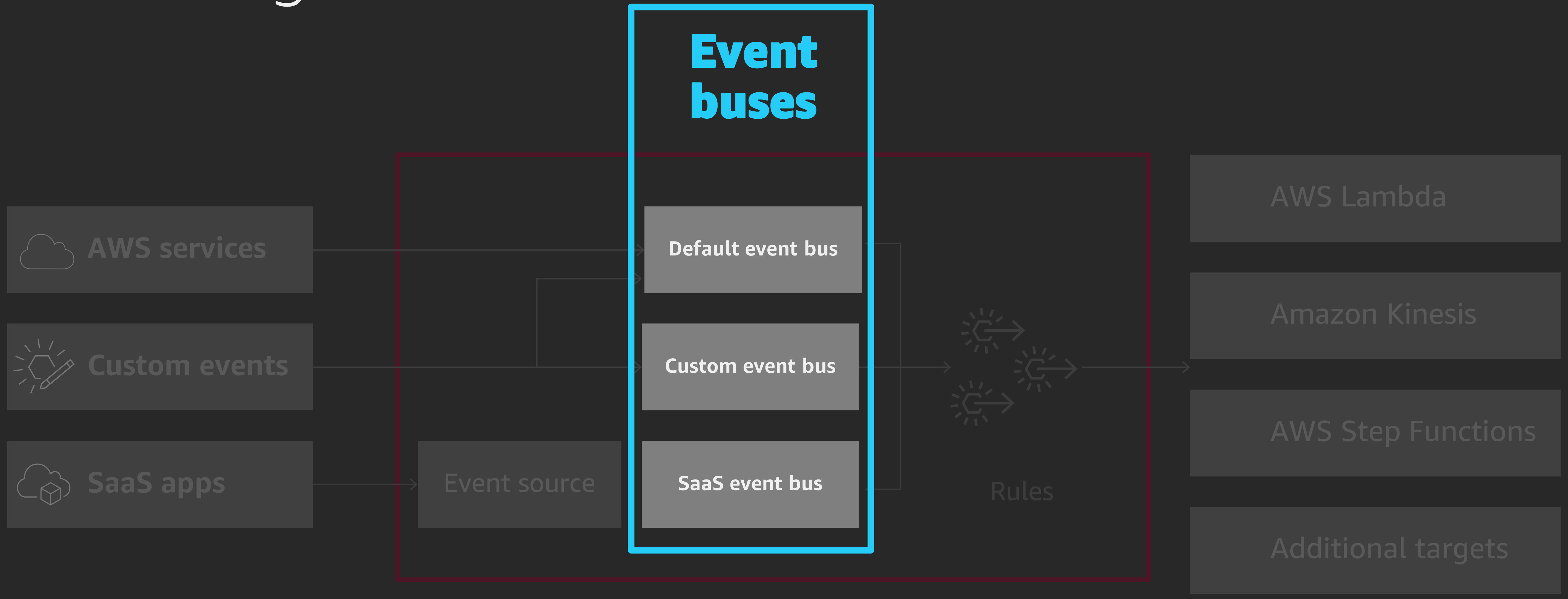
EventBridge



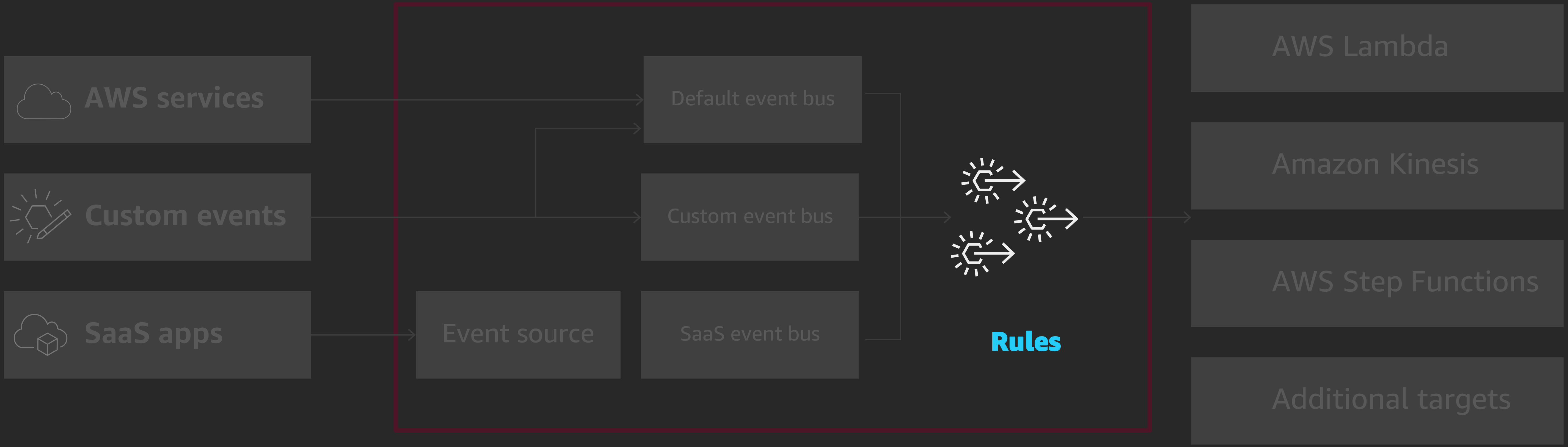
EventBridge



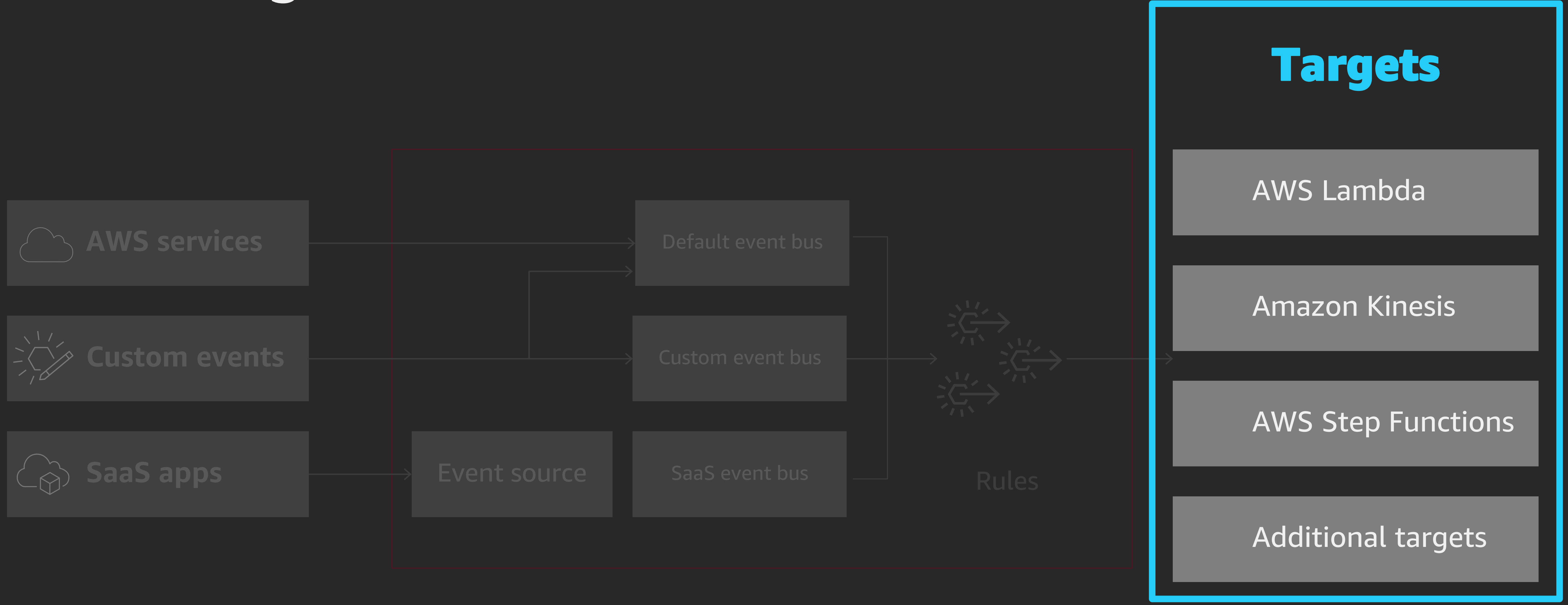
EventBridge



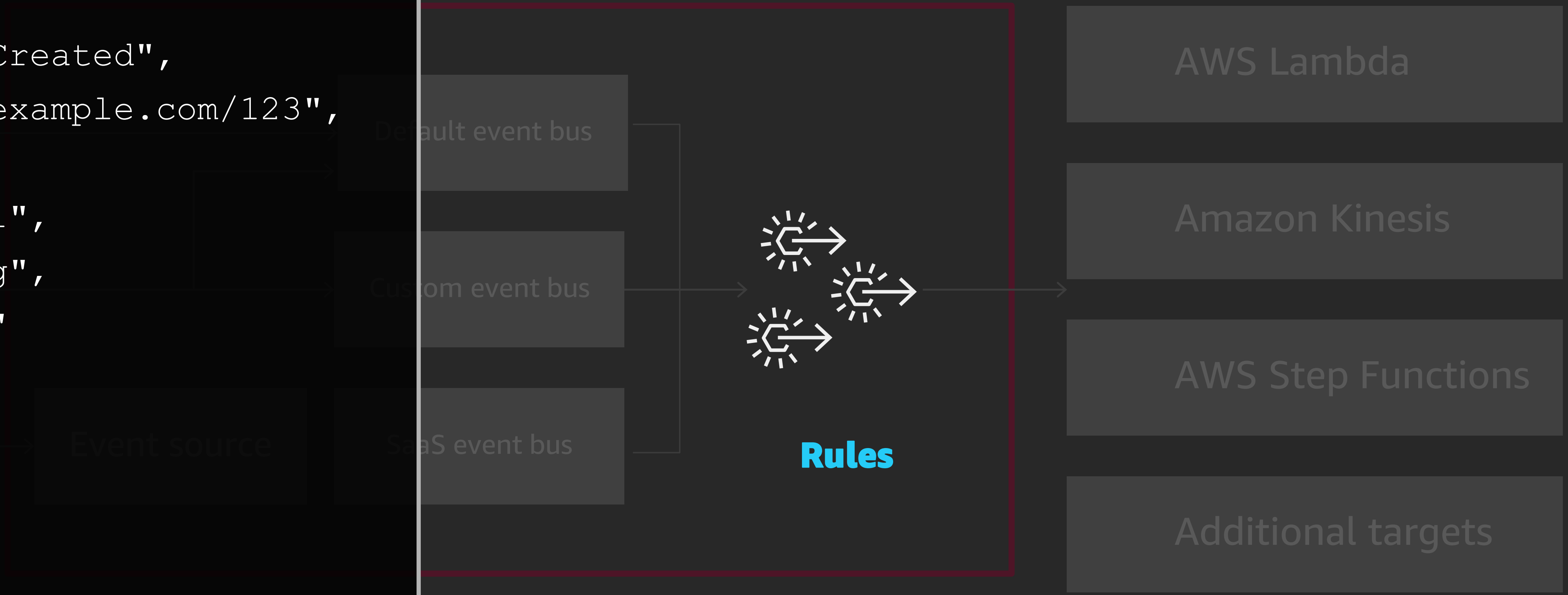
EventBridge



EventBridge



EventBridge



Example event:

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

EventBridge

Example event:

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

Example rule:

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

EventBridge

Example event:

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

Example rule:

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

EventBridge

Example event:

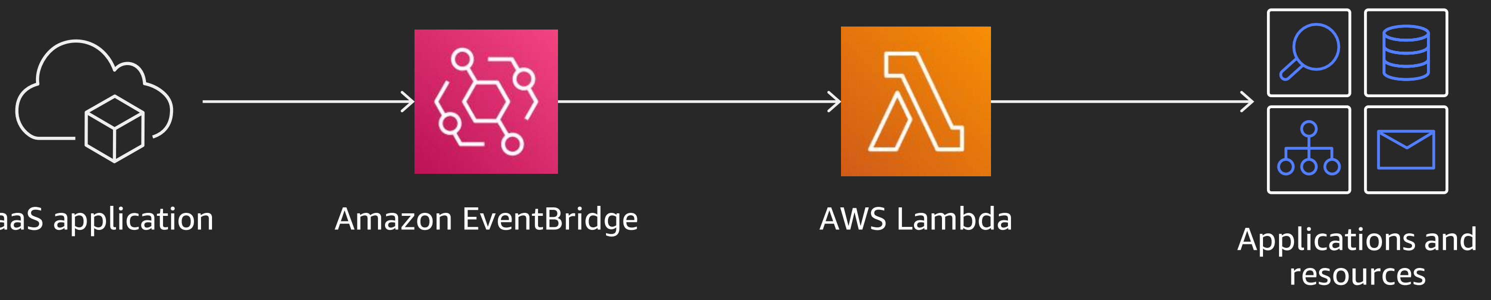
```
{
  "detail-type": "Ticket Created",
  "source": "aws.partner/example.com/123",
  "detail": {
    "ticketId": ”987654321",
    "department": "billing",
    "creator": "user12345"
    ...
  }
}
```

Example rule:

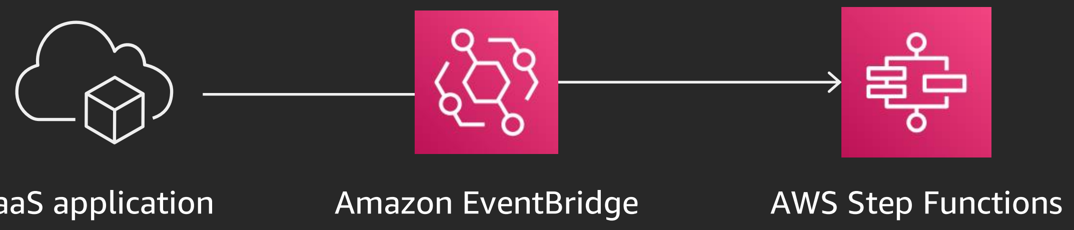
```
{
  "detail-type": ["Ticket Resolved"]
}
```

Common use cases

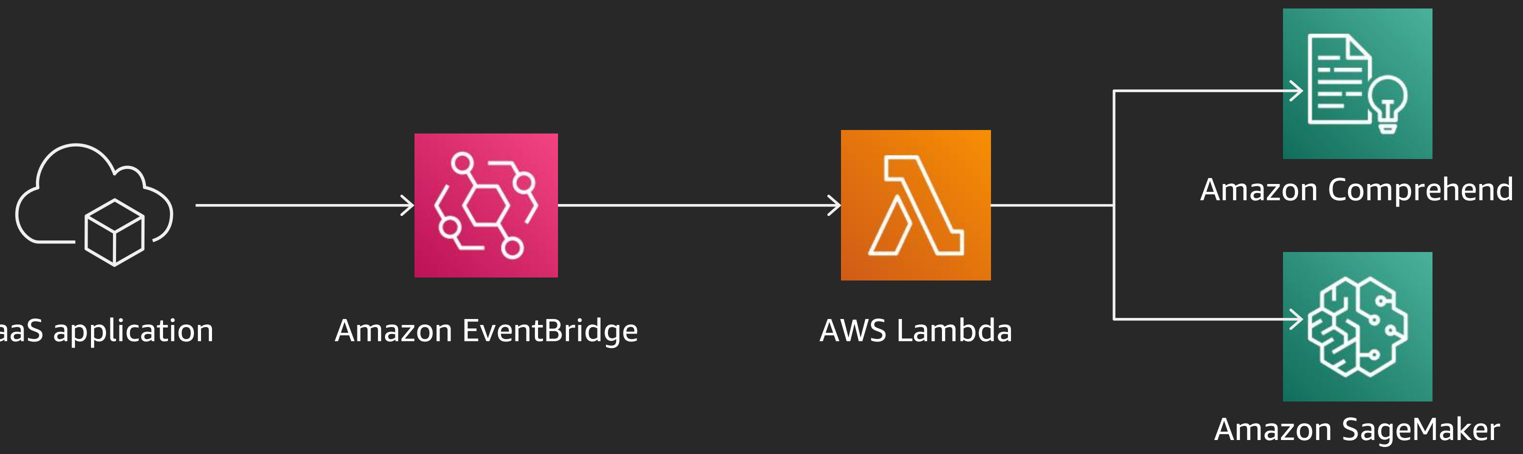
Take action



Run workflows

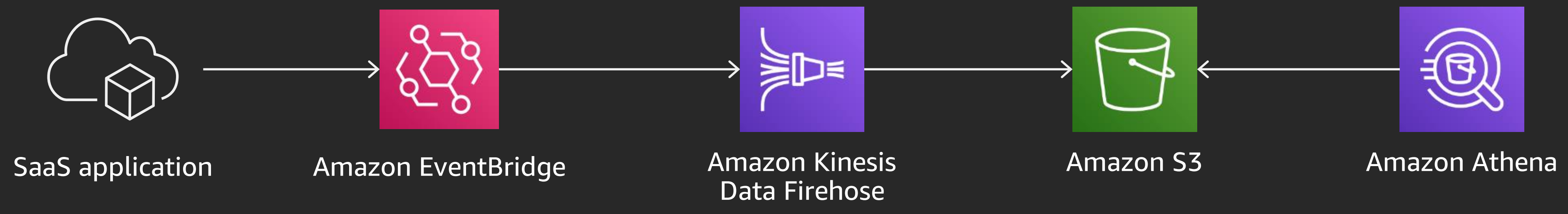


Apply intelligence

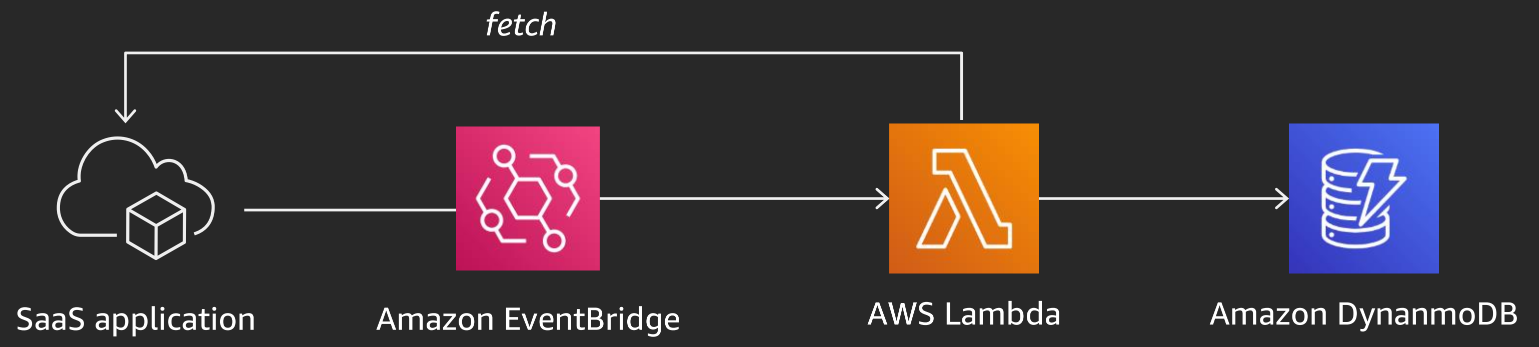


Common use cases

Audit and analyze



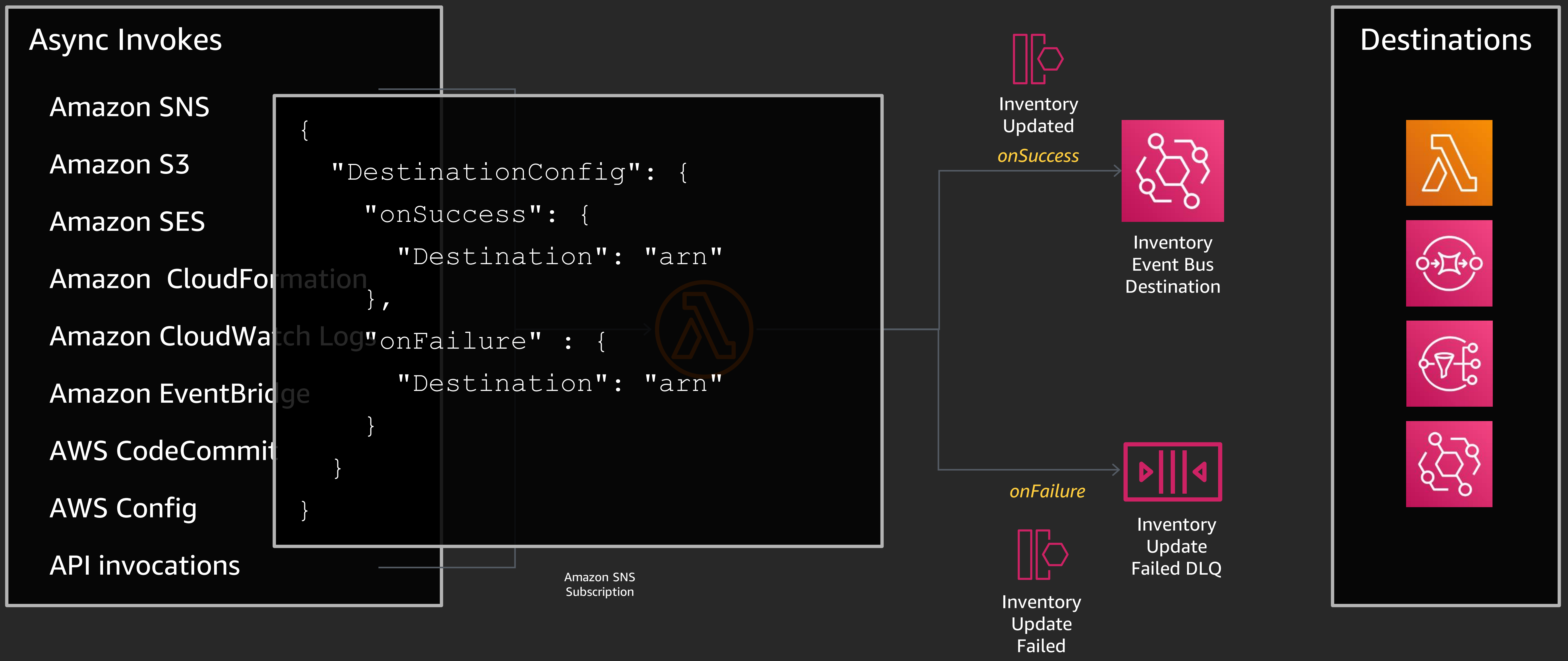
Synchronize data



AWS Lambda Destinations

Gain visibility to asynchronous invocation results and route the results to an AWS service without writing code

AWS Lambda Destinations



```
{
  "DestinationConfig": {
    "onSuccess": {
      "Destination": "arn"
    },
    "onFailure" : {
      "Destination": "arn"
    }
  }
}
```

Workshop

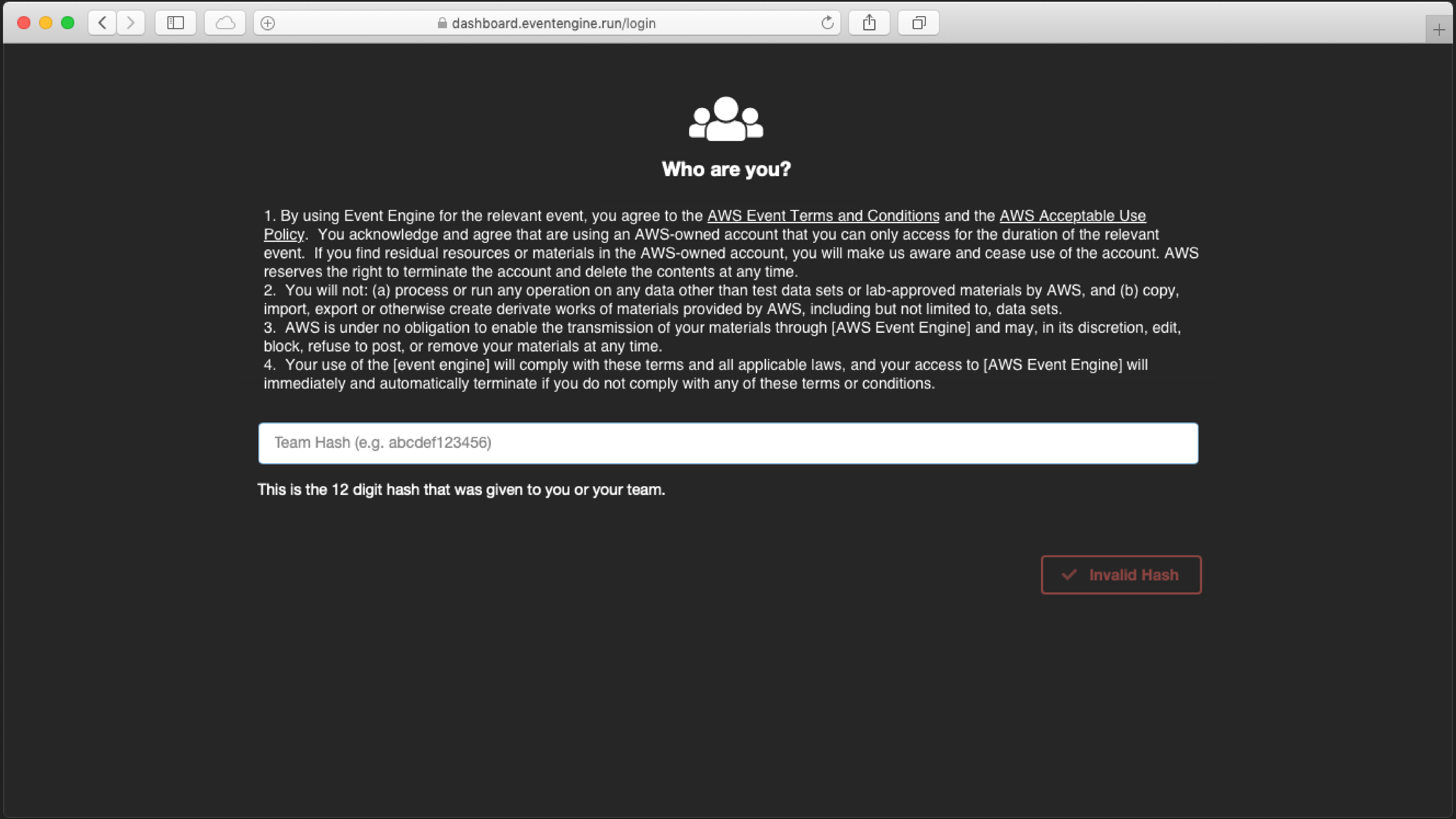
AWS re:Invent

aws

Getting started

Instructions: https://event-driven-architecture.workshop.aws

Lab environment: https://dashboard.eventengine.run



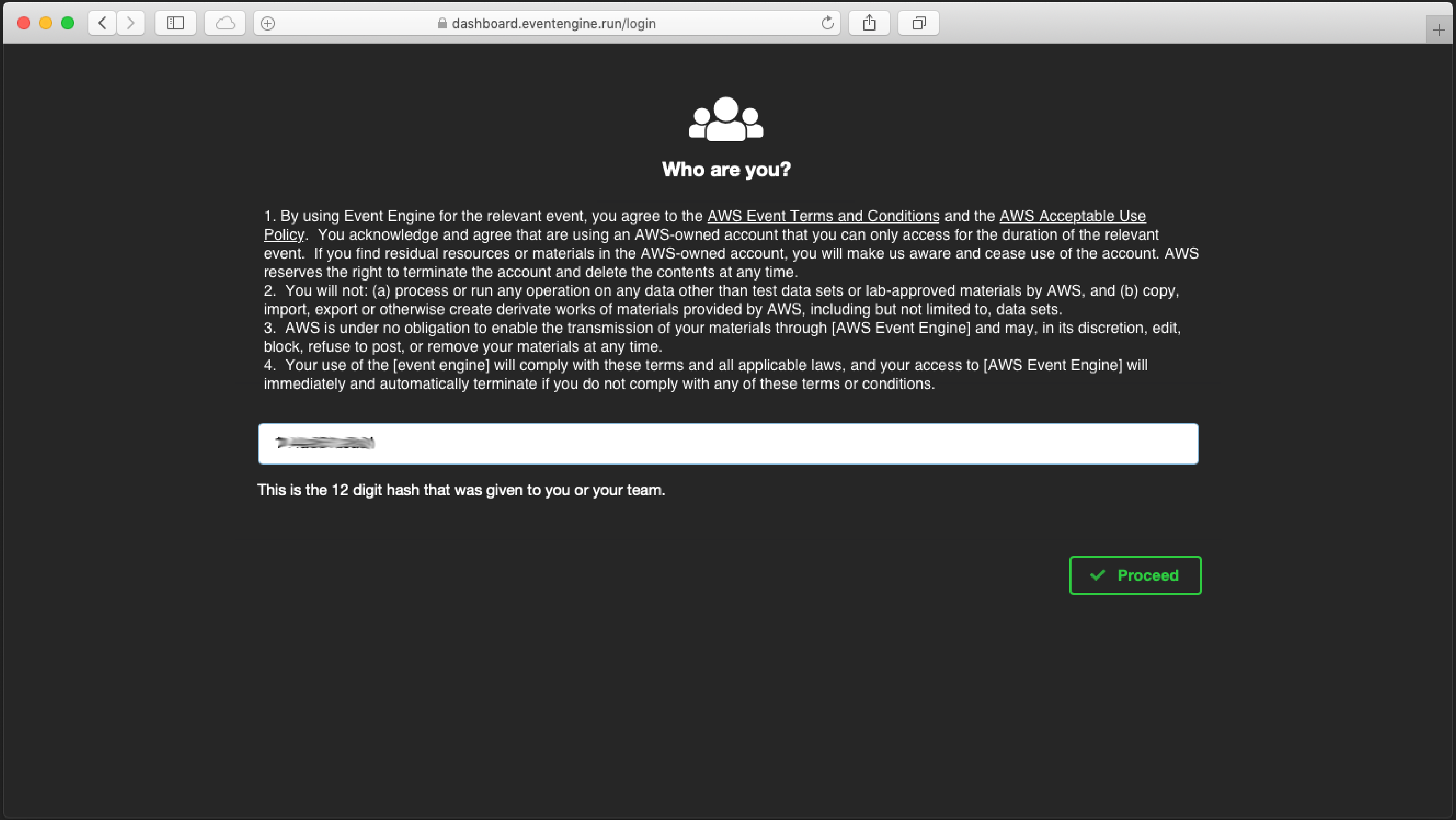
dashboard.eventengine.run/login

Who are you?

1. By using Event Engine for the relevant event, you agree to the AWS Event Terms and Conditions and the AWS Acceptable Use Policy. You acknowledge and agree that are using an AWS-owned account that you can only access for the duration of the relevant event. If you find residual resources or materials in the AWS-owned account, you will make us aware and cease use of the account. AWS reserves the right to terminate the account and delete the contents at any time.
2. You will not: (a) process or run any operation on any data other than test data sets or lab-approved materials by AWS, and (b) copy, import, export or otherwise create derivate works of materials provided by AWS, including but not limited to, data sets.
3. AWS is under no obligation to enable the transmission of your materials through [AWS Event Engine] and may, in its discretion, edit, block, refuse to post, or remove your materials at any time.
4. Your use of the [event engine] will comply with these terms and all applicable laws, and your access to [AWS Event Engine] will immediately and automatically terminate if you do not comply with any of these terms or conditions.

Team Hash (e.g. abcdef123456)

This is the 12 digit hash that was given to you or your team.

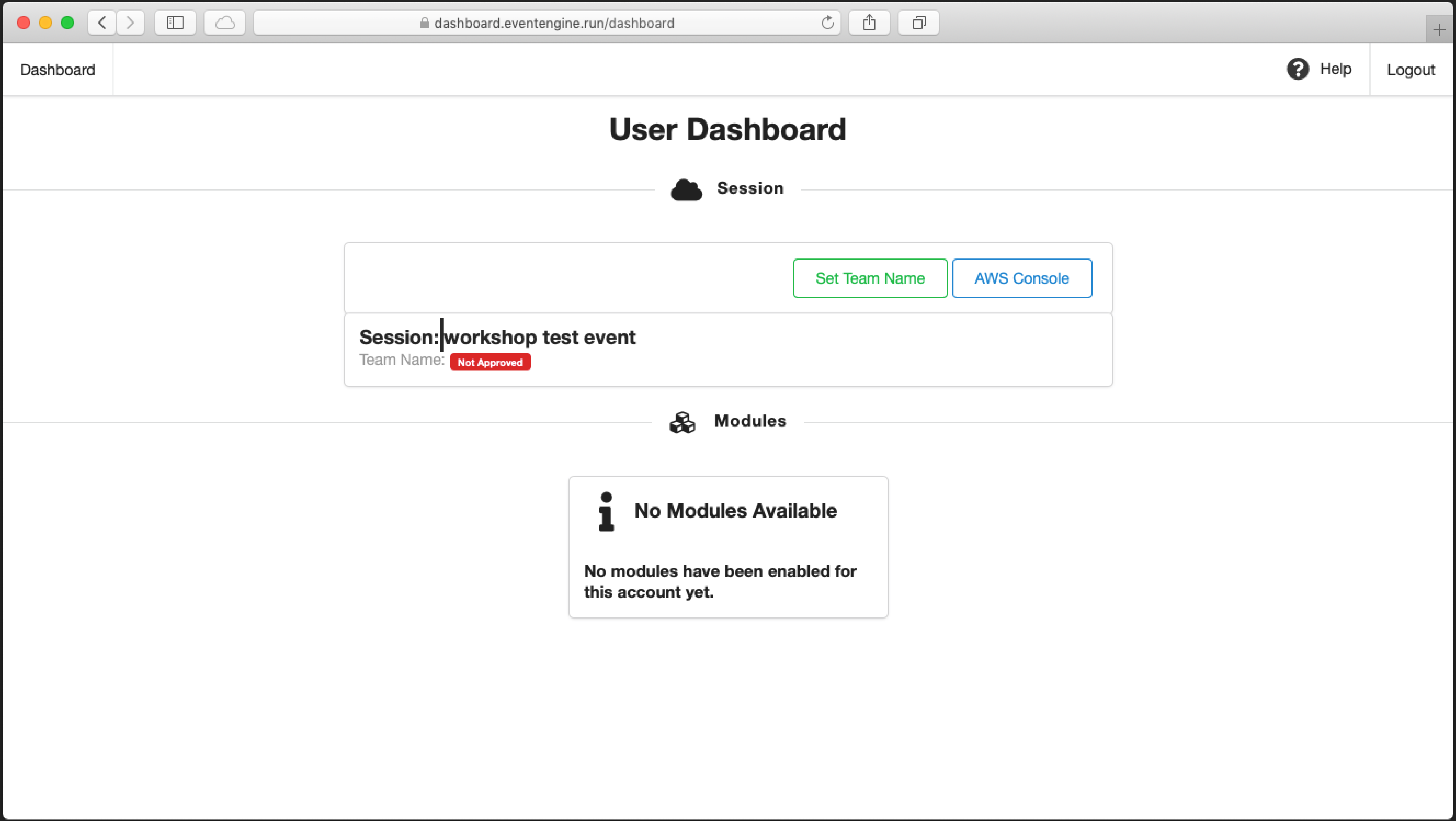


Who are you?

1. By using Event Engine for the relevant event, you agree to the AWS Event Terms and Conditions and the AWS Acceptable Use Policy. You acknowledge and agree that are using an AWS-owned account that you can only access for the duration of the relevant event. If you find residual resources or materials in the AWS-owned account, you will make us aware and cease use of the account. AWS reserves the right to terminate the account and delete the contents at any time.
2. You will not: (a) process or run any operation on any data other than test data sets or lab-approved materials by AWS, and (b) copy, import, export or otherwise create derivate works of materials provided by AWS, including but not limited to, data sets.
3. AWS is under no obligation to enable the transmission of your materials through [AWS Event Engine] and may, in its discretion, edit, block, refuse to post, or remove your materials at any time.
4. Your use of the [event engine] will comply with these terms and all applicable laws, and your access to [AWS Event Engine] will immediately and automatically terminate if you do not comply with any of these terms or conditions.

[redacted]

This is the 12 digit hash that was given to you or your team.


dashboard.eventengine.run/dashboard

Dashboard

Help

Logout

User Dashboard

Session

Set Team Name

AWS Console

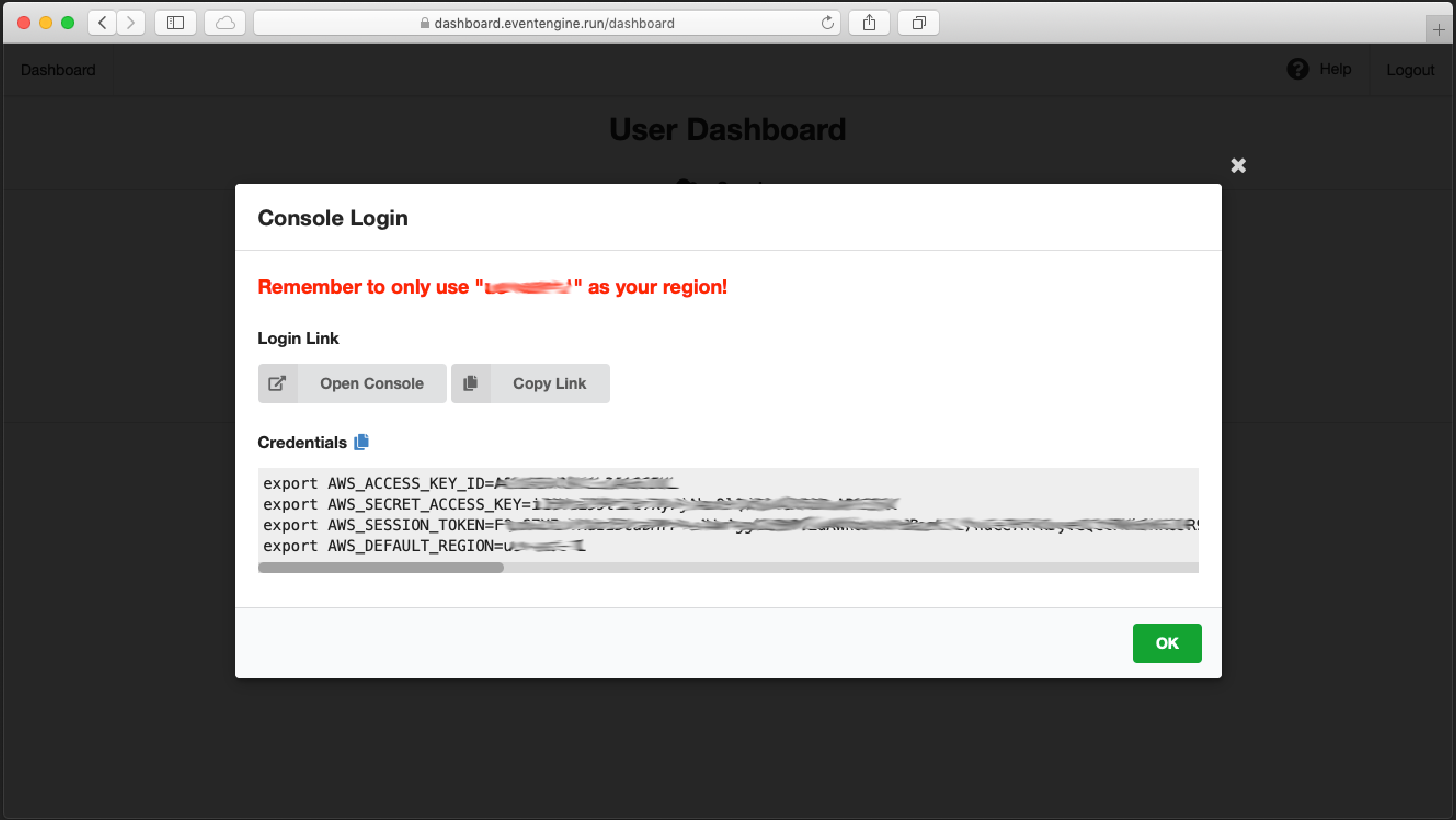
Session: workshop test event

Team Name: Not Approved

Modules

No Modules Available

No modules have been enabled for this account yet.



dashboard.eventengine.run/dashboard

Dashboard

Help

Logout

User Dashboard

Console Login

Remember to only use "[redacted]" as your region!

Login Link

Open Console

Copy Link

Credentials

```
export AWS_ACCESS_KEY_ID=[redacted]
export AWS_SECRET_ACCESS_KEY=[redacted]
export AWS_SESSION_TOKEN=[redacted]
export AWS_DEFAULT_REGION=[redacted]
```

OK

Thank you!

Please complete the session survey in the mobile app.