AWS re:Invent

ARC314-R

Decoupled microservices: Building scalable applications

Dirk Fröhner
Solutions Architect
Amazon Web Services

Christian Müller
Solutions Architect
Amazon Web Services

AWS re:Invent

aws

Agenda

Introduction

Application integration patterns

Concrete use cases – Our labs today

Choose your own adventure – Work on your most relevant labs

Related breakouts

API315 Application integration patterns for microservices

API304 – Scalable serverless event-driven apps using Amazon SQS & Lambda

API306 – Building event-driven architectures

API307 – Build efficient and scalable distributed applications using Amazon MQ

API309 – Durable serverless architecture: Working with dead-letter queues

API311 – Managing business processes using AWS Step Functions

API312 – How to select the right application integration service

API316 – Building serverless workflows using AWS Step Functions

API318 – Deep dive on event-driven development with Amazon EventBridge

Introduction

> “If your application is **cloud-native**, or **large-scale**, or **distributed**, and **doesn’t include** a **messaging** component, that’s probably a **bug**.”

Tim Bray

Distinguished Engineer

AWS Messaging, Workflow Management

AWS re:Invent

Potential drawbacks of synchronous systems

Synchronous systems are tightly coupled

A problem in a synchronous downstream dependency has immediate impact on upstream callers

Retries from upstream callers can all too easily fan out and amplify problems

Photo: Dirk Fröhner

Application integration patterns

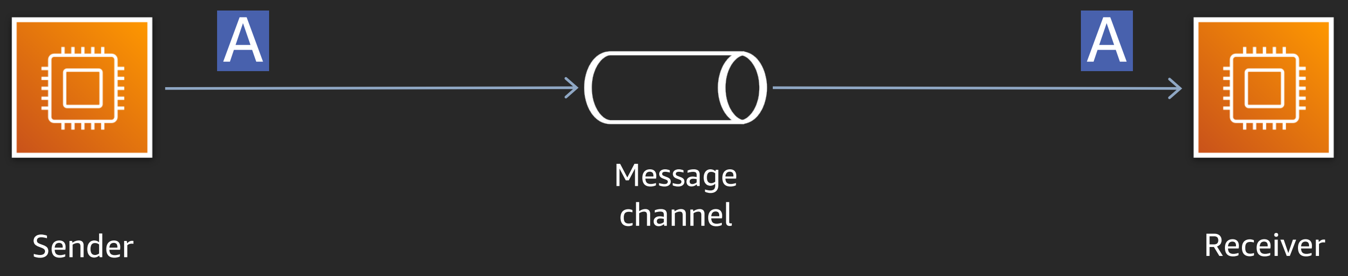
Message exchange

One-way

Request-response

Message exchange

One-way



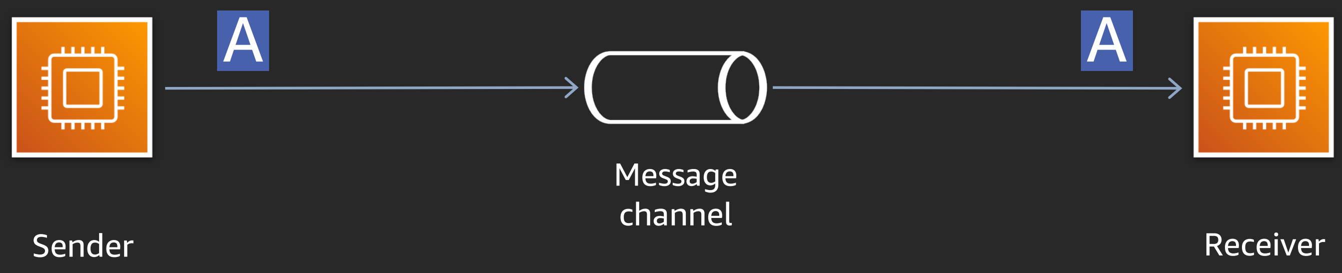
No response expected

Synchronous vs. fire-and-forget

Request-response

Message exchange

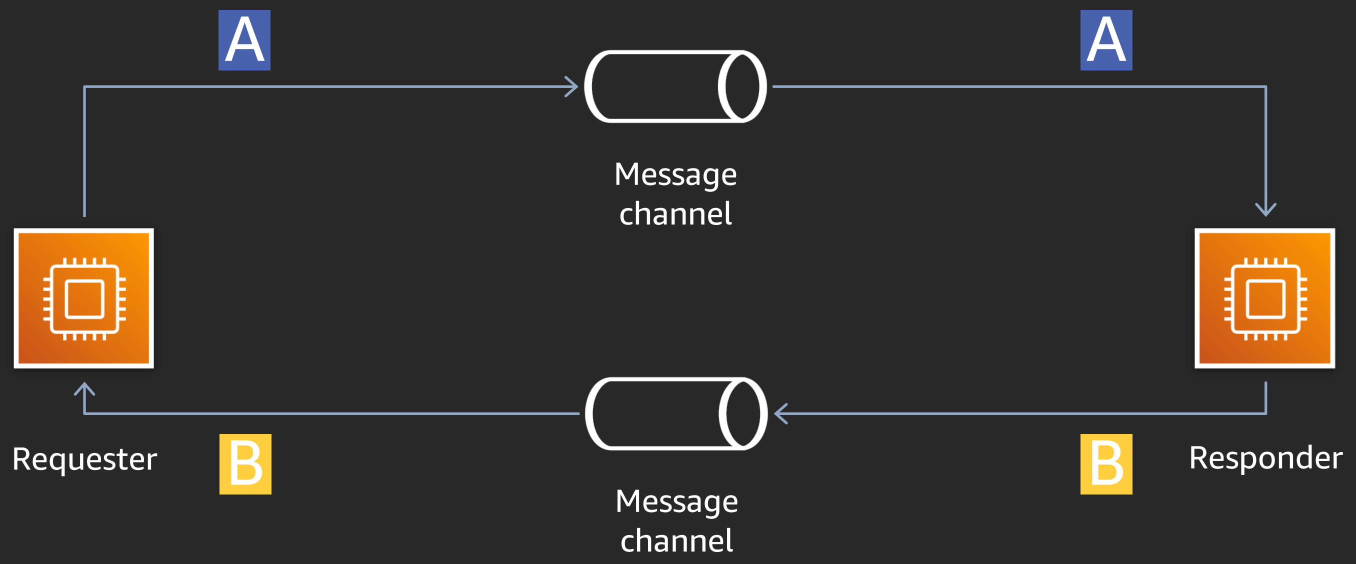
One-way



No response expected

Synchronous vs. fire-and-forget

Request-response



Response expected

Return address

Correlation ID

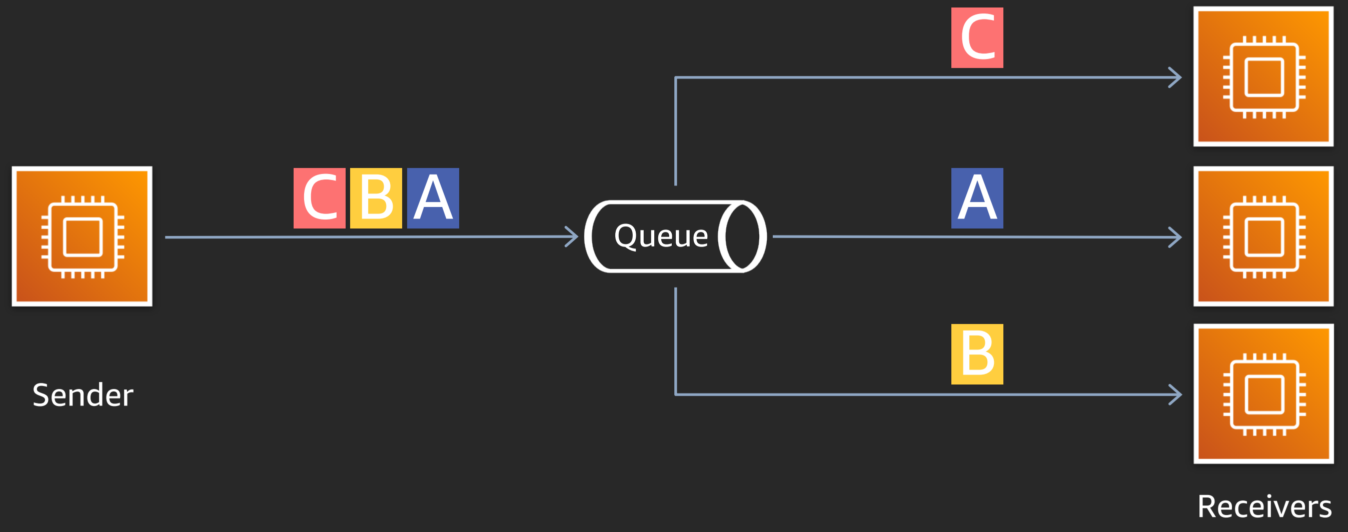
Message channels

Point-to-point (queue)

Publish-subscribe (topic)

Message channels

Point-to-point (queue)



Consumed by one receiver

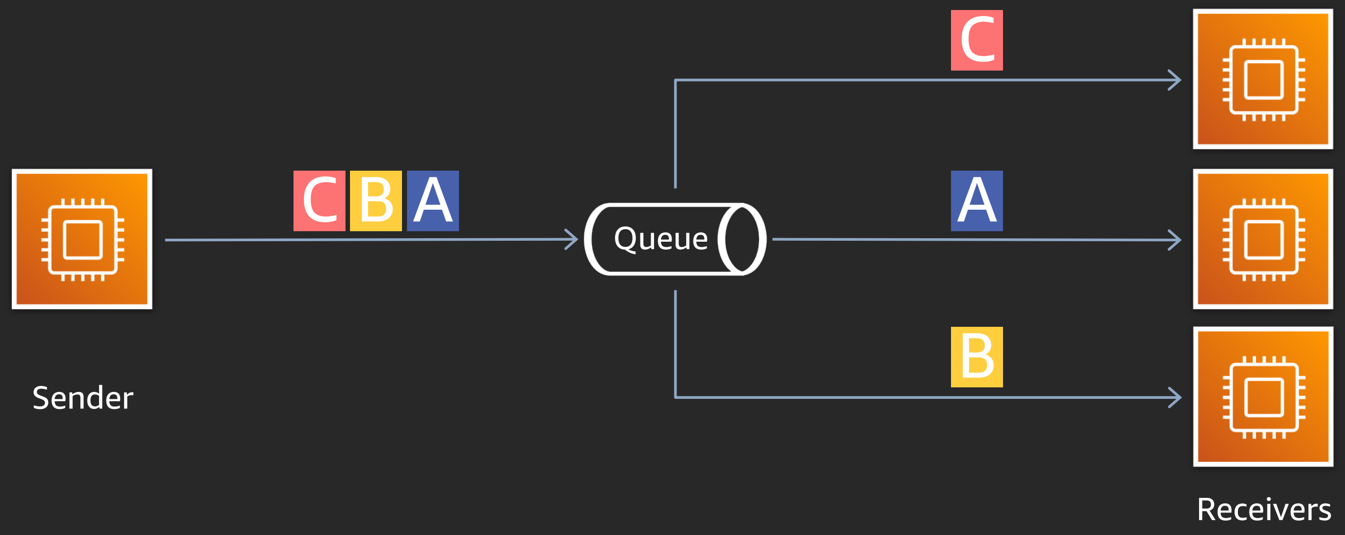
Easy to scale

Flatten peak loads

Publish-subscribe (topic)

Message channels

Point-to-point (queue)

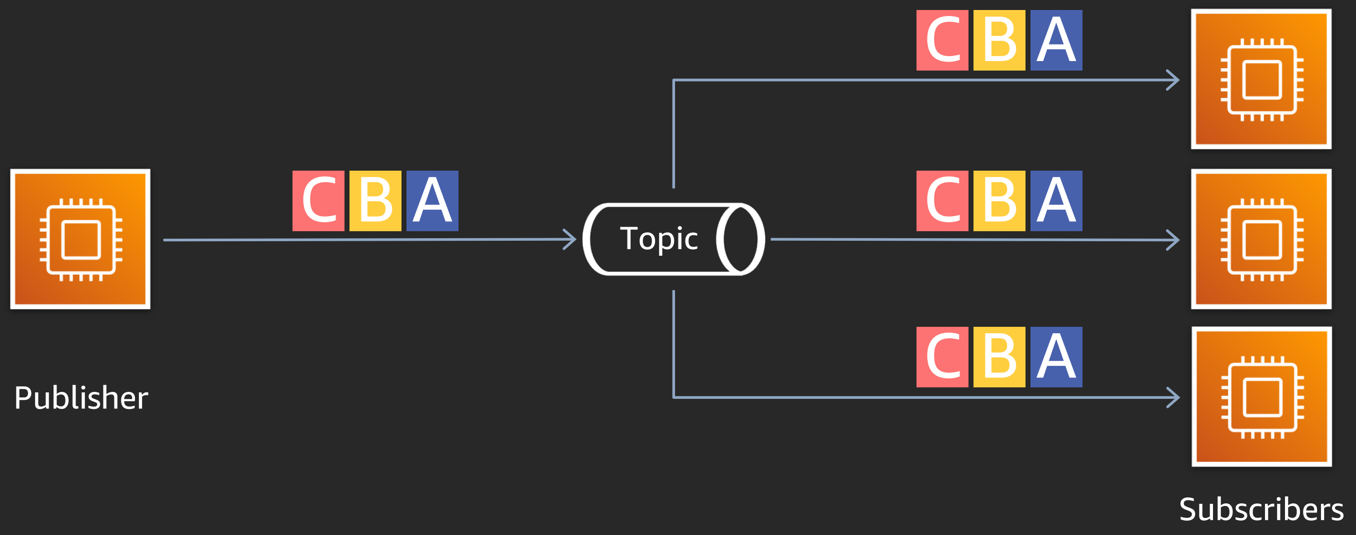


Consumed by one receiver

Easy to scale

Flatten peak loads

Publish-subscribe (topic)

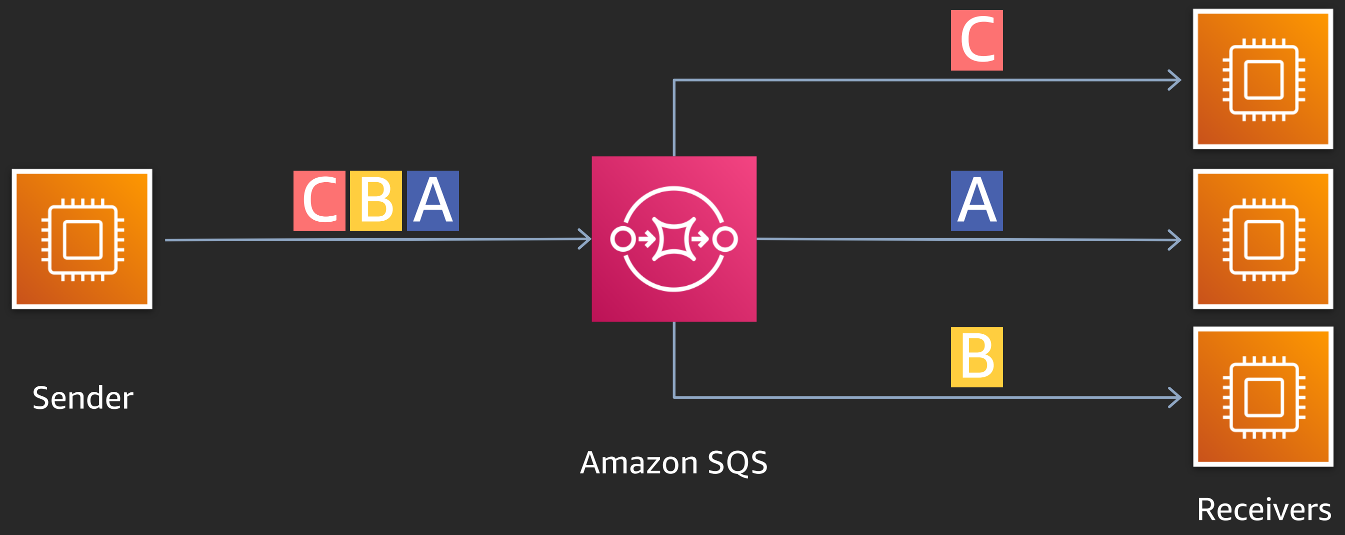


Consumed by all subscribers

Durable subscriber

Message channels

Point-to-point (queue)

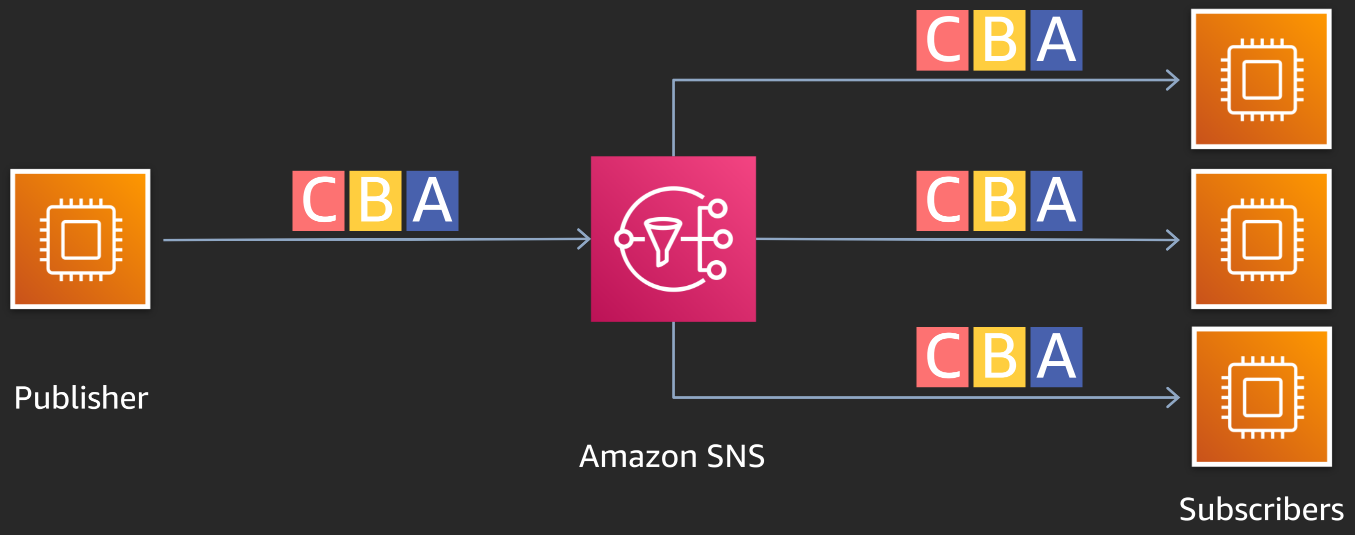


AWS service for queue functionality:

Amazon Simple Queue Services (Amazon SQS)

Serverless & cloud-native

Publish-subscribe (topic)



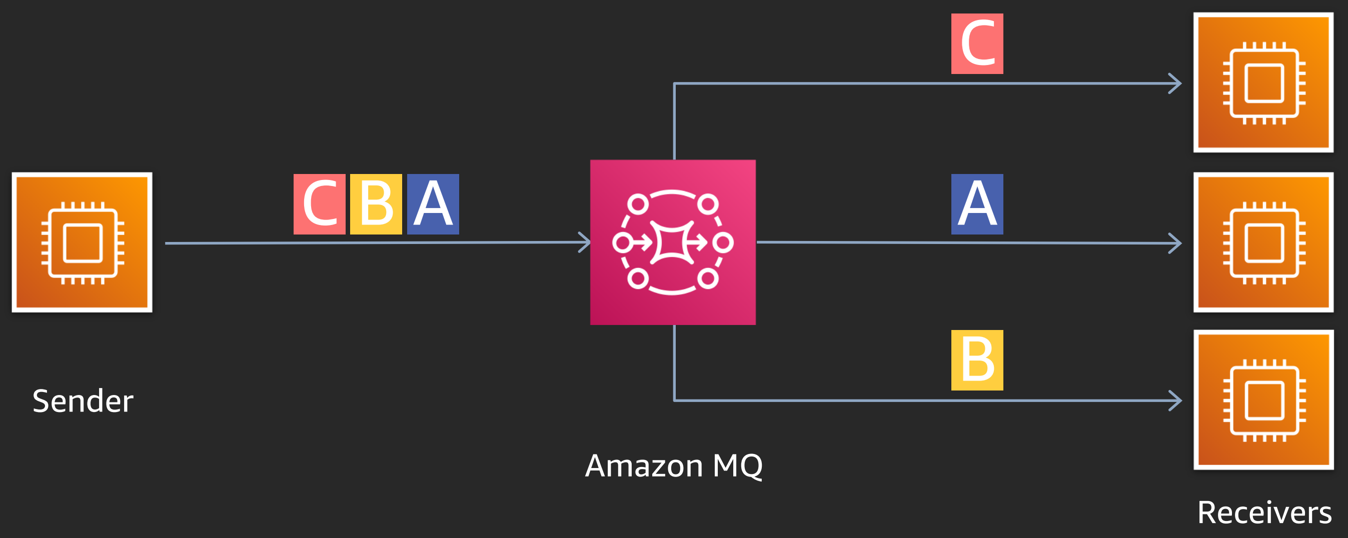
AWS service for topic functionality:

Amazon Simple Notification Service (Amazon SNS)

Serverless & cloud-native

Message channels

Point-to-point (queue)

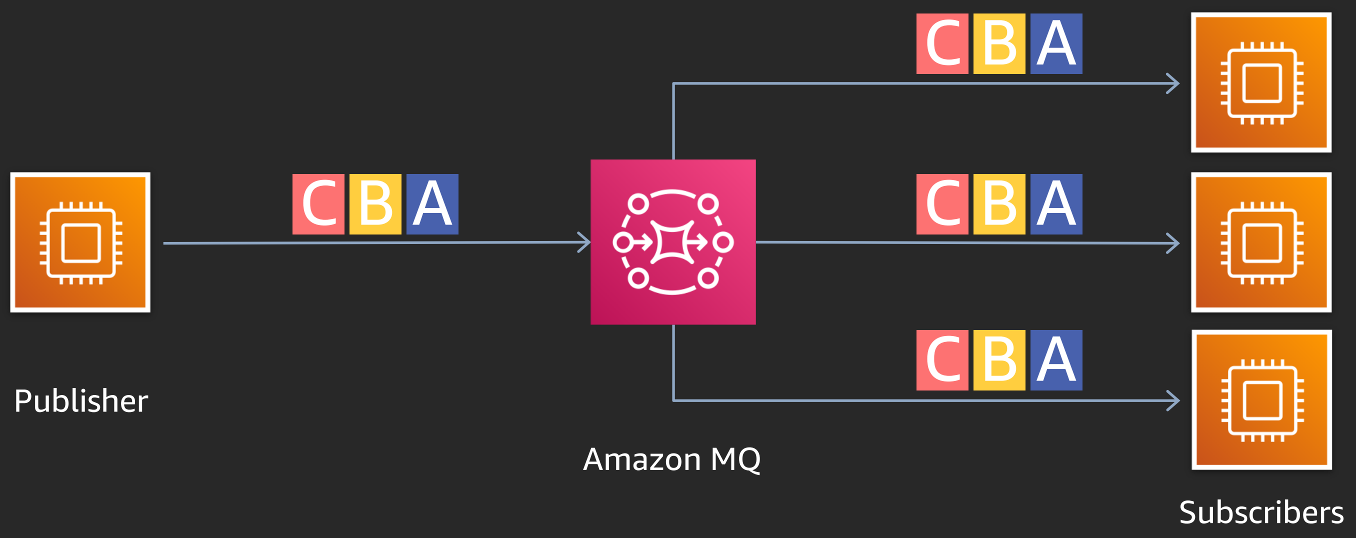


AWS service for queue functionality (non-serverless):

Amazon MQ (managed Apache Active MQ)

For apps constrained to protocols like JMS, AMQP, etc.

Publish-subscribe (topic)



AWS service for topic functionality (non-serverless):

Amazon MQ (managed Apache Active MQ)

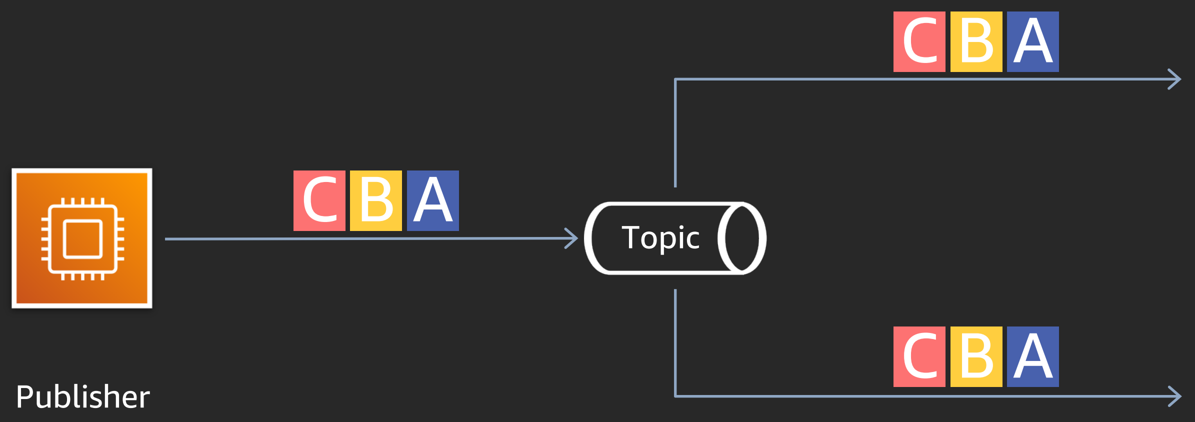
For apps constrained to protocols like JMS, AMQP, etc.

Message channels

Topic-queue-chaining

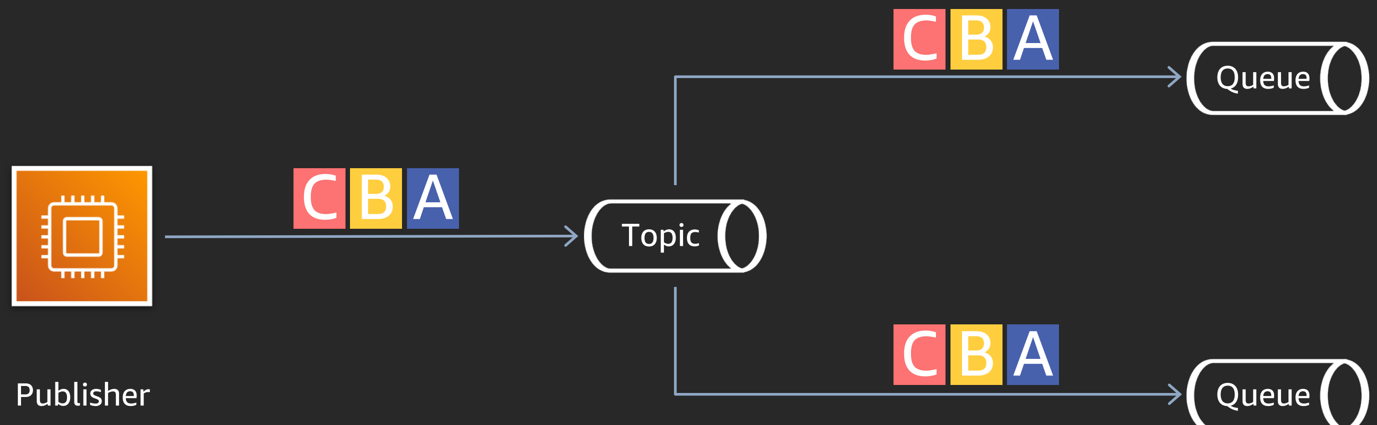
Message channels

Topic-queue-chaining



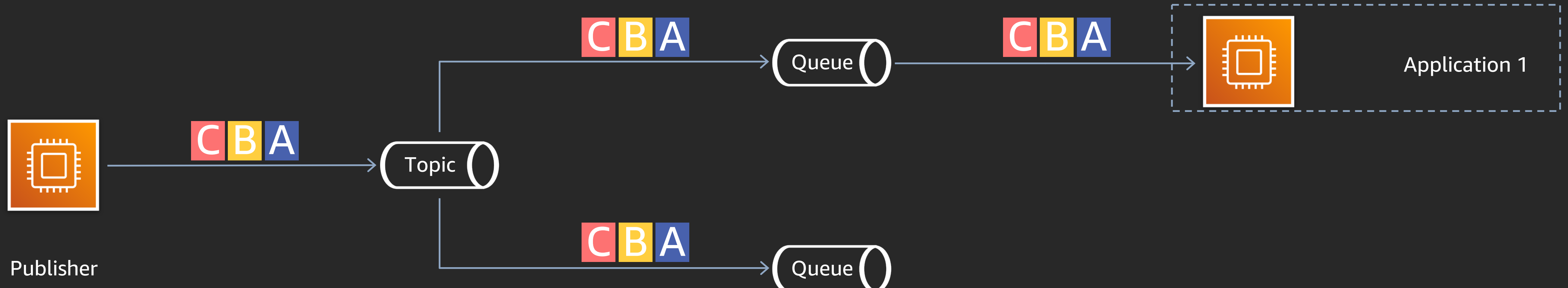
Message channels

Topic-queue-chaining



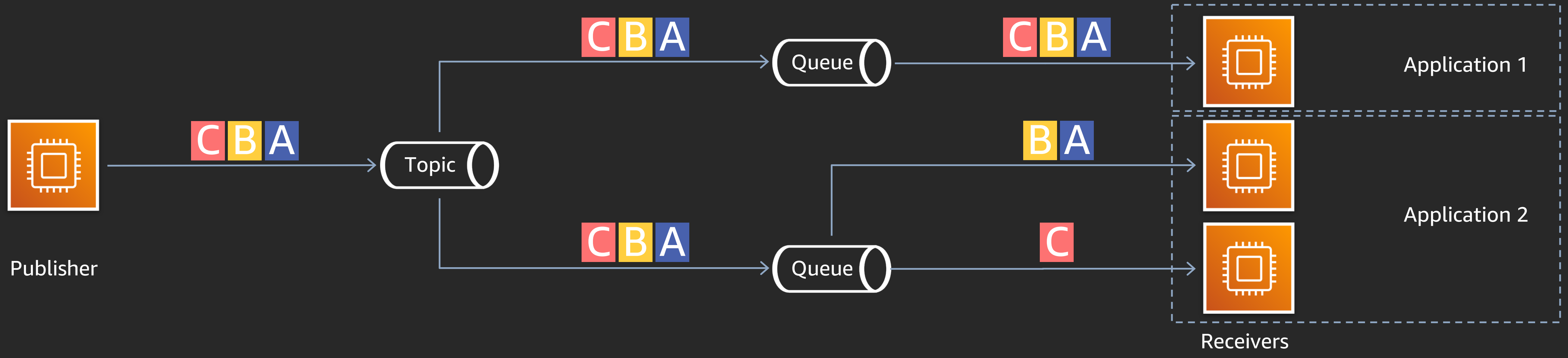
Message channels

Topic-queue-chaining



Message channels

Topic-queue-chaining



Allows fan-out and receiver scale-out at the same time

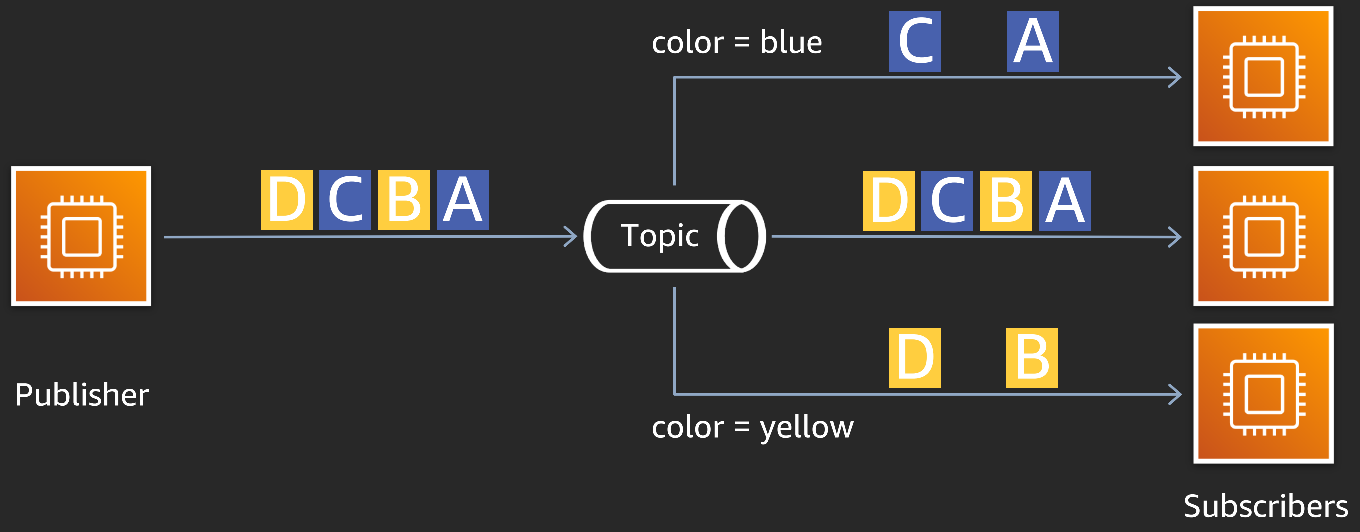
Message routing

Message filter

Recipient list

Message routing

Message filter



Receive only a relevant subset of messages

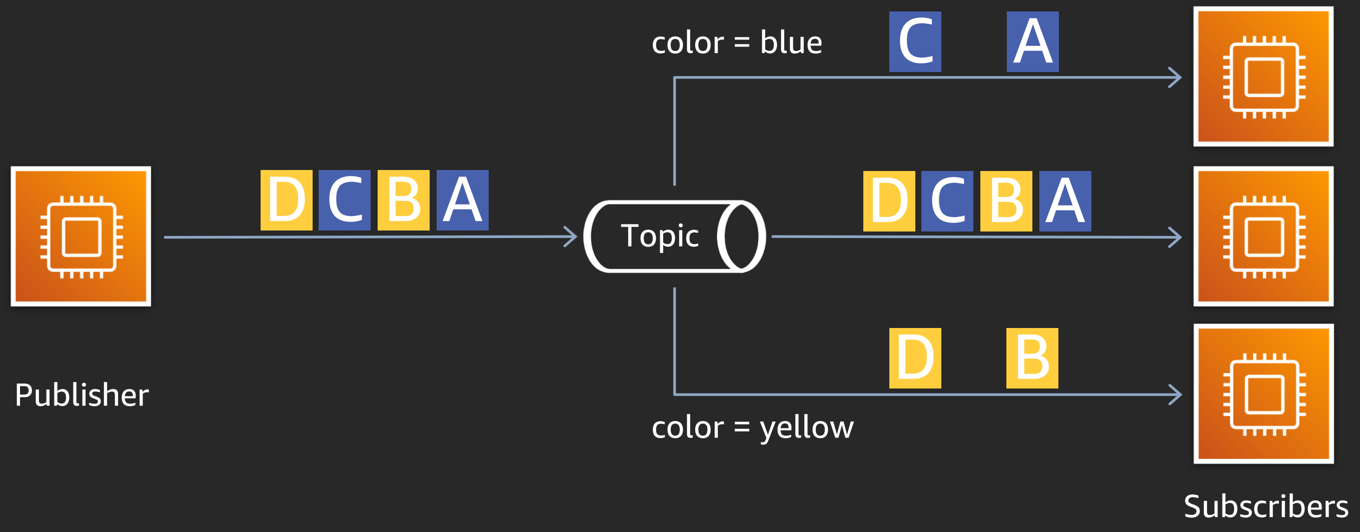
Controlled by subscriber

Publisher remains completely unaware

Recipient list

Message routing

Message filter

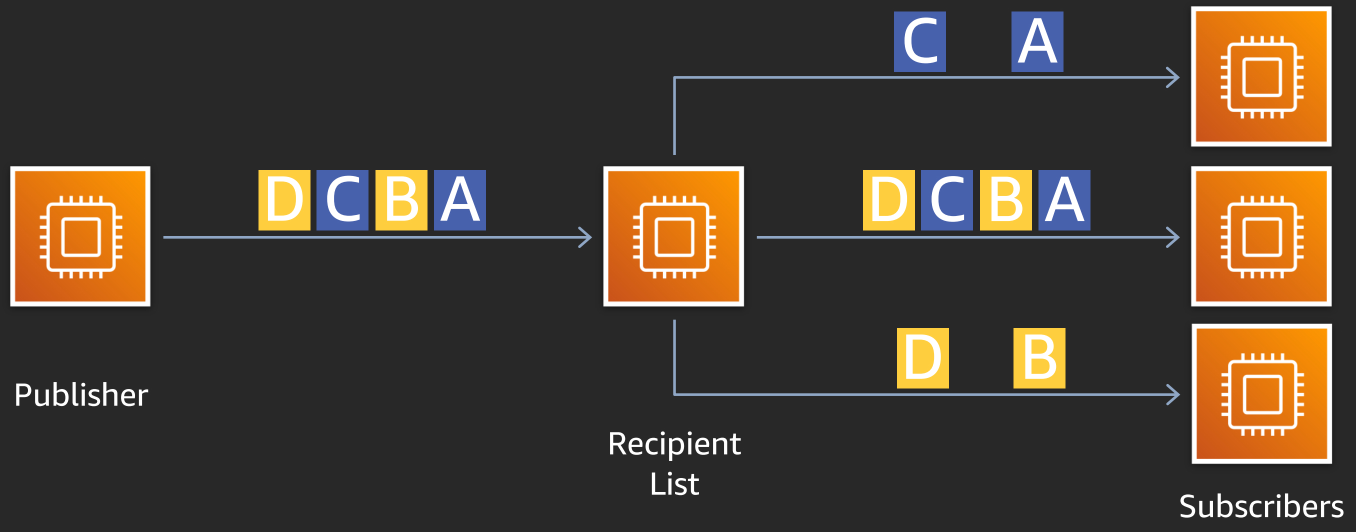


Receive only a relevant subset of messages

Controlled by subscriber

Publisher remains completely unaware

Recipient list



Send only a relevant subset of messages to a subscriber

Controlled by publisher or separate component

Potentially adds coupling

Message routing

Scatter-gather

Message routing

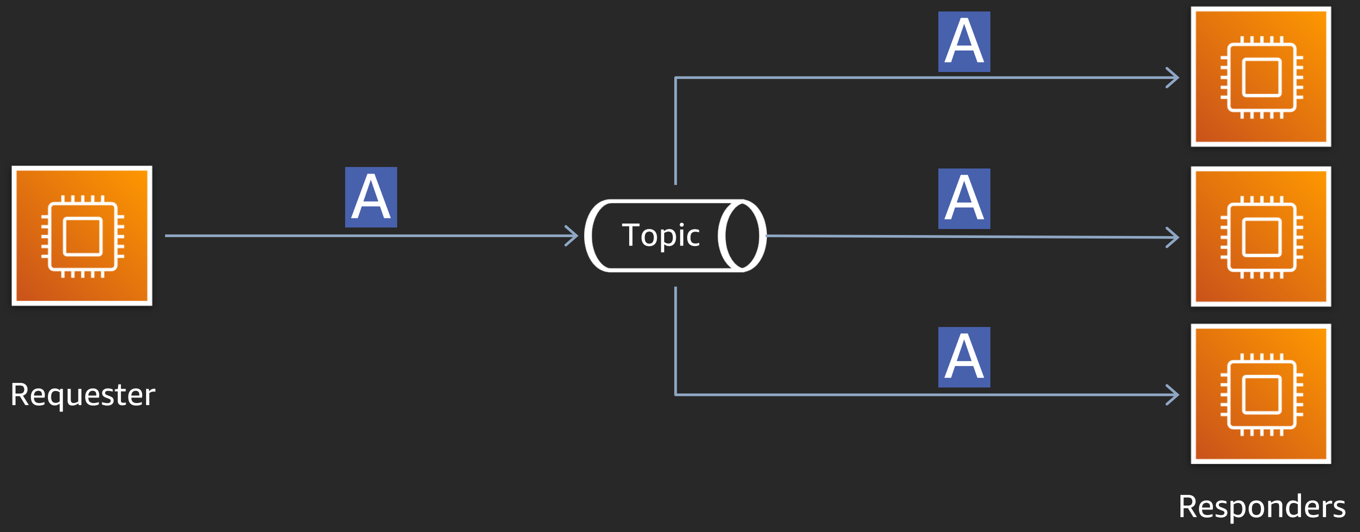
Scatter-gather

How to distribute a request across potentially interested/relevant parties and capture their individual responses?

- RFQ scenarios, or search for best response
- Parallel processing scenarios; for example, divide and conquer

Message routing

Scatter-gather

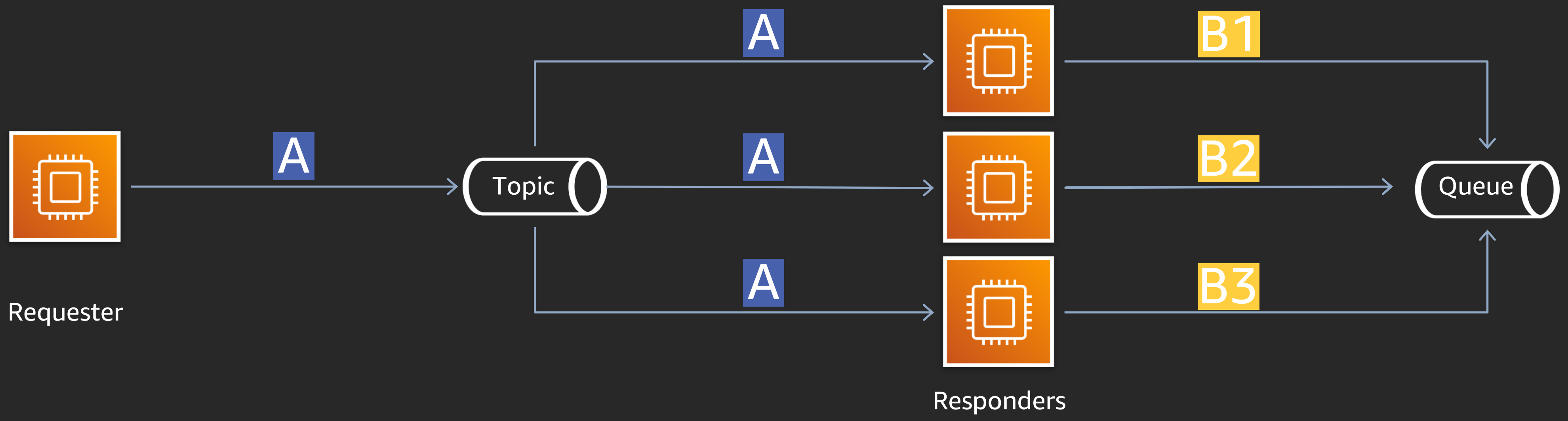


How to distribute a request across potentially interested/relevant parties and capture their individual responses?

- RFQ scenarios, or search for best response
- Parallel processing scenarios; for example, divide and conquer

Message routing

Scatter-gather

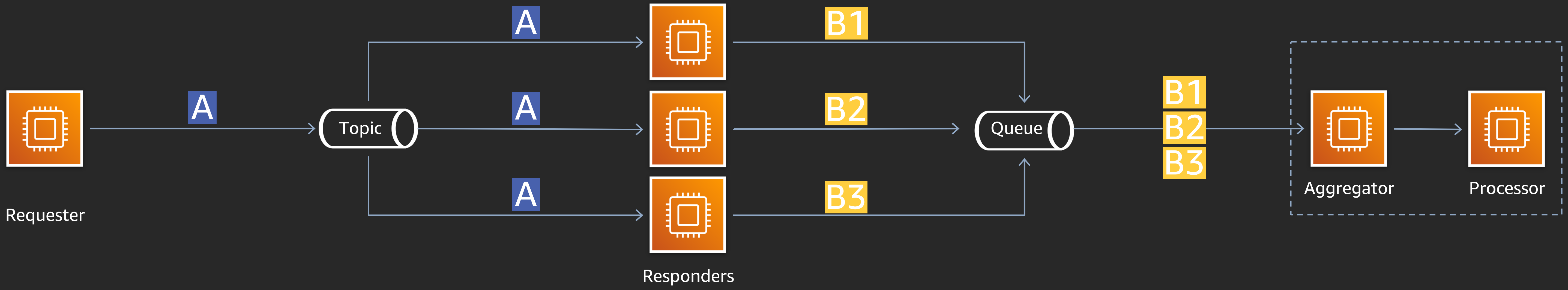


How to distribute a request across potentially interested/relevant parties and capture their individual responses?

- RFQ scenarios, or search for best response
- Parallel processing scenarios; for example, divide and conquer

Message routing

Scatter-gather



How to distribute a request across potentially interested/relevant parties and capture their individual responses?

- RFQ scenarios, or search for best response
- Parallel processing scenarios; for example, divide and conquer

Message routing

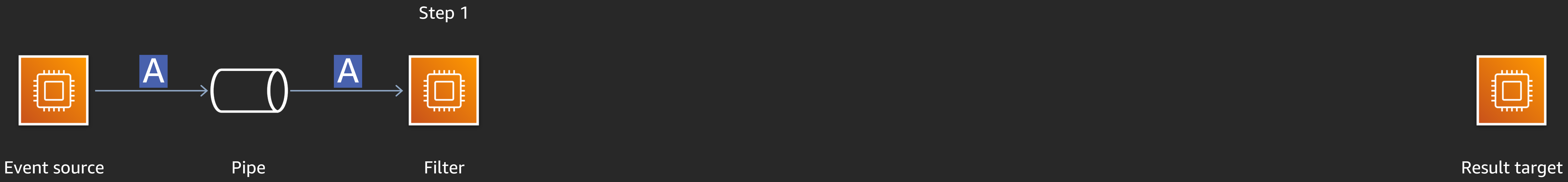
Pipes and filters

Message routing

Pipes and filters

Message routing

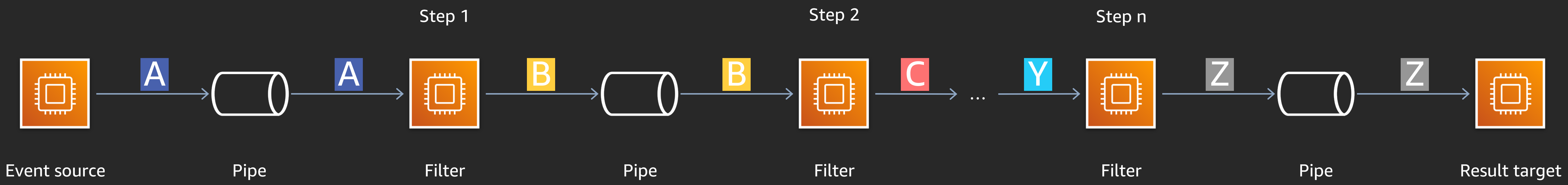
Pipes and filters



Event triggers chain of processing steps (“filters”)

Message routing

Pipes and filters

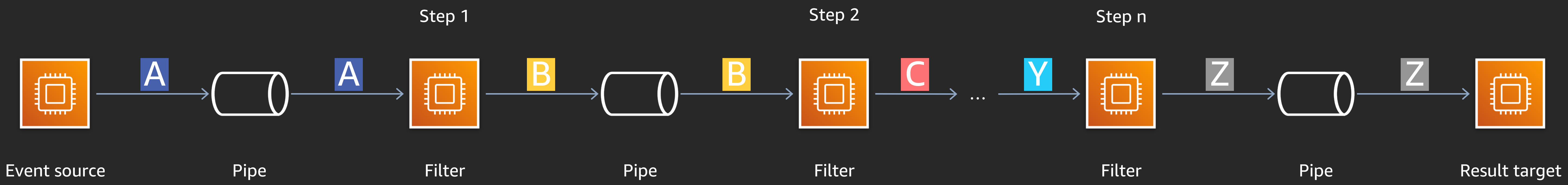


Event triggers chain of processing steps (“filters”)

Knowledge of destination for next step(s) is wired into each filter

Message routing

Pipes and filters



Event triggers chain of processing steps (“filters”)

Knowledge of destination for next step(s) is wired into each filter

Similar patterns: Chain of responsibility, processing pipeline, saga choreography

Message routing

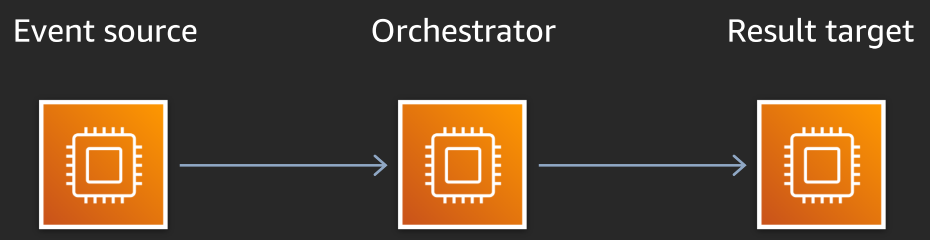
Saga orchestration

Message routing

Saga orchestration

Message routing

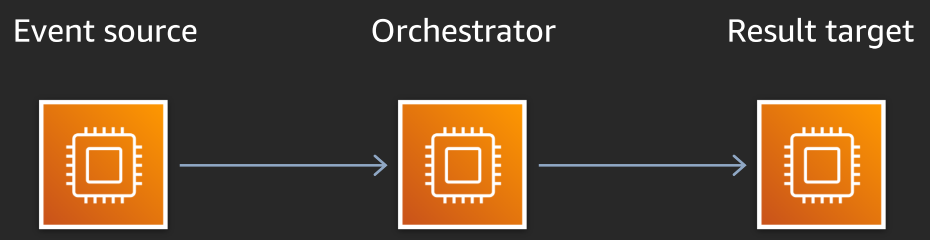
Saga orchestration



Event triggers orchestrated workflow

Message routing

Saga orchestration

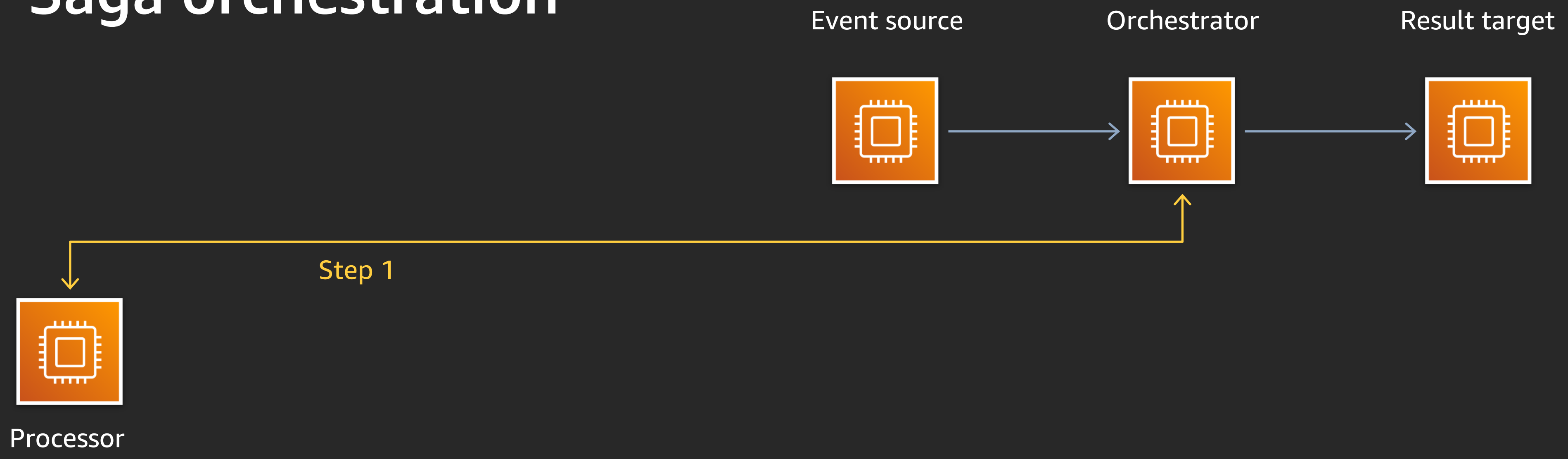


Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component, as well as for potential rollback

Message routing

Saga orchestration



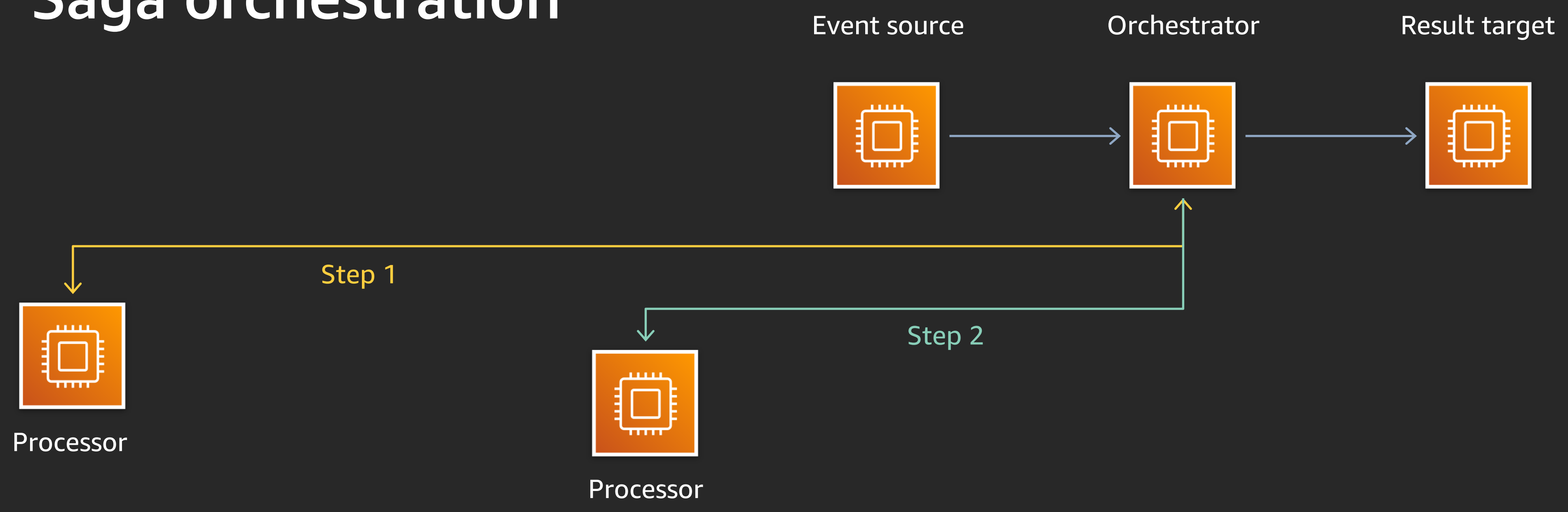
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component, as well as for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing

Saga orchestration



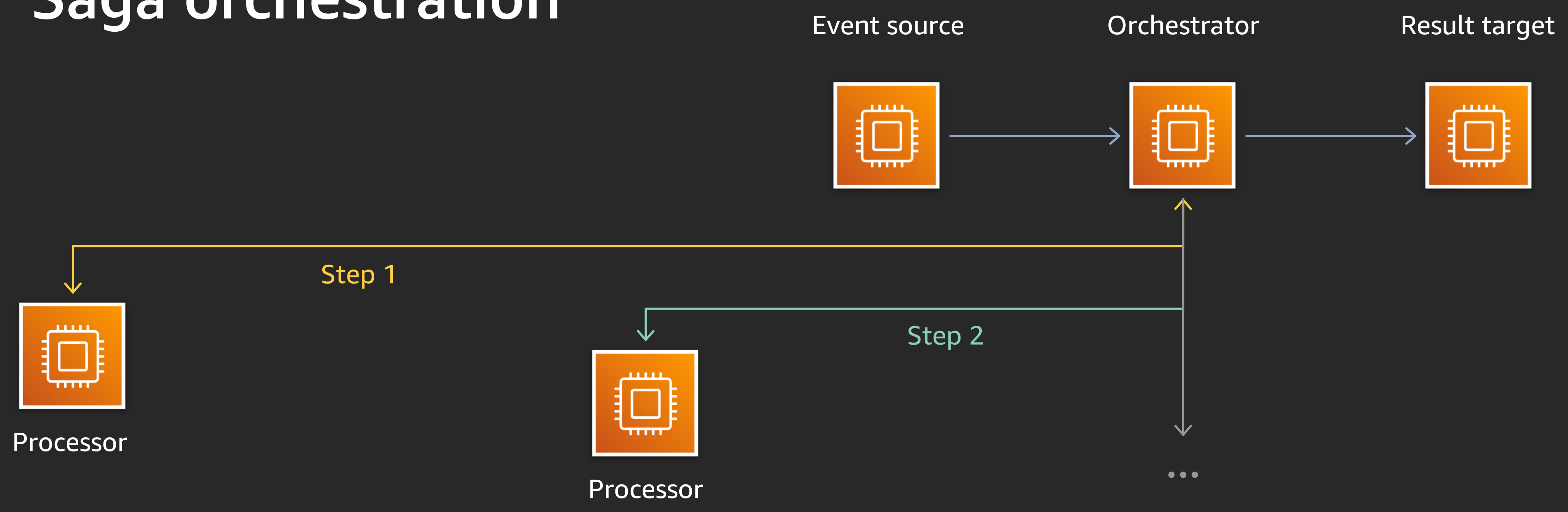
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component, as well as for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing

Saga orchestration



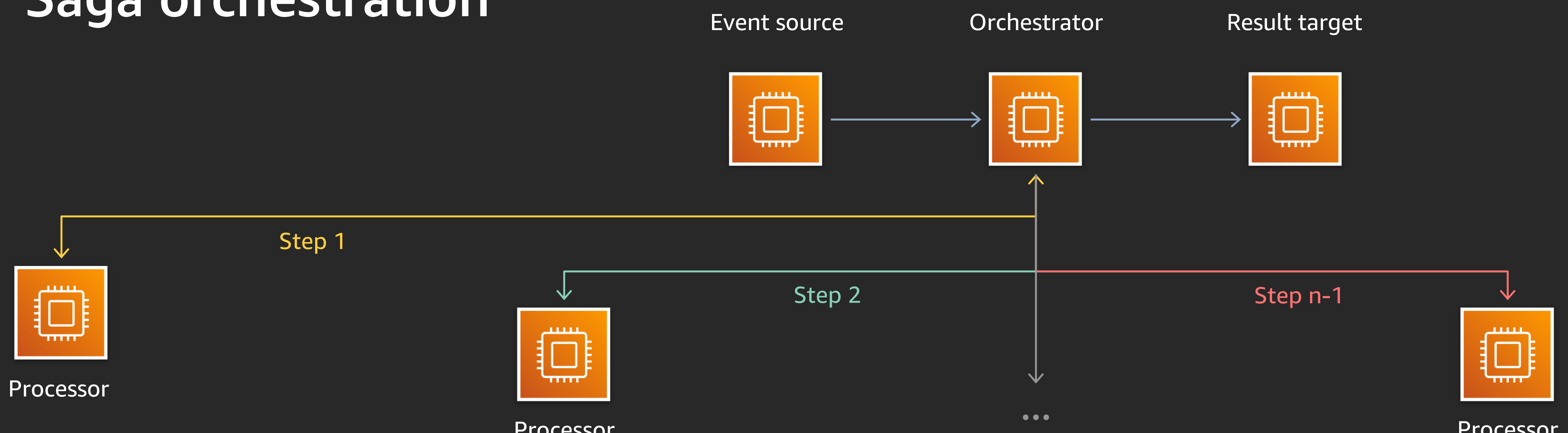
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component, as well as for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing

Saga orchestration



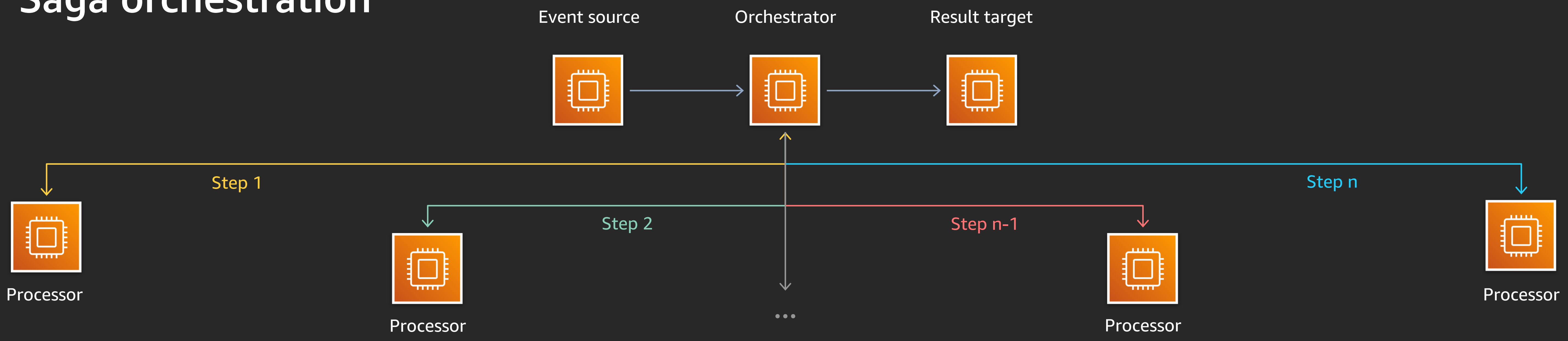
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component, as well as for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing

Saga orchestration



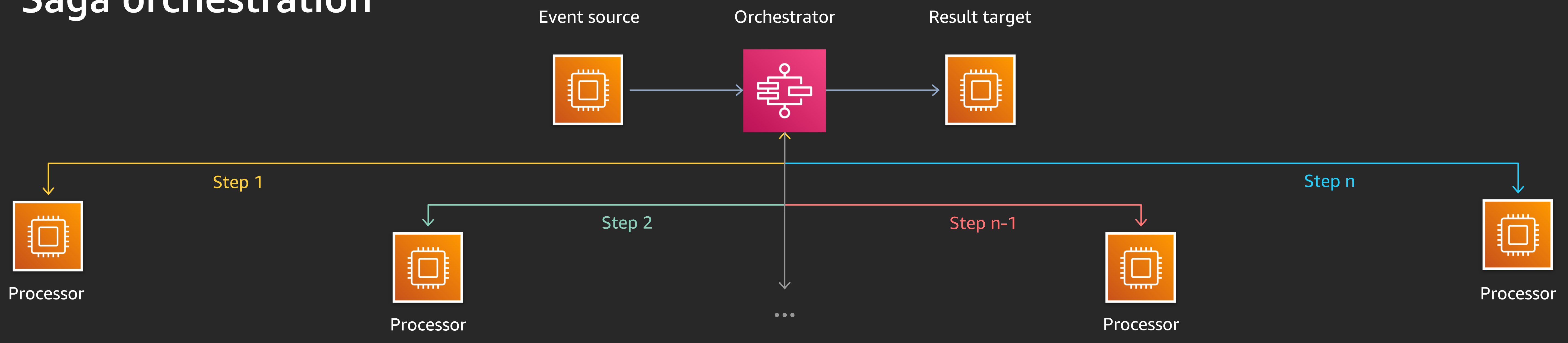
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component, as well as for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing

Saga orchestration



AWS service for saga orchestration (serverless):

AWS Step Functions

Concrete use cases and labs for today

Context: Wild Rydes, Inc.

Choose your path

We have four labs for you today, plus a common foundation lab

After intro of use cases, context, and patterns, you can pick the most relevant labs for you

We will summarize the labs again for you afterward

Photo: Dirk Fröhner

Choose your path

Photo: Dirk Fröhner

Choose your path

Foundation

Lab 0

Photo: Dirk Fröhner

Choose your path

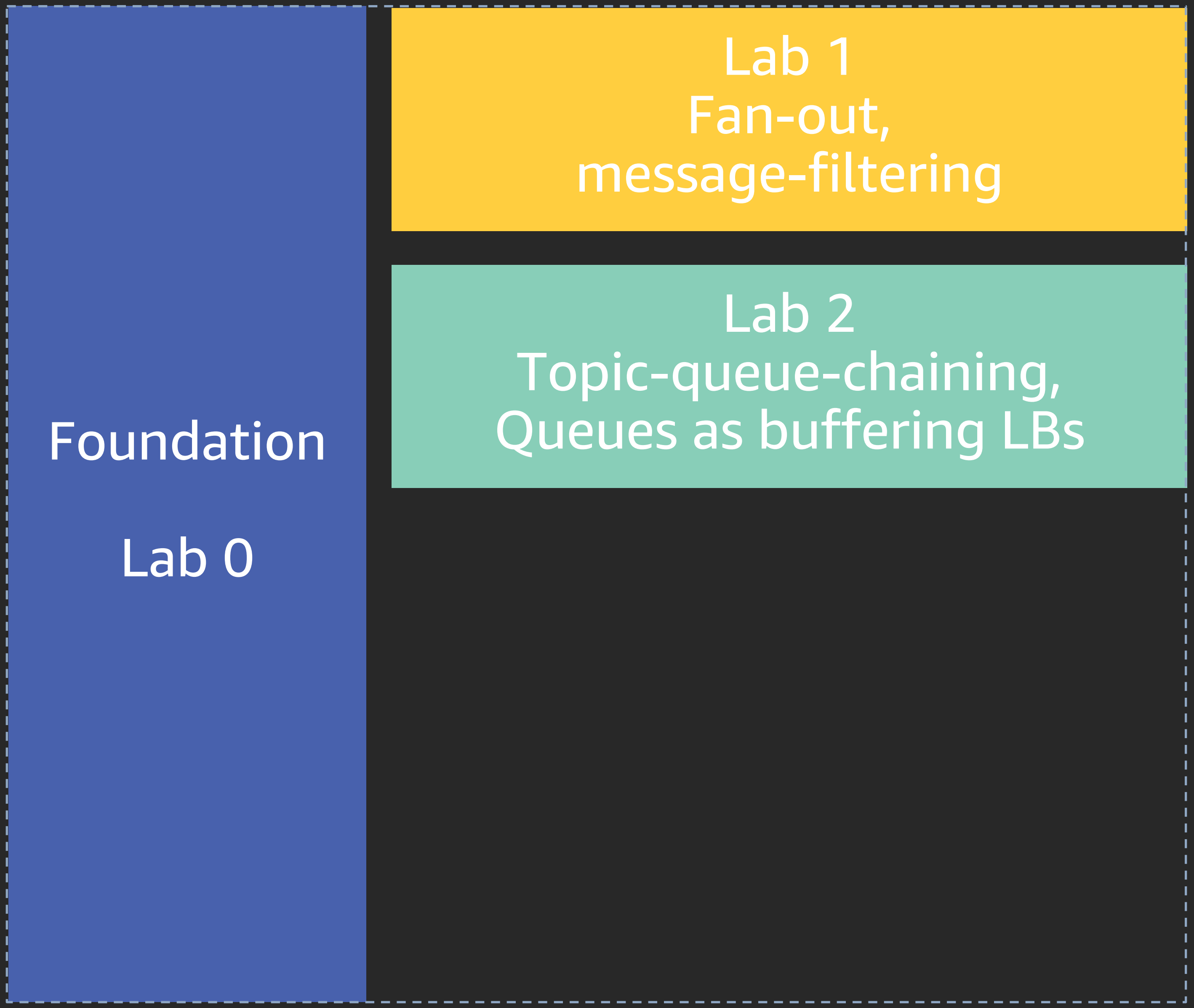
Foundation

Lab 0

Lab 1
Fan-out,
message-filtering

Photo: Dirk Fröhner

Choose your path



Foundation

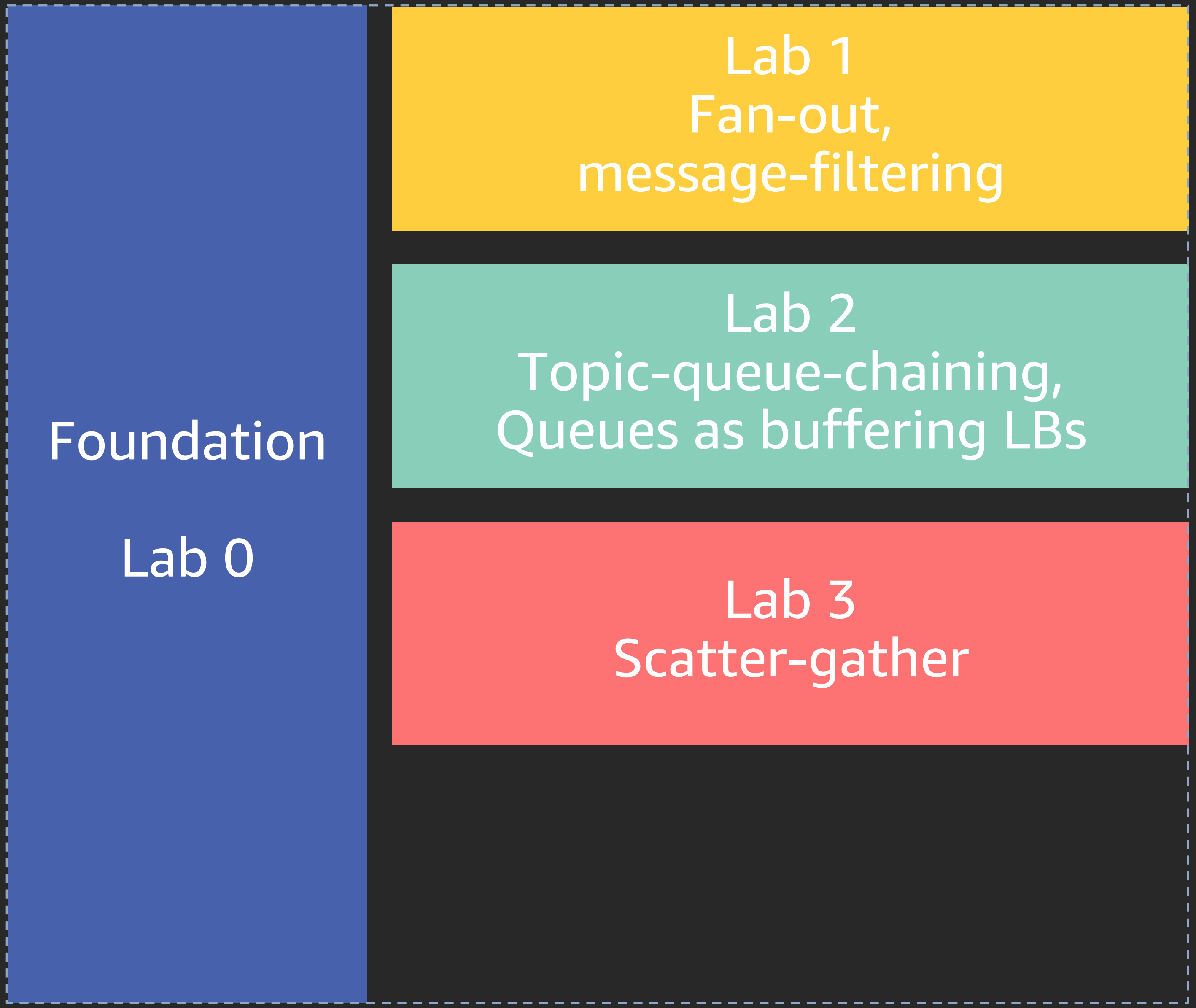
Lab 0

Lab 1
Fan-out,
message-filtering

Lab 2
Topic-queue-chaining,
Queues as buffering LBs

Photo: Dirk Fröhner

Choose your path



Foundation

Lab 0

Lab 1
Fan-out,
message-filtering

Lab 2
Topic-queue-chaining,
Queues as buffering LBs

Lab 3
Scatter-gather

Photo: Dirk Fröhner

Choose your path

Foundation

Lab 0

Lab 1
Fan-out,
message-filtering

Lab 2
Topic-queue-chaining,
Queues as buffering LBs

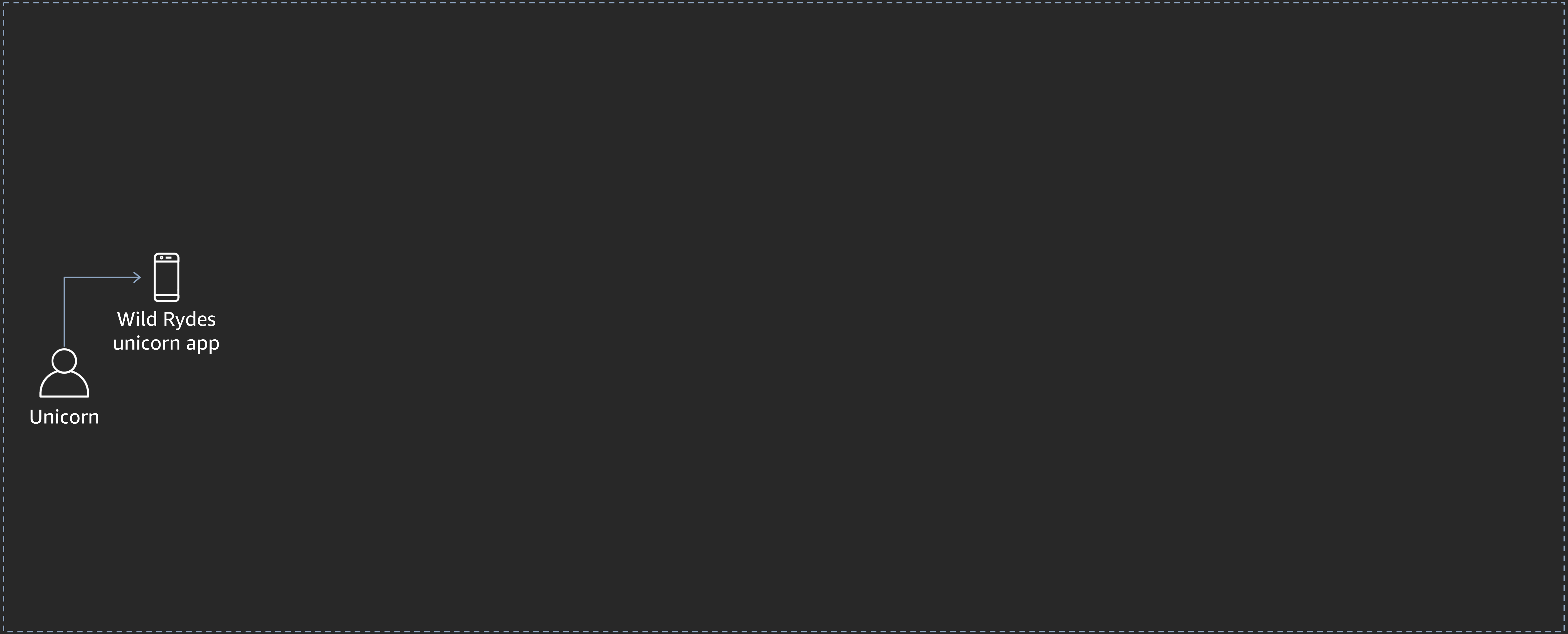
Lab 3
Scatter-gather

Lab 4
Saga orchestration

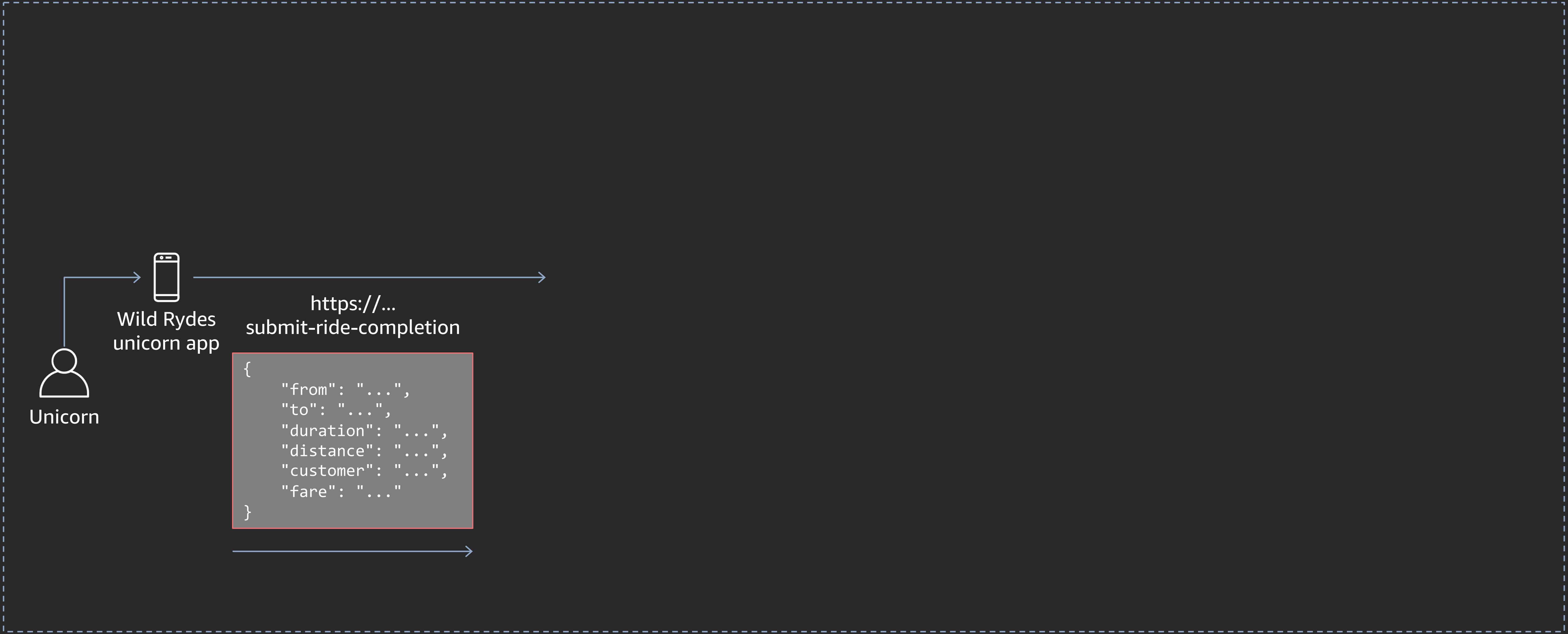
Photo: Dirk Fröhner

Use case: Submit a ride completion Context for labs 1 + 2

Use case: Submit a ride completion

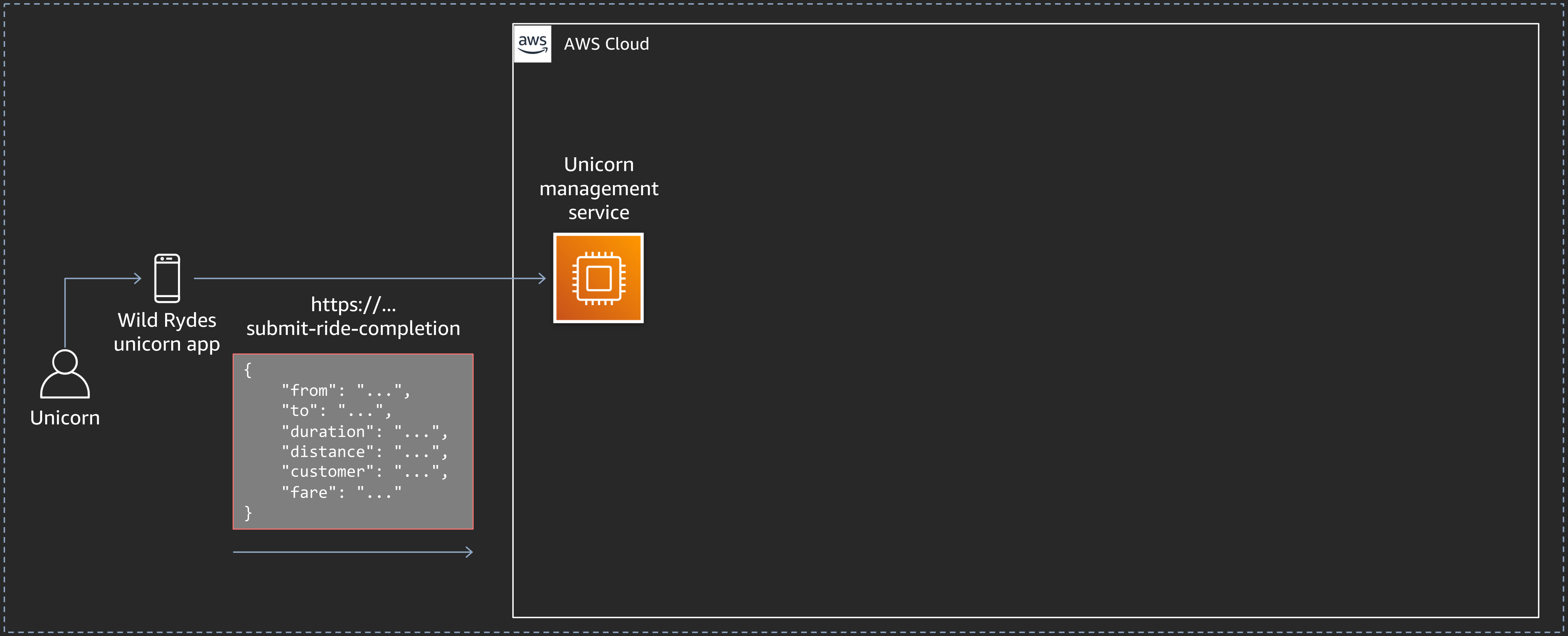


Use case: Submit a ride completion



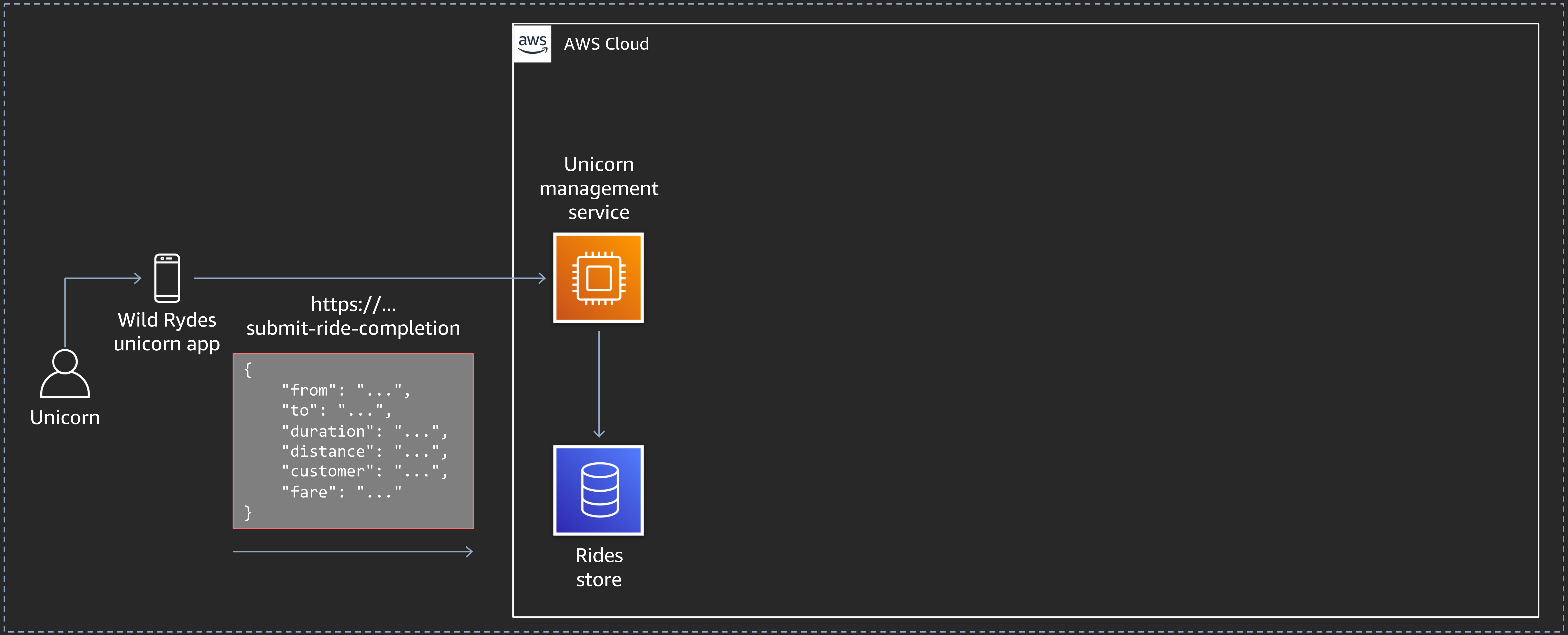
```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion



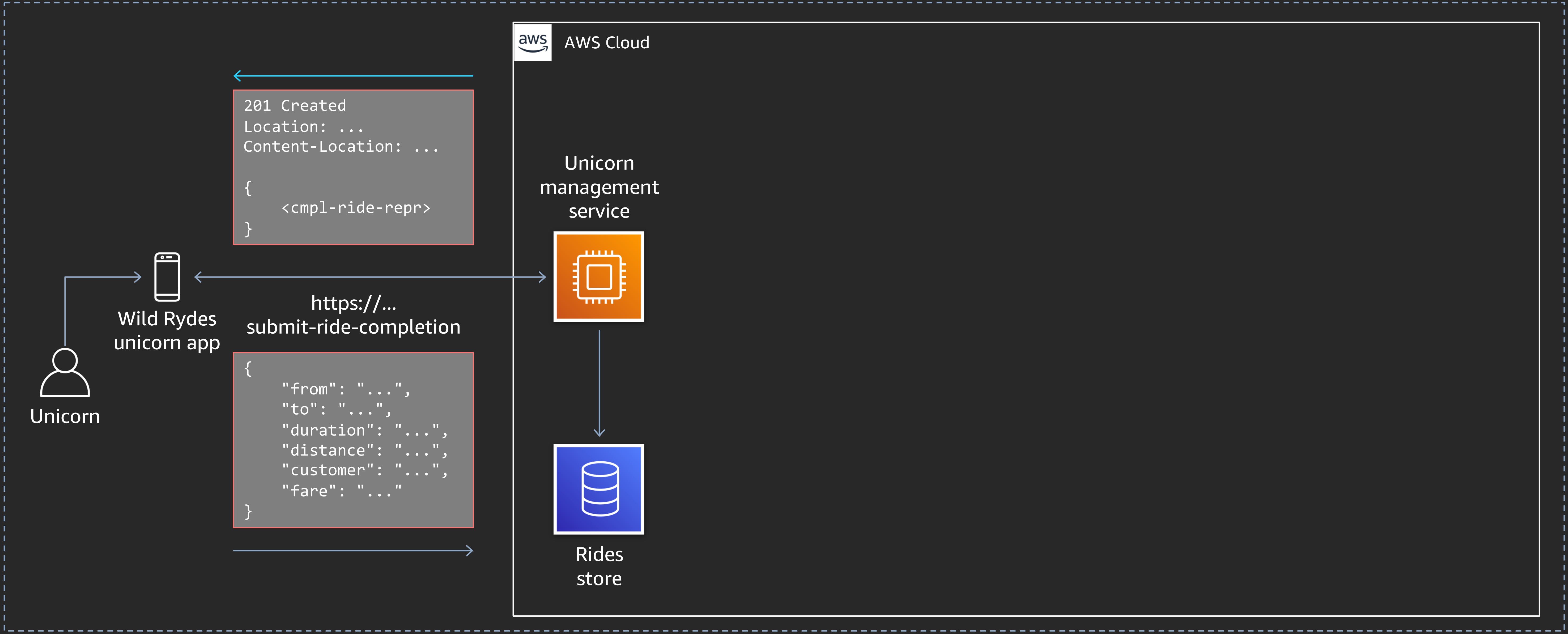
```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion



```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

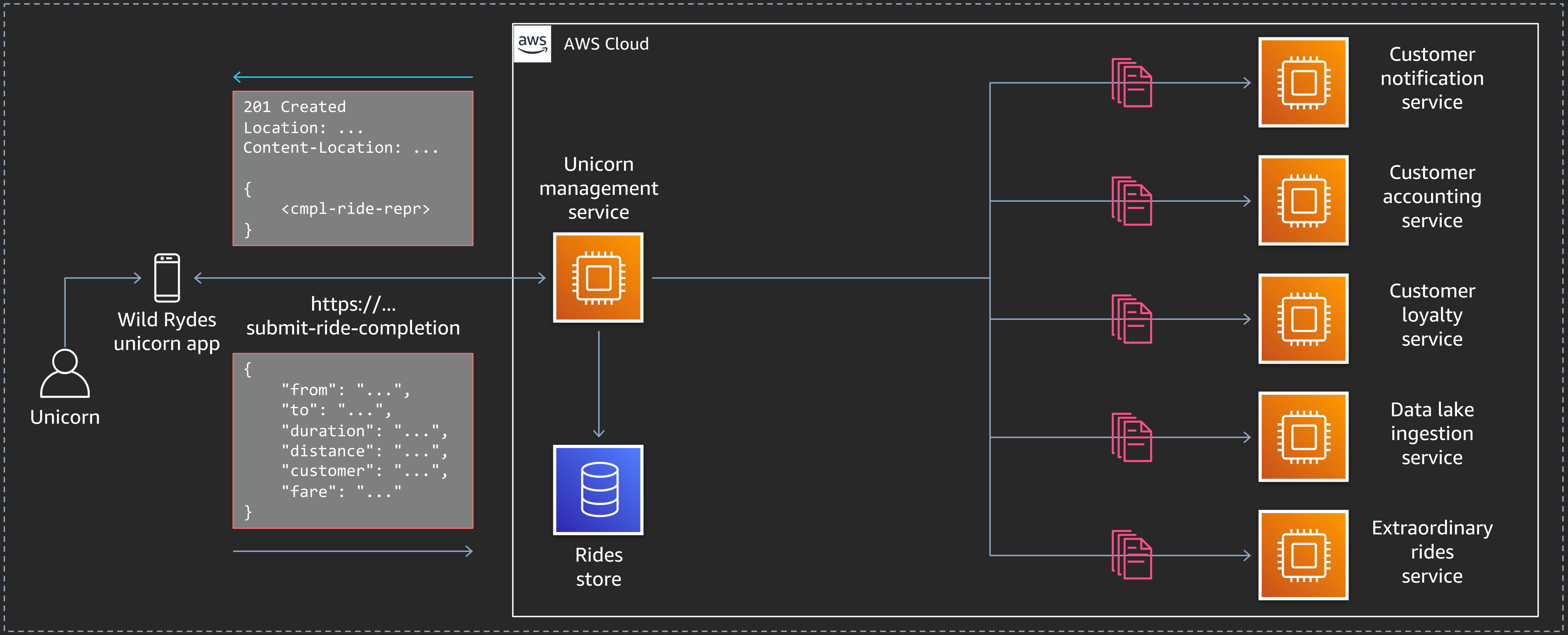
Use case: Submit a ride completion


AWS Cloud
Unicorn management service
Rides store
Wild Rydes unicorn app
Unicorn
https://... submit-ride-completion

```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

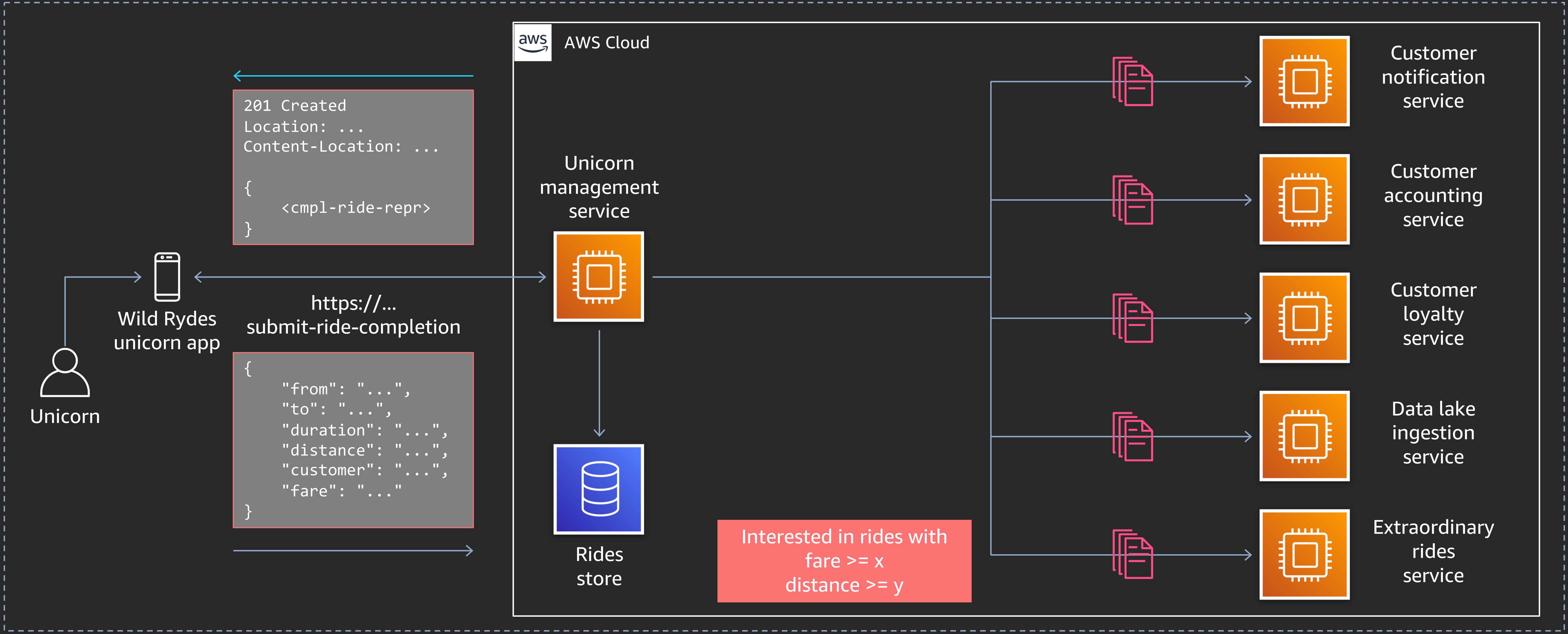
Use case: Submit a ride completion


AWS Cloud
Unicorn
Wild Rydes unicorn app
https://...
submit-ride-completion
Unicorn management service
Rides store
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

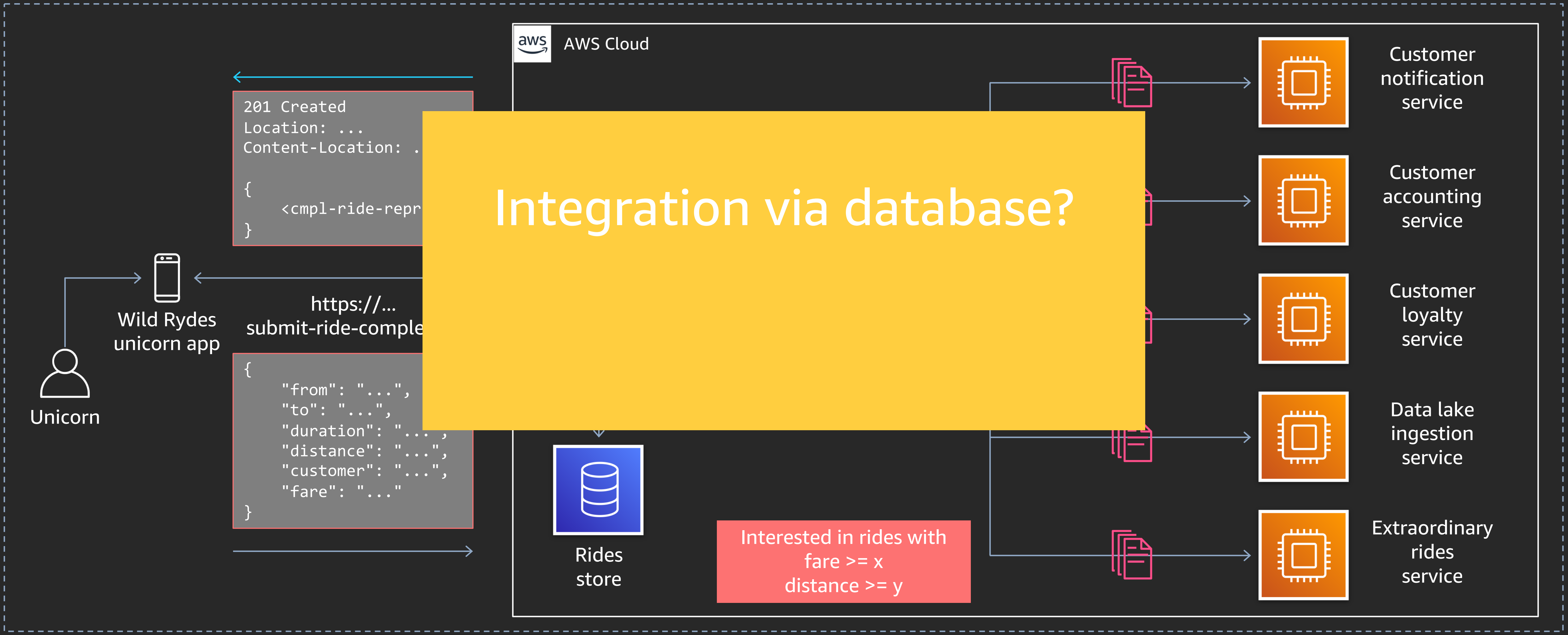
Use case: Submit a ride completion



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion



AWS Cloud
Unicorn
Wild Rydes unicorn app
https://...
submit-ride-comple
Rides store
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

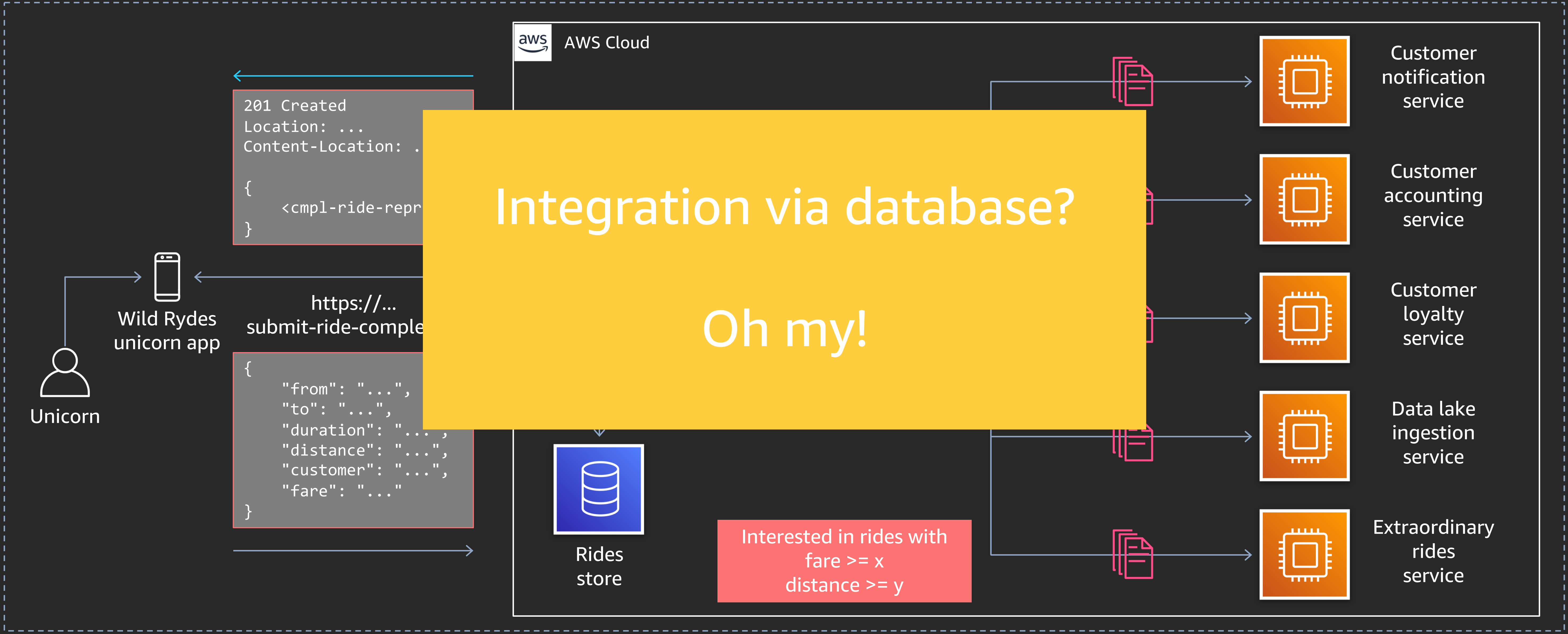
```
201 Created
Location: ...
Content-Location: .

{
    <cmpl-ride-repr
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Interested in rides with
fare >= x
distance >= y

Integration via database?

Use case: Submit a ride completion



```
201 Created
Location: ...
Content-Location: .

{
    <cmpl-ride-repr
}
```

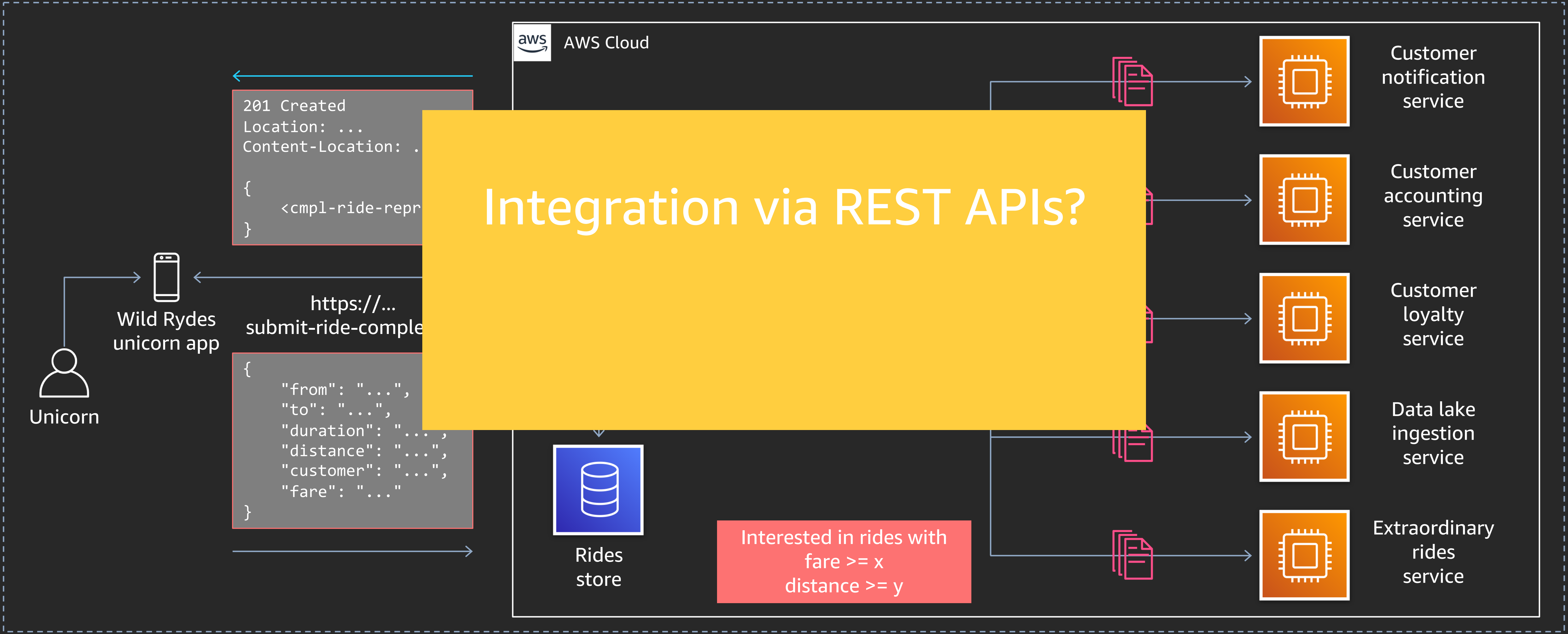
```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Interested in rides with
fare >= x
distance >= y

Integration via database?

Oh my!

Use case: Submit a ride completion



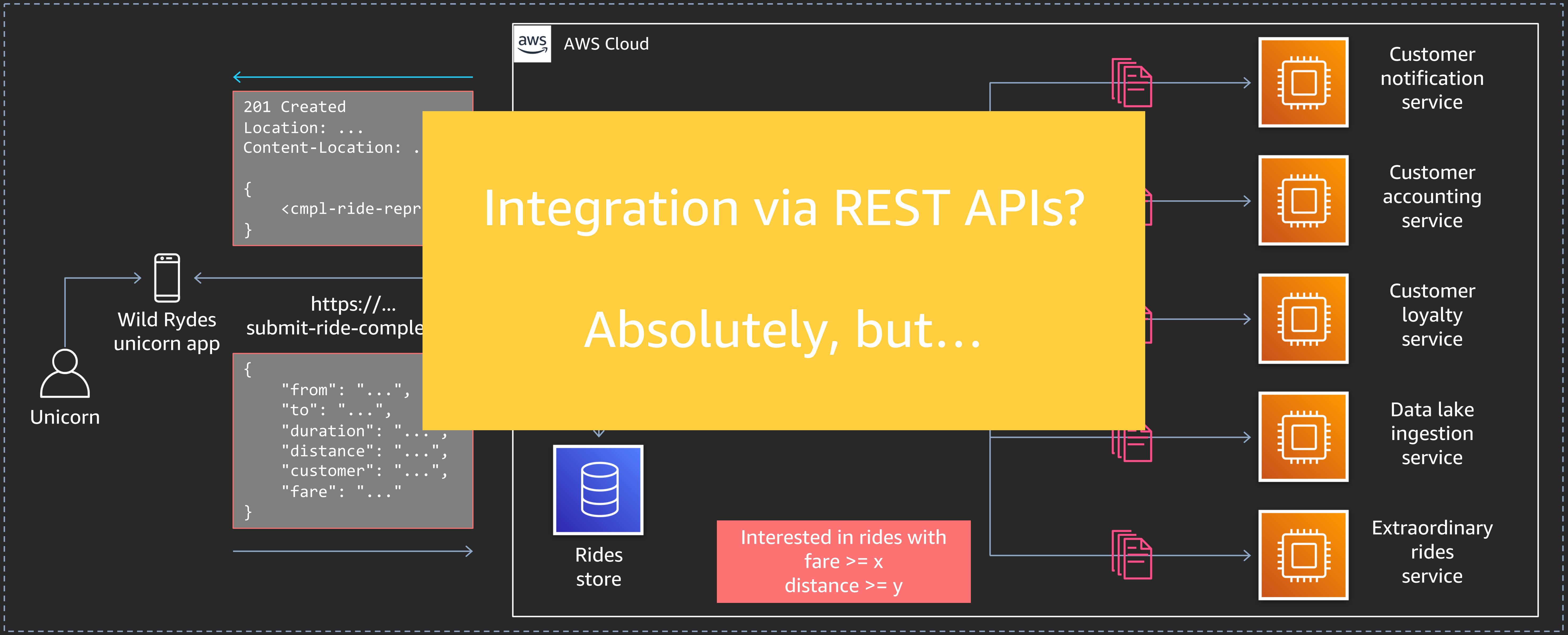
```
201 Created
Location: ...
Content-Location: .
{
    <cmpl-ride-repr
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

AWS Cloud
Unicorn
Wild Rydes unicorn app
https://...
submit-ride-comple
Rides store
Interested in rides with fare >= x distance >= y
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

Integration via REST APIs?

Use case: Submit a ride completion



```
201 Created
Location: ...
Content-Location: .

{
    <cmpl-ride-repr
}
```

Wild Rydes unicorn app

Unicorn

https://...
submit-ride-comple

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

AWS Cloud

Rides store

Interested in rides with
fare >= x
distance >= y

Customer notification service

Customer accounting service

Customer loyalty service

Data lake ingestion service

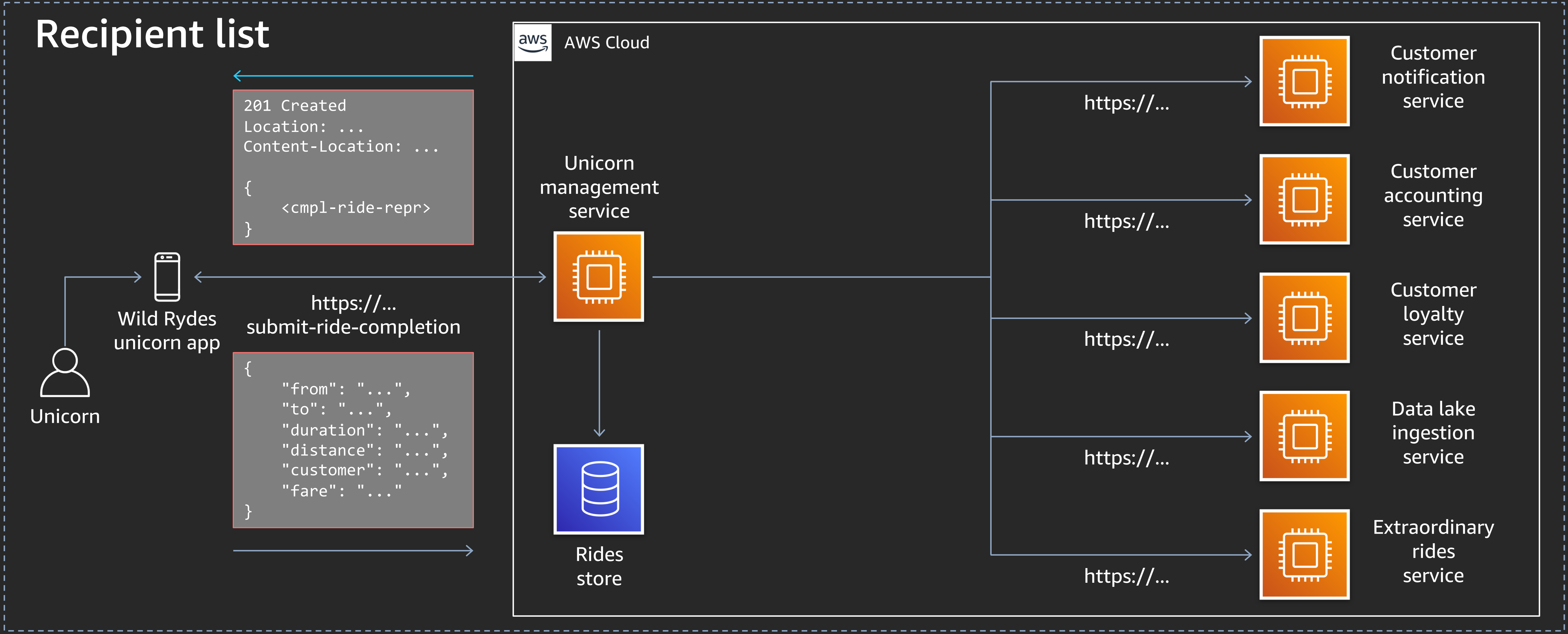
Extraordinary rides service

Integration via REST APIs?

Absolutely, but…

Use case: Submit a ride completion

Recipient list

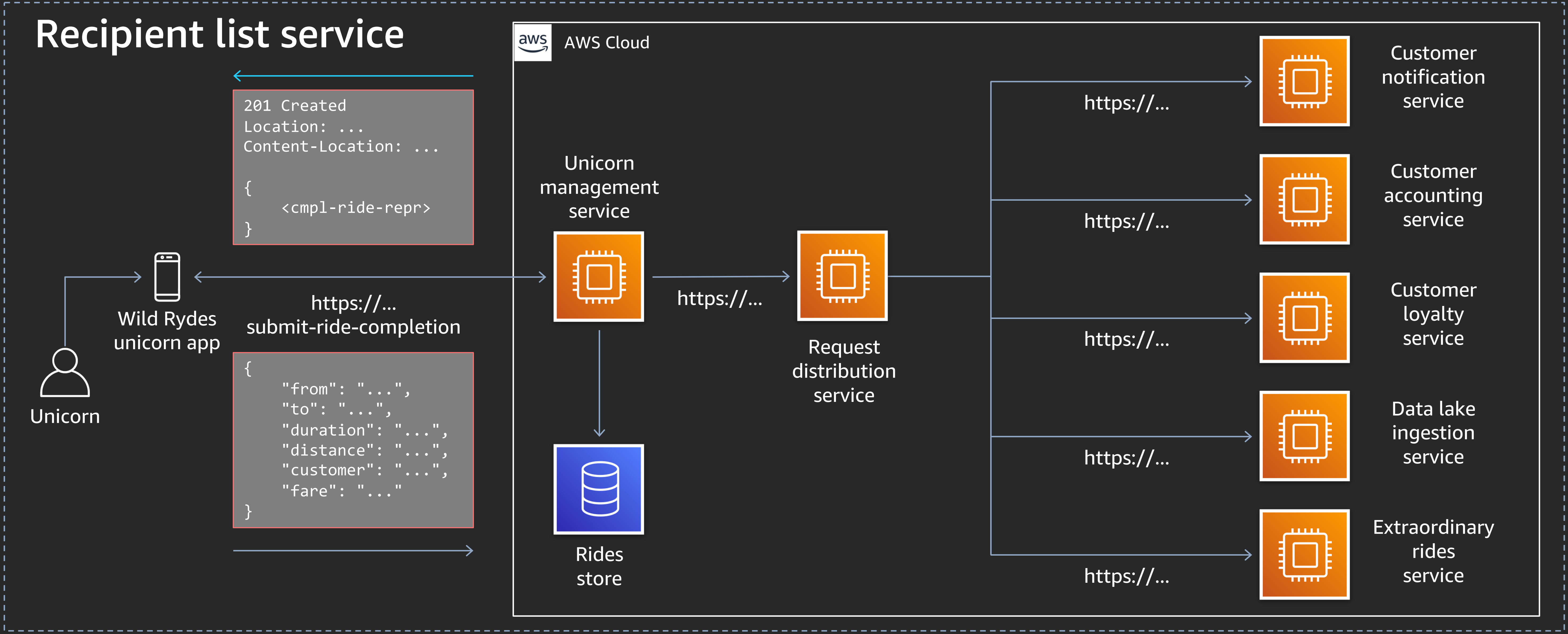

AWS Cloud
Unicorn
Wild Rydes unicorn app
https://... submit-ride-completion
Unicorn management service
Rides store
https://...
Customer notification service
https://...
Customer accounting service
https://...
Customer loyalty service
https://...
Data lake ingestion service
https://...
Extraordinary rides service

```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion

Recipient list service

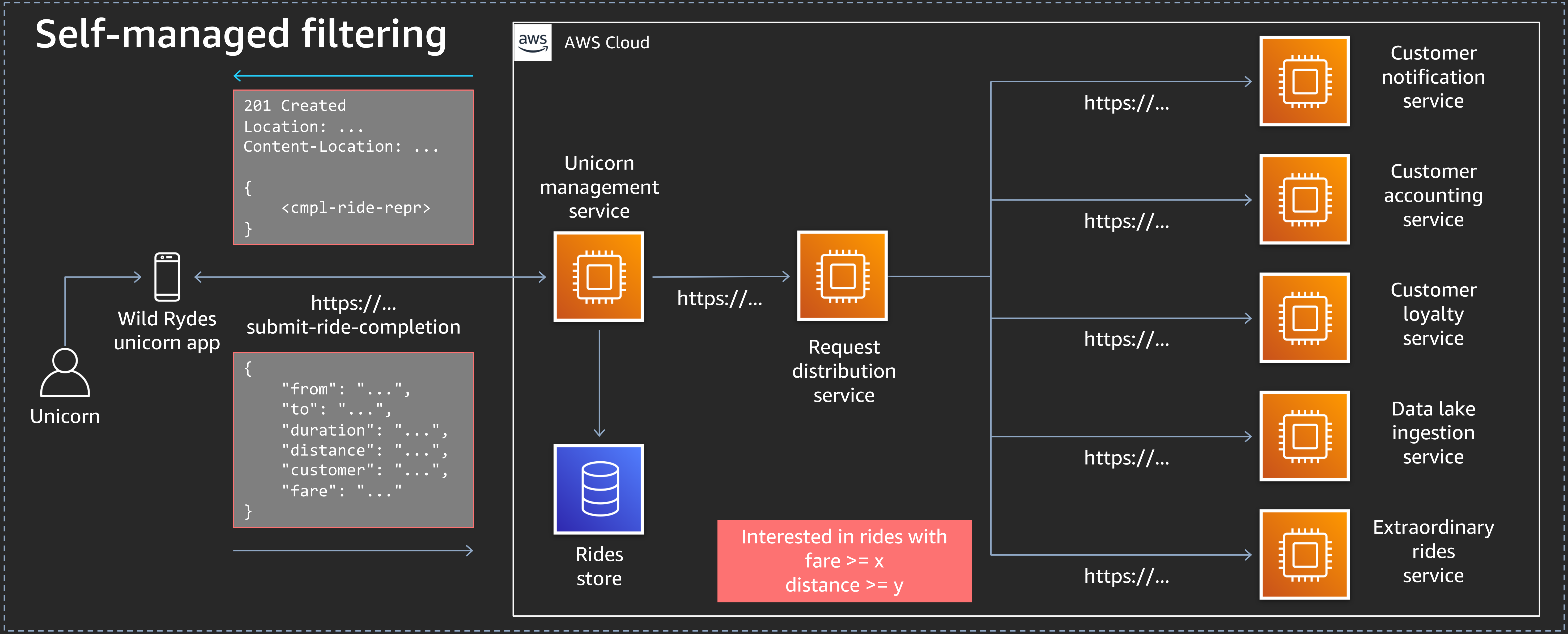


```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion

Self-managed filtering

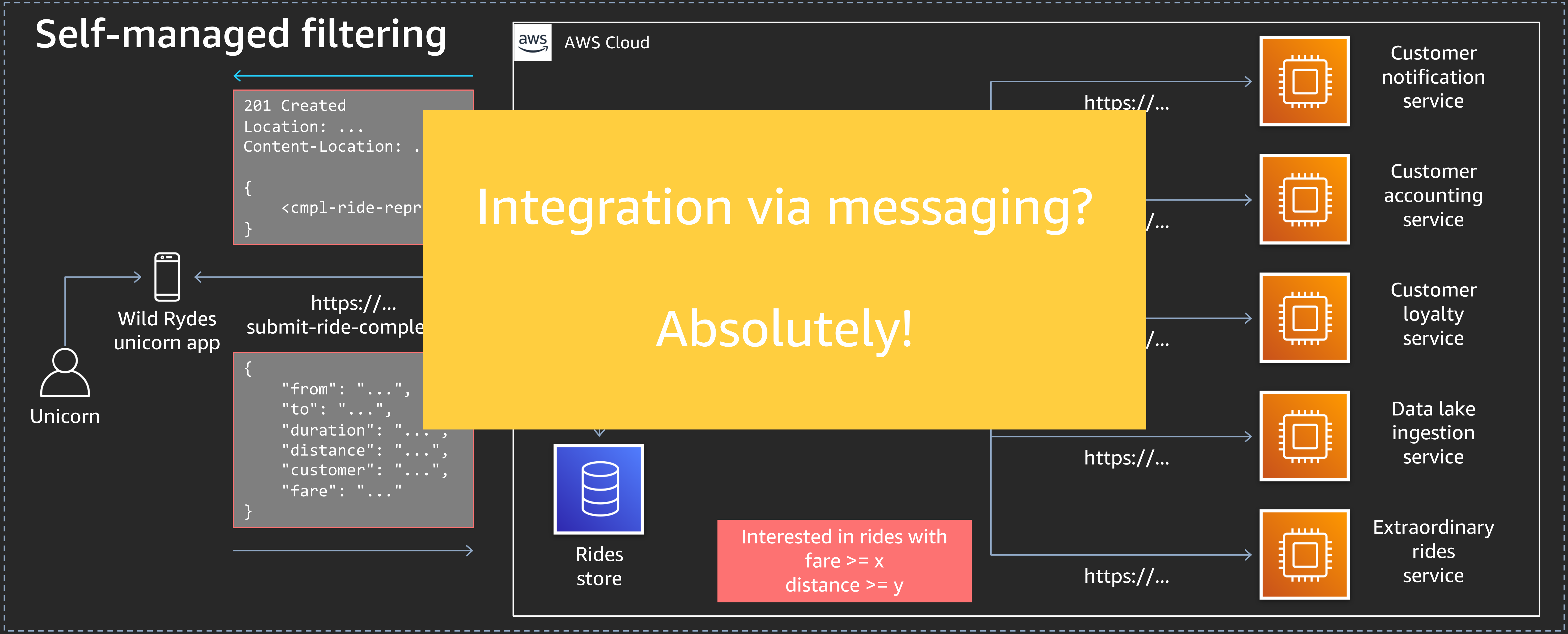


```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Unicorn
Wild Rydes unicorn app
https://...
submit-ride-completion
AWS Cloud
Unicorn management service
Rides store
https://...
Request distribution service
Interested in rides with fare >= x distance >= y
https://...
Customer notification service
https://...
Customer accounting service
https://...
Customer loyalty service
https://...
Data lake ingestion service
https://...
Extraordinary rides service

Use case: Submit a ride completion



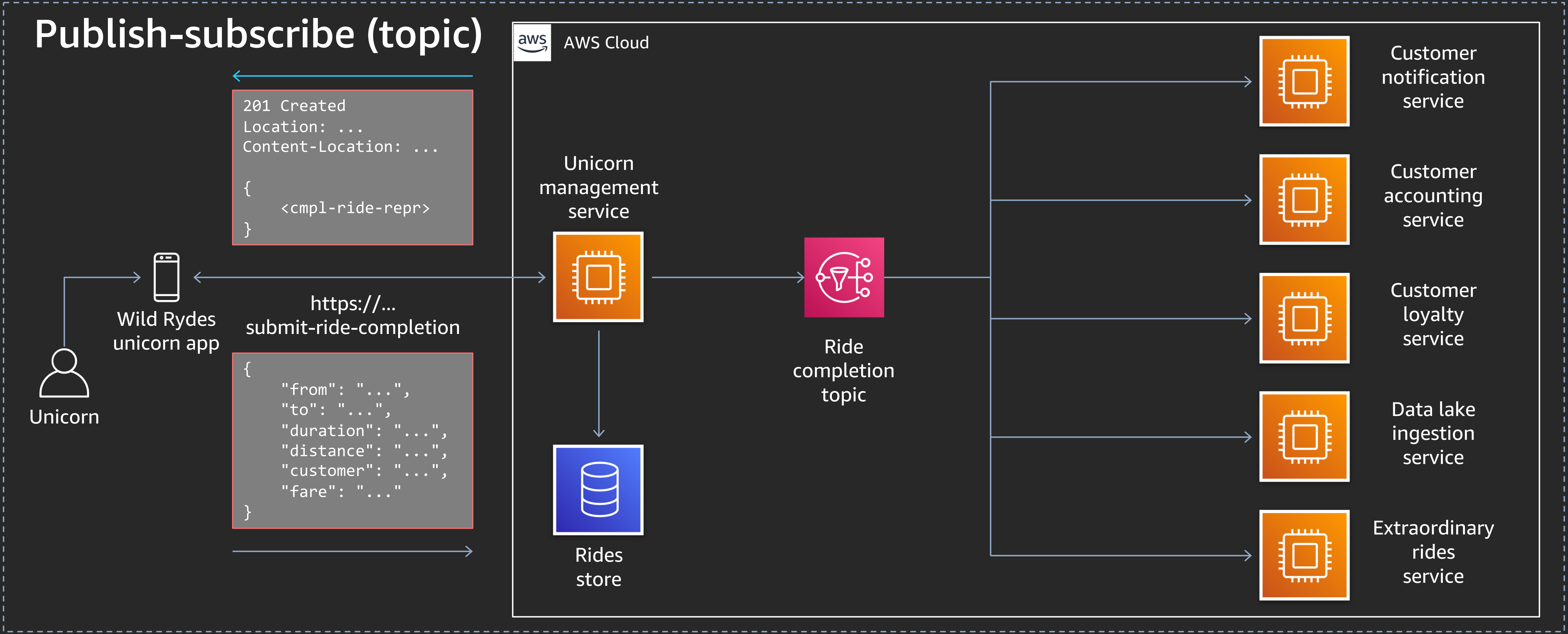
```
201 Created
Location: ...
Content-Location: .

{
    <cmpl-ride-repr
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Integration via messaging?

Absolutely!

Use case: Submit a ride completion

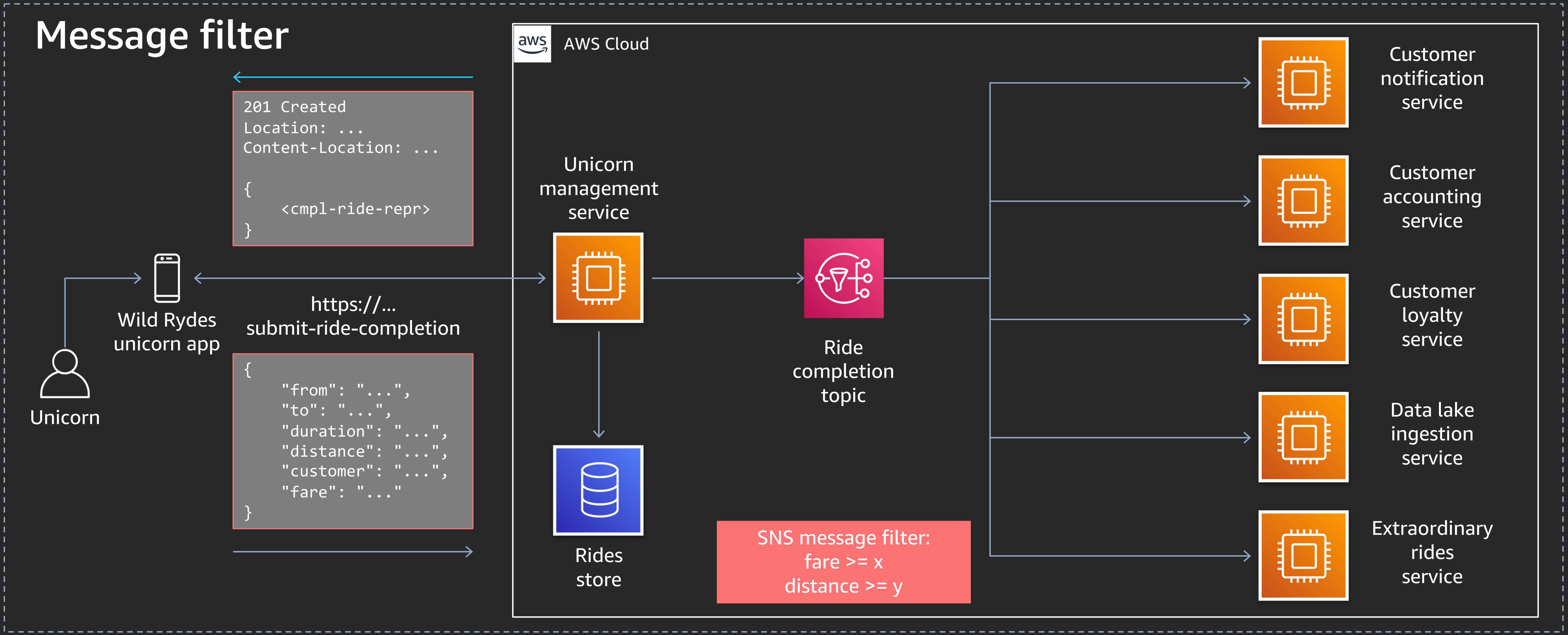

Publish-subscribe (topic)
AWS Cloud
Unicorn
Wild Rydes unicorn app
https://...
submit-ride-completion
Unicorn management service
Rides store
Ride completion topic
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion

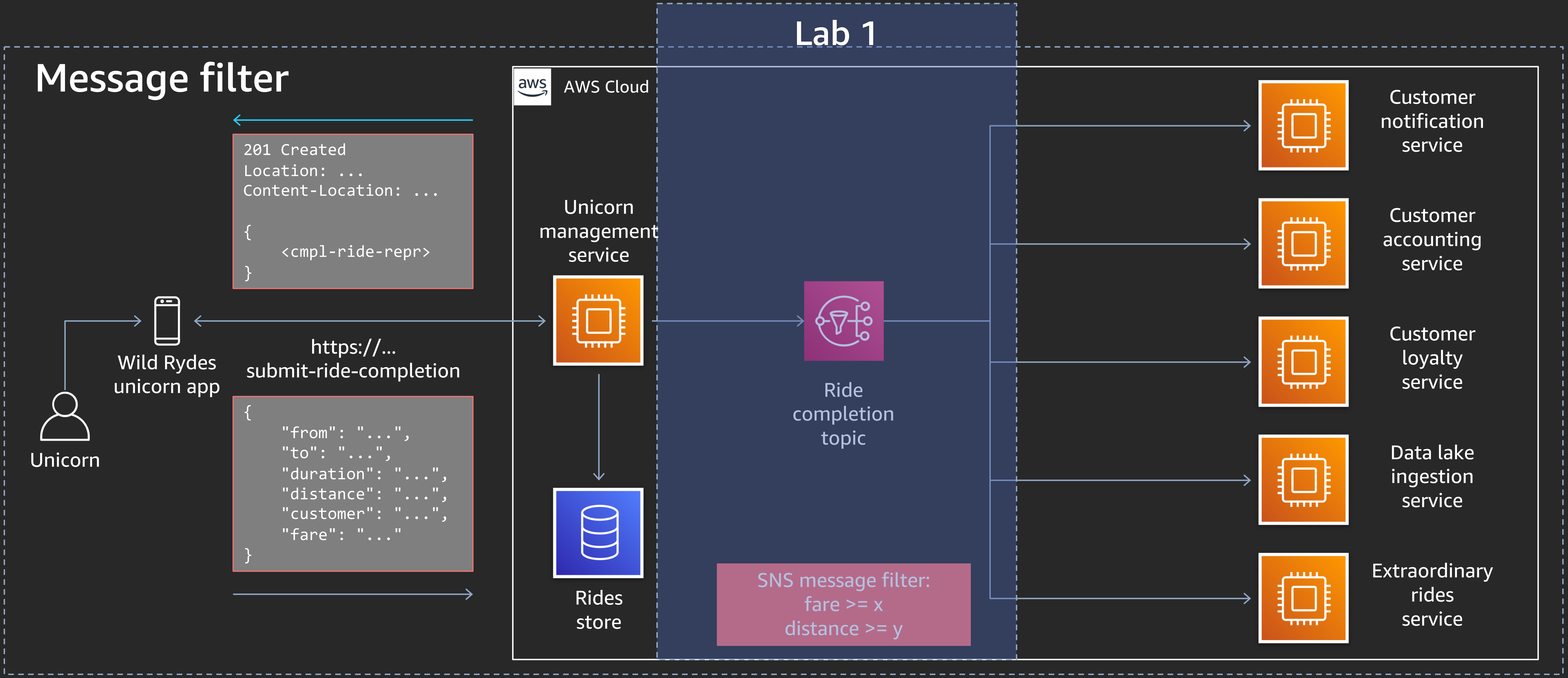
Message filter



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

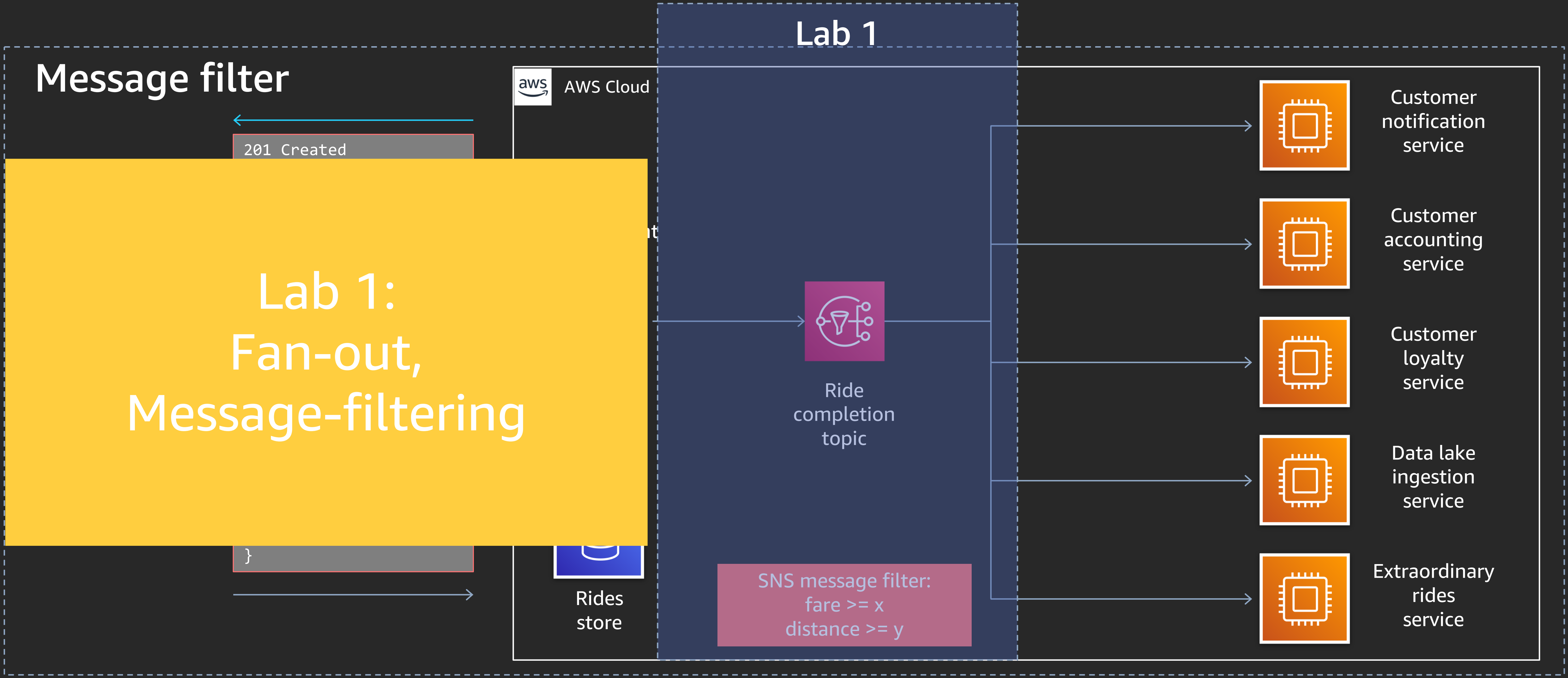
Use case: Submit a ride completion



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

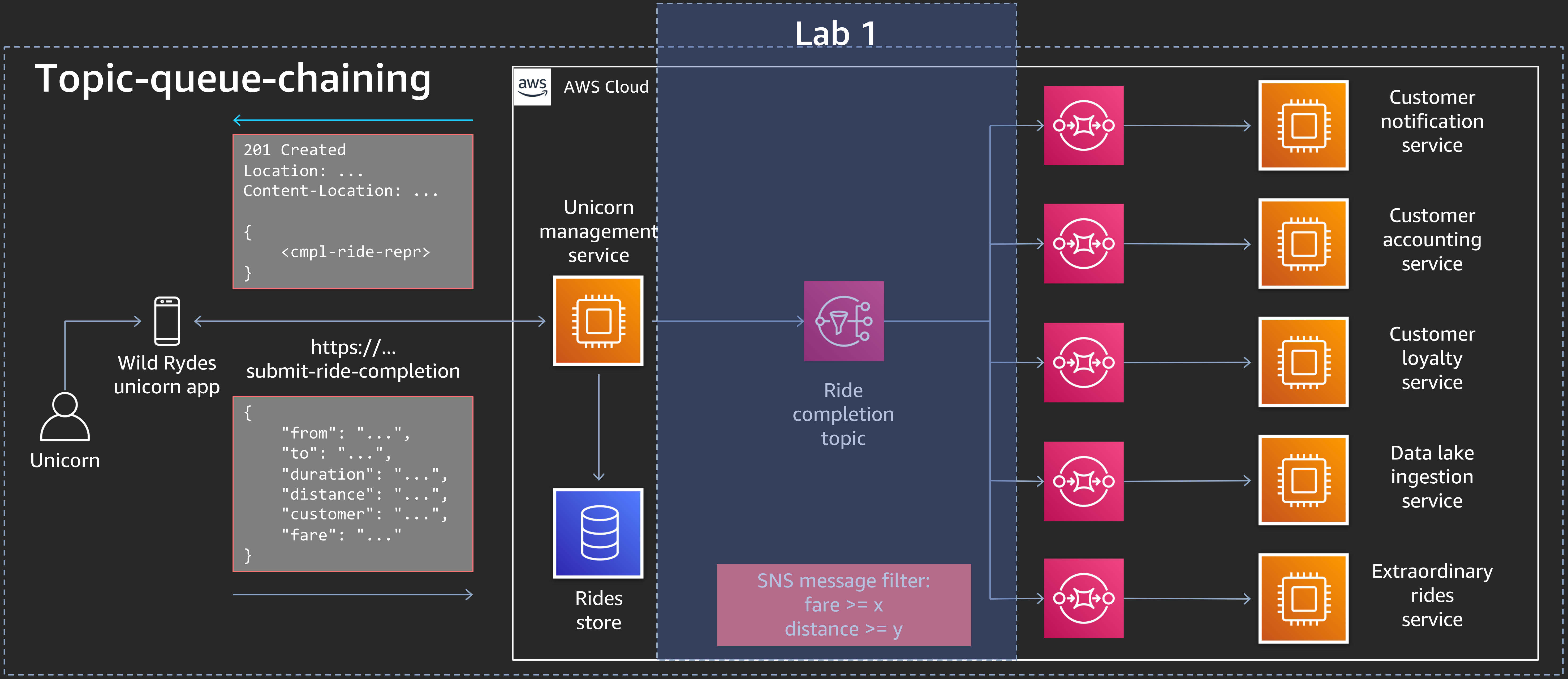
```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion



Lab 1:
Fan-out,
Message-filtering

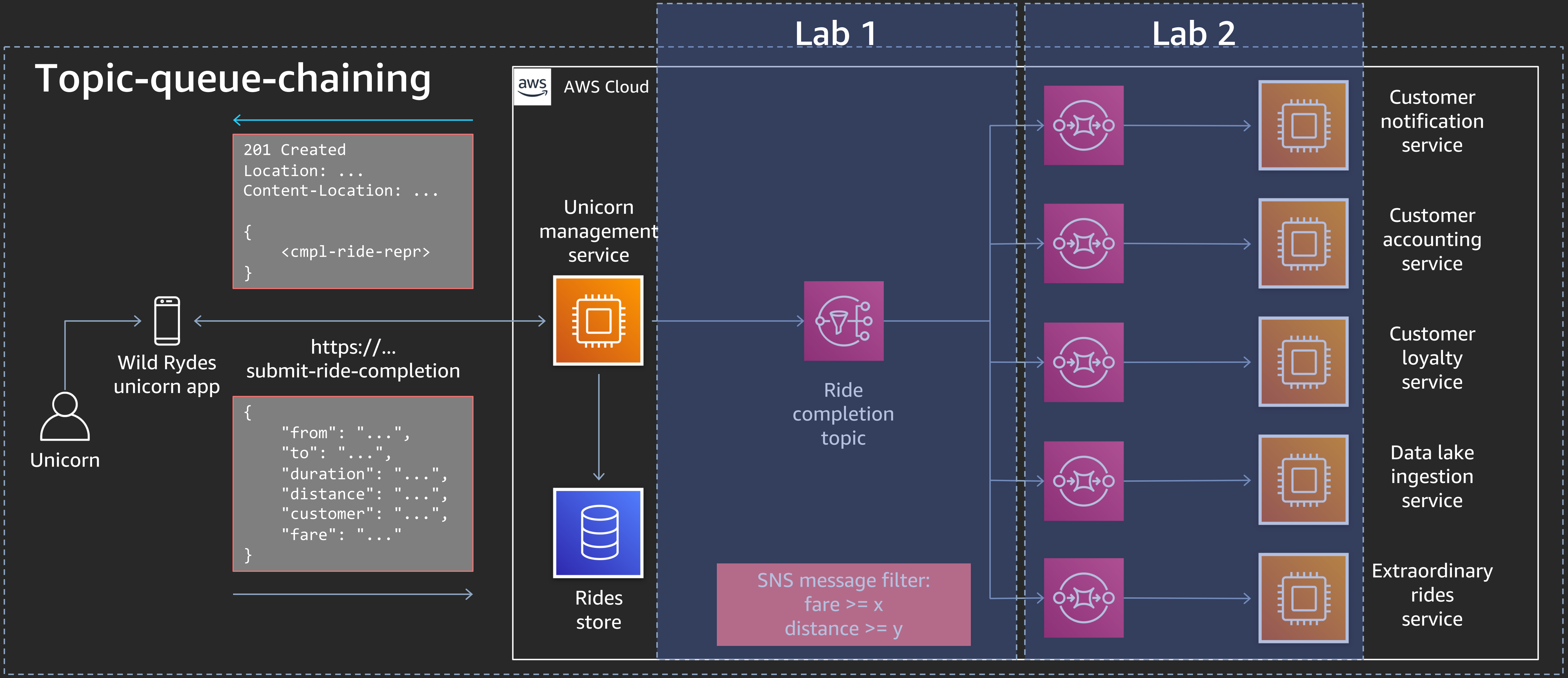
Use case: Submit a ride completion



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion

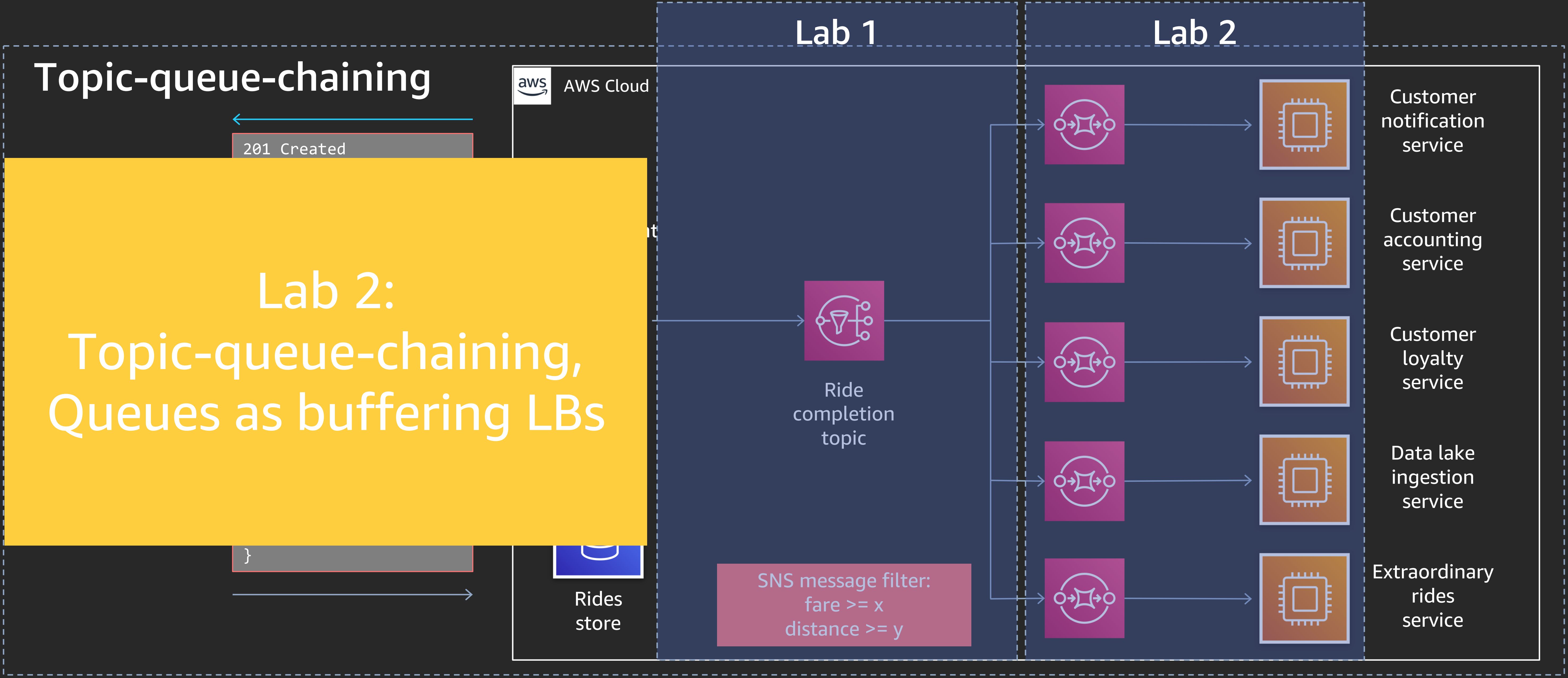


```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Topic-queue-chaining
Lab 1
Lab 2
AWS Cloud
Unicorn
Wild Rydes unicorn app
https://... submit-ride-completion
Unicorn management service
Rides store
Ride completion topic
SNS message filter: fare >= x distance >= y
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

Use case: Submit a ride completion



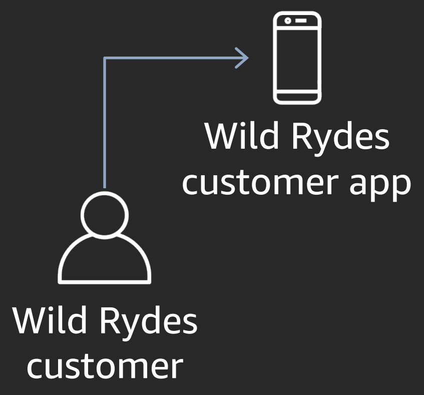
Lab 2:
Topic-queue-chaining,
Queues as buffering LBs

Use case: Instant ride RFQ Context for lab 3

AWS re:Invent

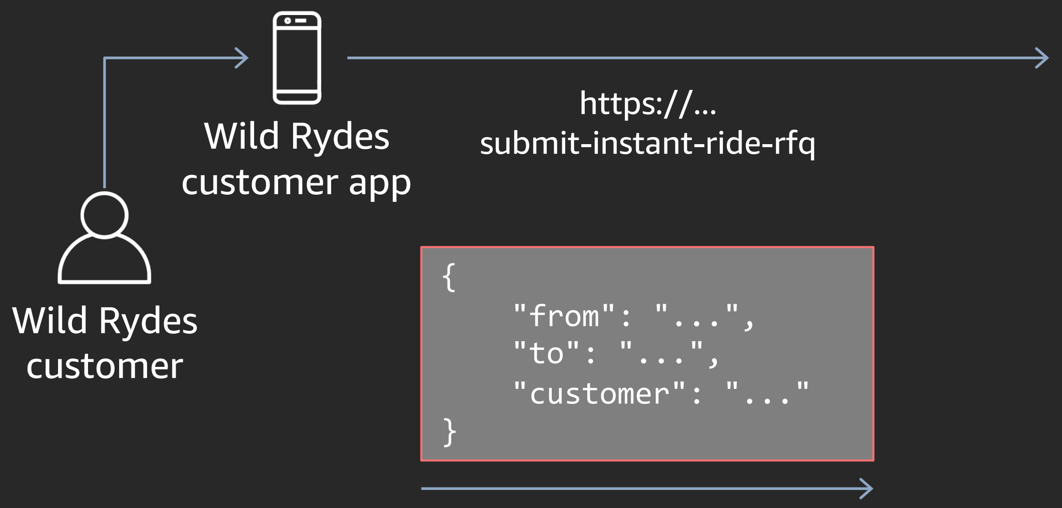
Use case: Instant ride RFQ

Scatter-gather



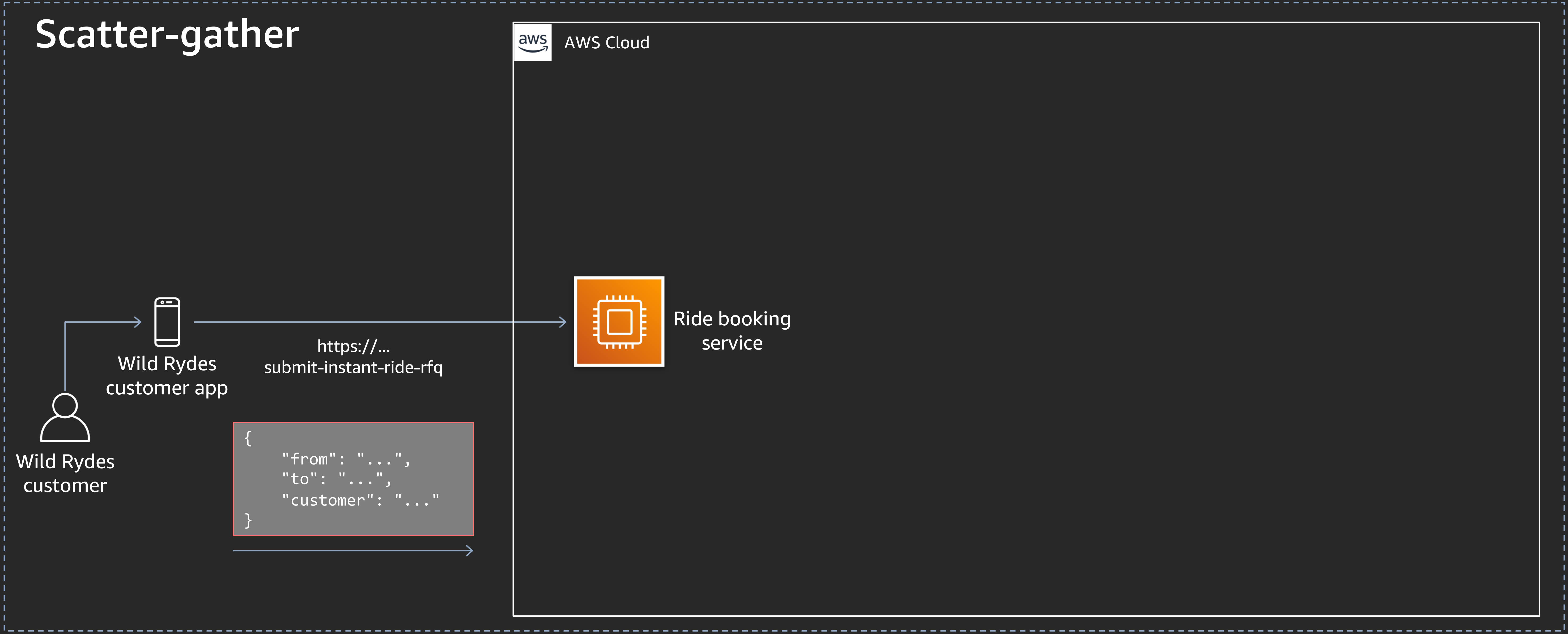
Use case: Instant ride RFQ

Scatter-gather



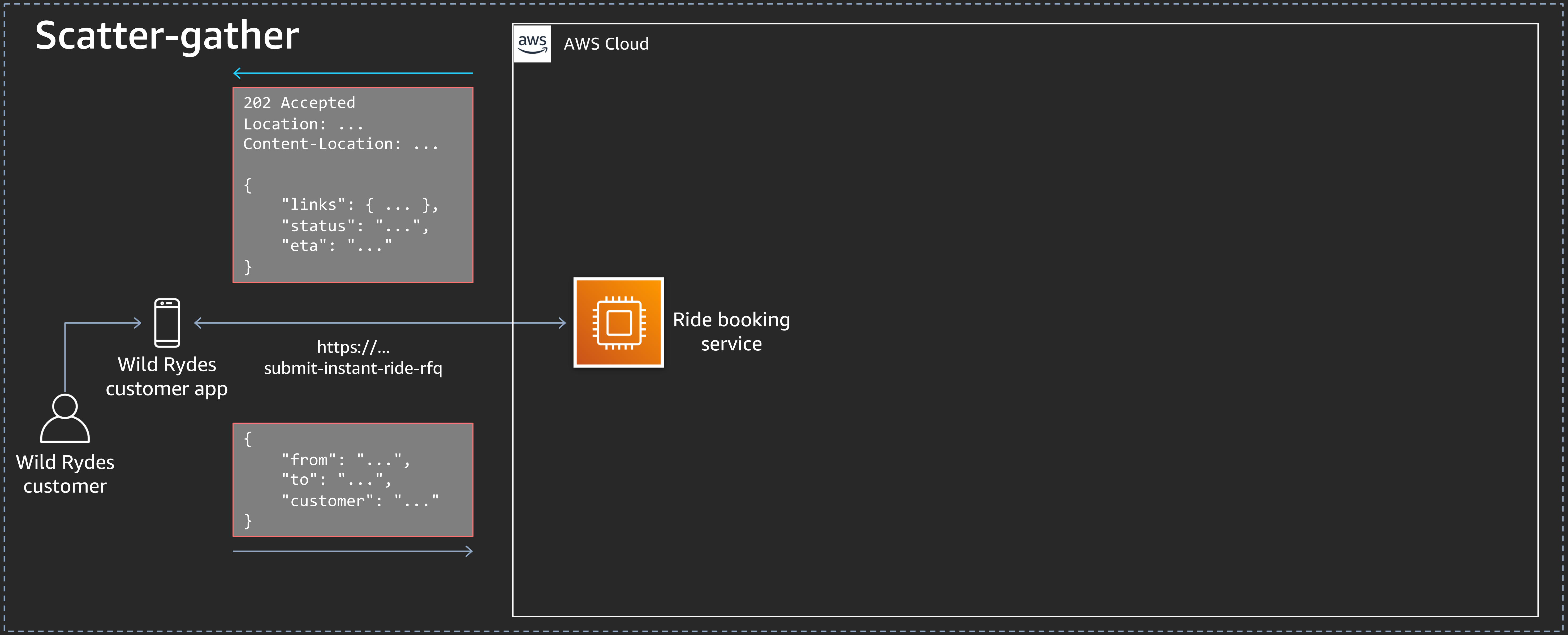
```
{
    "from": "...",
    "to": "...",
    "customer": "..."
}
```

Use case: Instant ride RFQ



```
{
    "from": "...",
    "to": "...",
    "customer": "..."
}
```

Use case: Instant ride RFQ



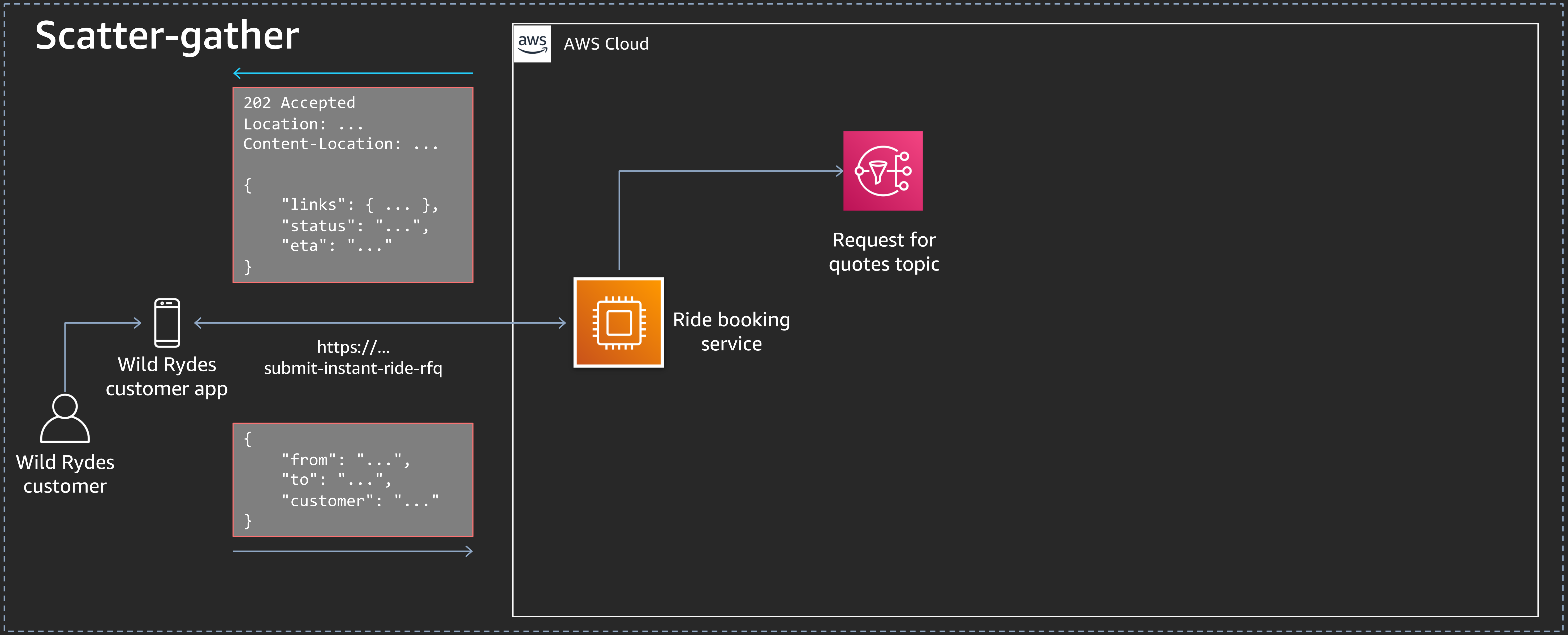
Scatter-gather

```
202 Accepted
Location: ...
Content-Location: ...

{
    "links": { ... },
    "status": "...",
    "eta": "..."
}
```

https://...
submit-instant-ride-rfq

```
{
    "from": "...",
    "to": "...",
    "customer": "..."
}
```

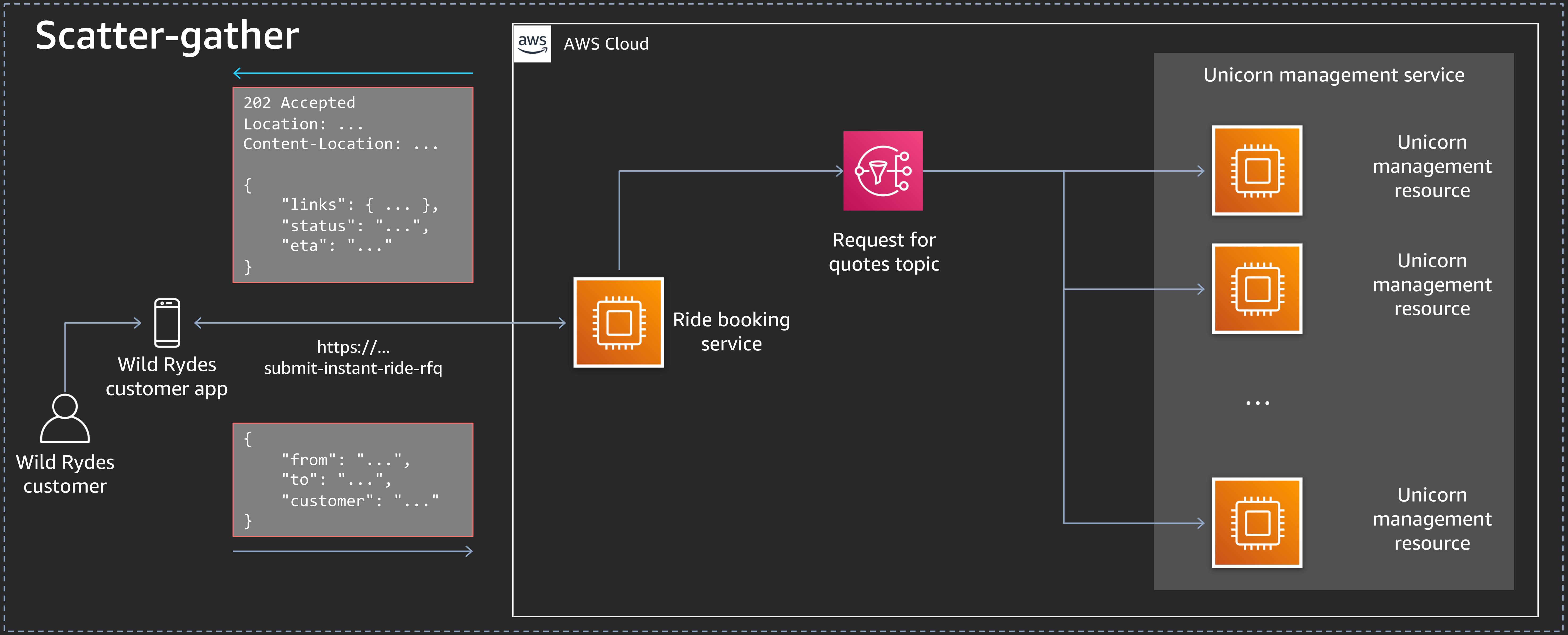
Use case: Instant ride RFQ



```
202 Accepted
Location: ...
Content-Location: ...

{
    "links": { ... },
    "status": "...",
    "eta": "..."
}
```

```
{
    "from": "...",
    "to": "...",
    "customer": "..."
}
```

Use case: Instant ride RFQ

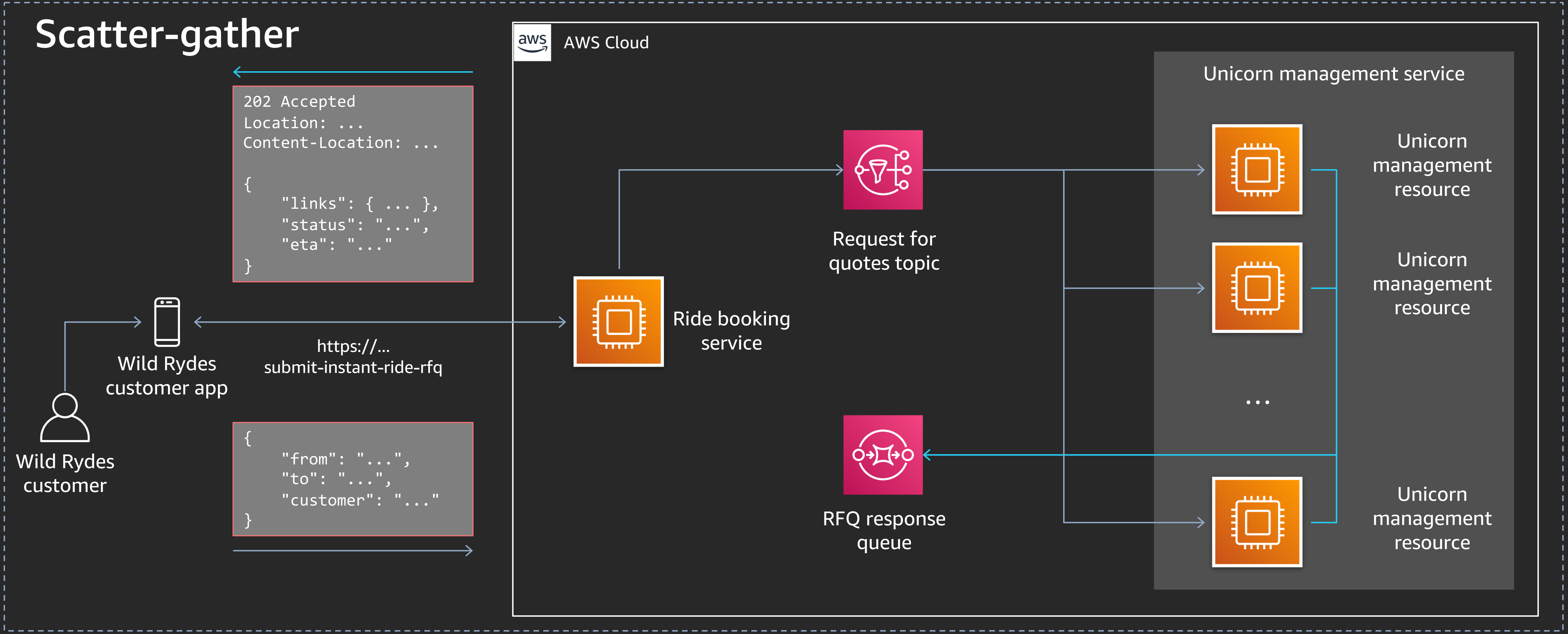


Scatter-gather

```
202 Accepted
Location: ...
Content-Location: ...

{
    "links": { ... },
    "status": "...",
    "eta": "..."
}
```

```
{
    "from": "...",
    "to": "...",
    "customer": "..."
}
```

Use case: Instant ride RFQ

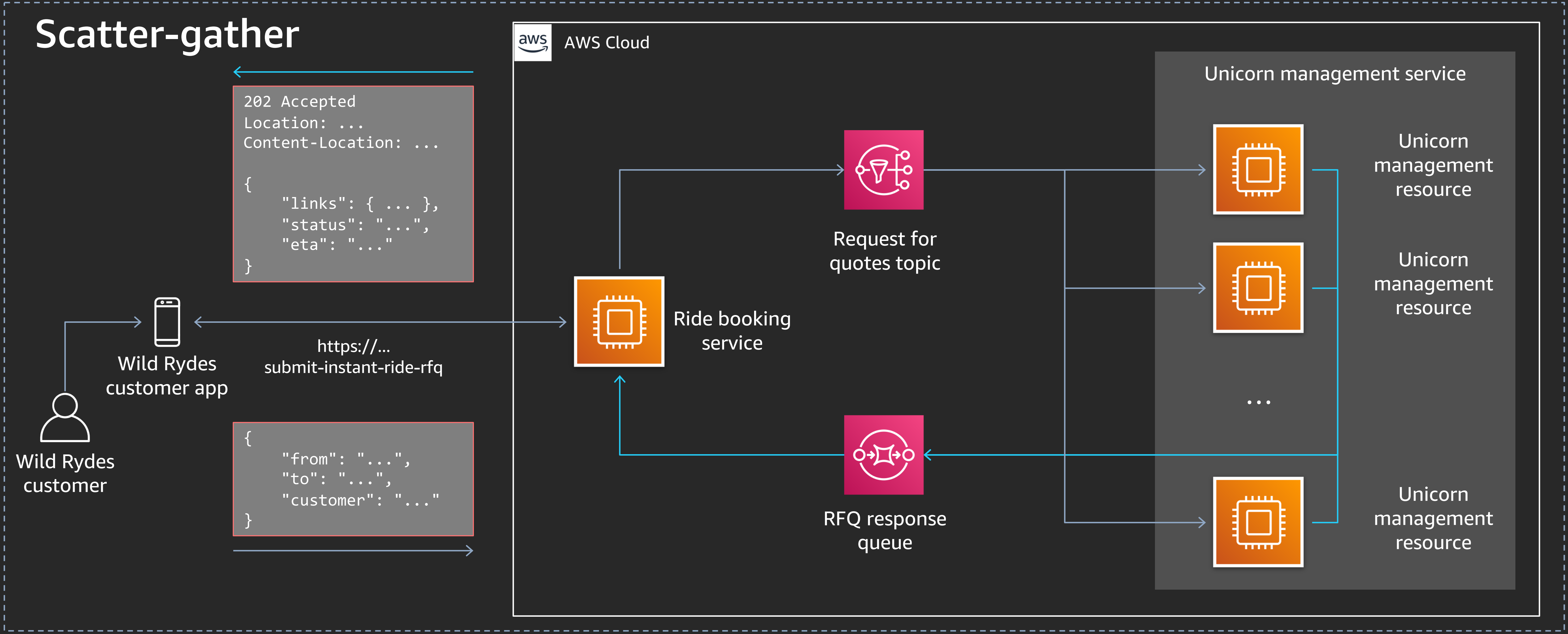


```
202 Accepted
Location: ...
Content-Location: ...

{
    "links": { ... },
    "status": "...",
    "eta": "..."
}
```

```
{
    "from": "...",
    "to": "...",
    "customer": "..."
}
```

Use case: Instant ride RFQ

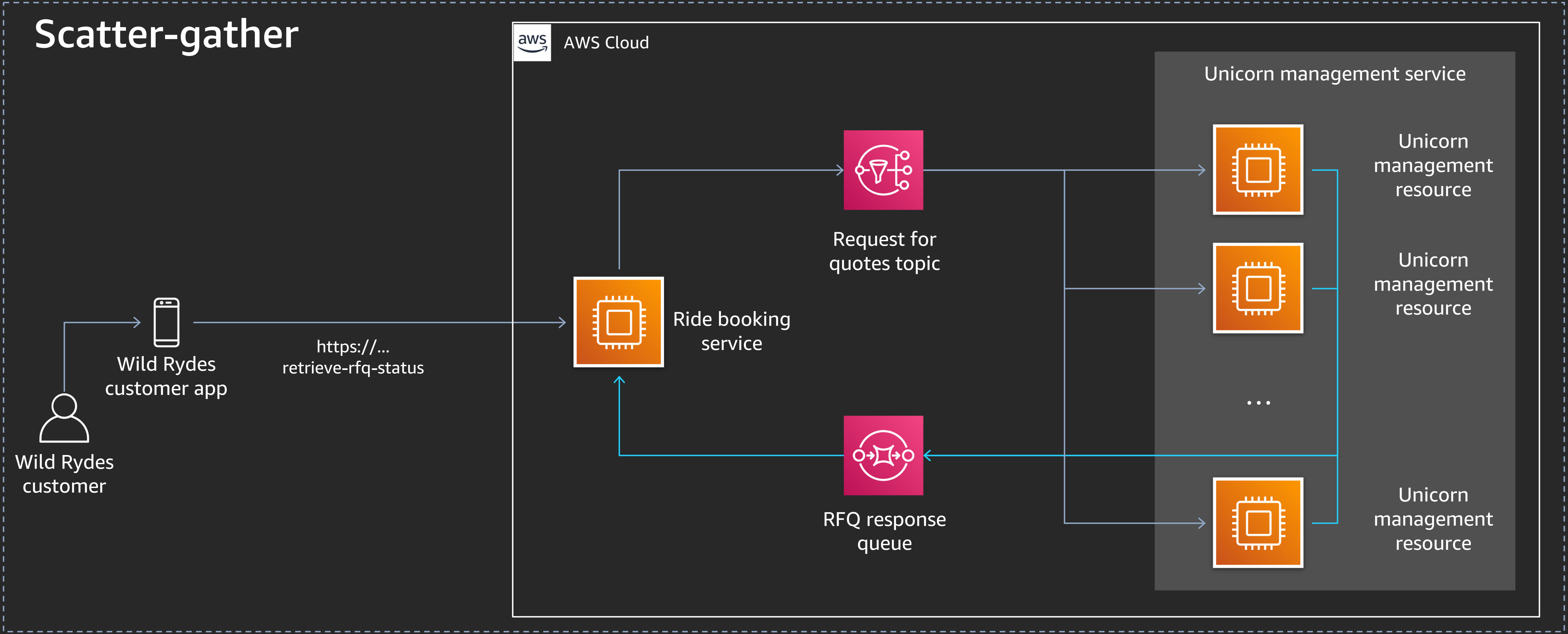
Scatter-gather



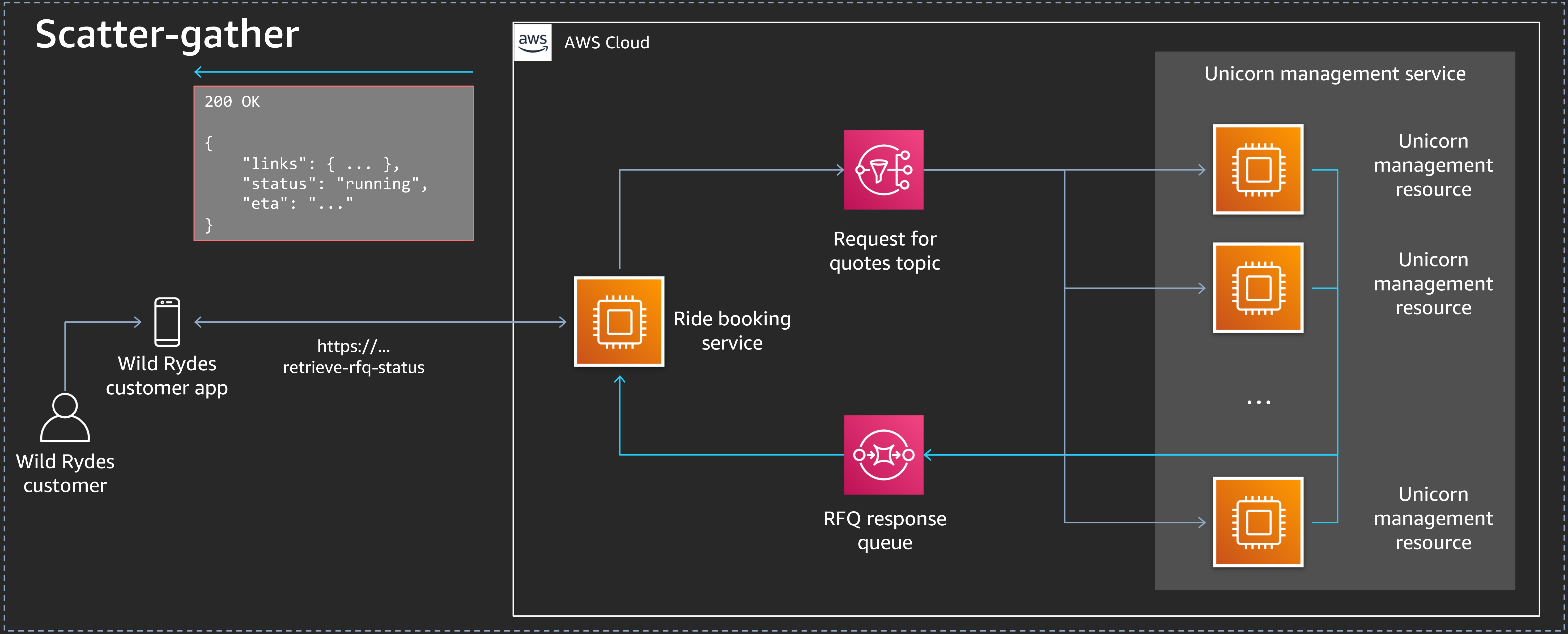
```
202 Accepted
Location: ...
Content-Location: ...

{
    "links": { ... },
    "status": "...",
    "eta": "..."
}
```

```
{
    "from": "...",
    "to": "...",
    "customer": "..."
}
```

Use case: Instant ride RFQ

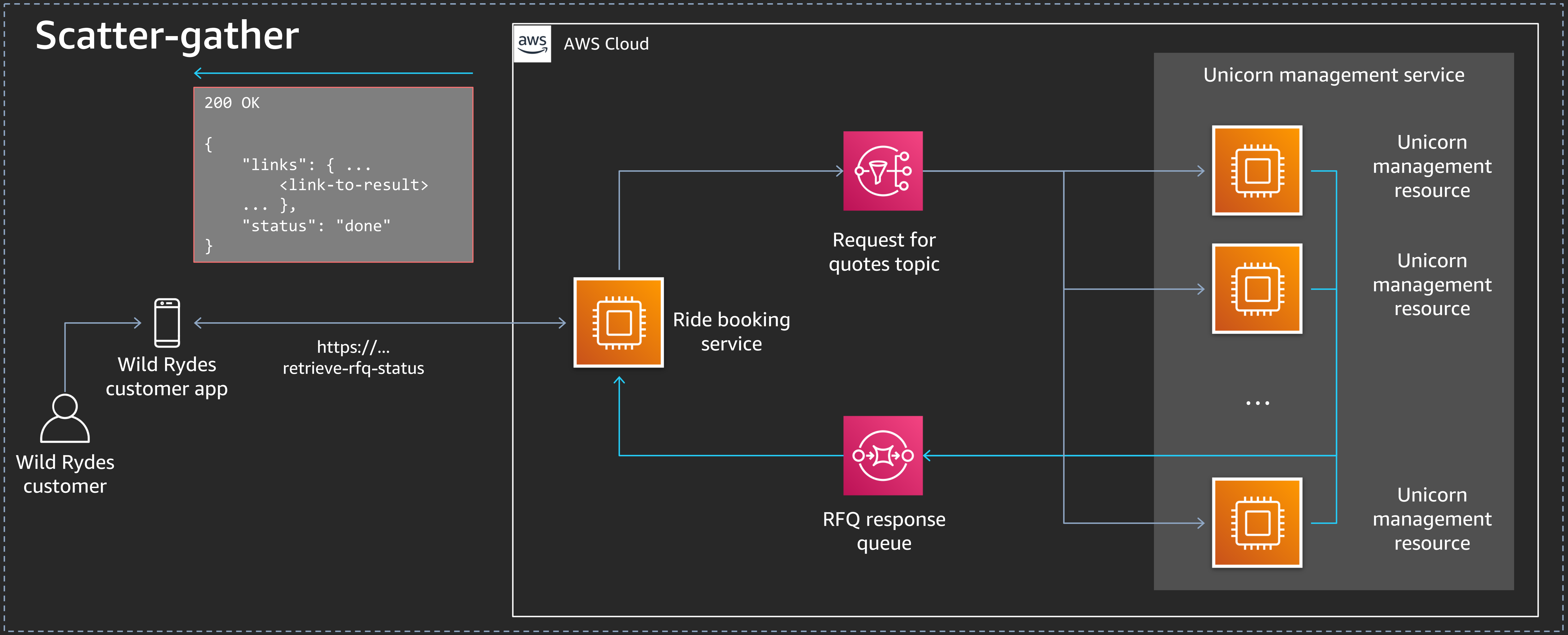


Use case: Instant ride RFQ



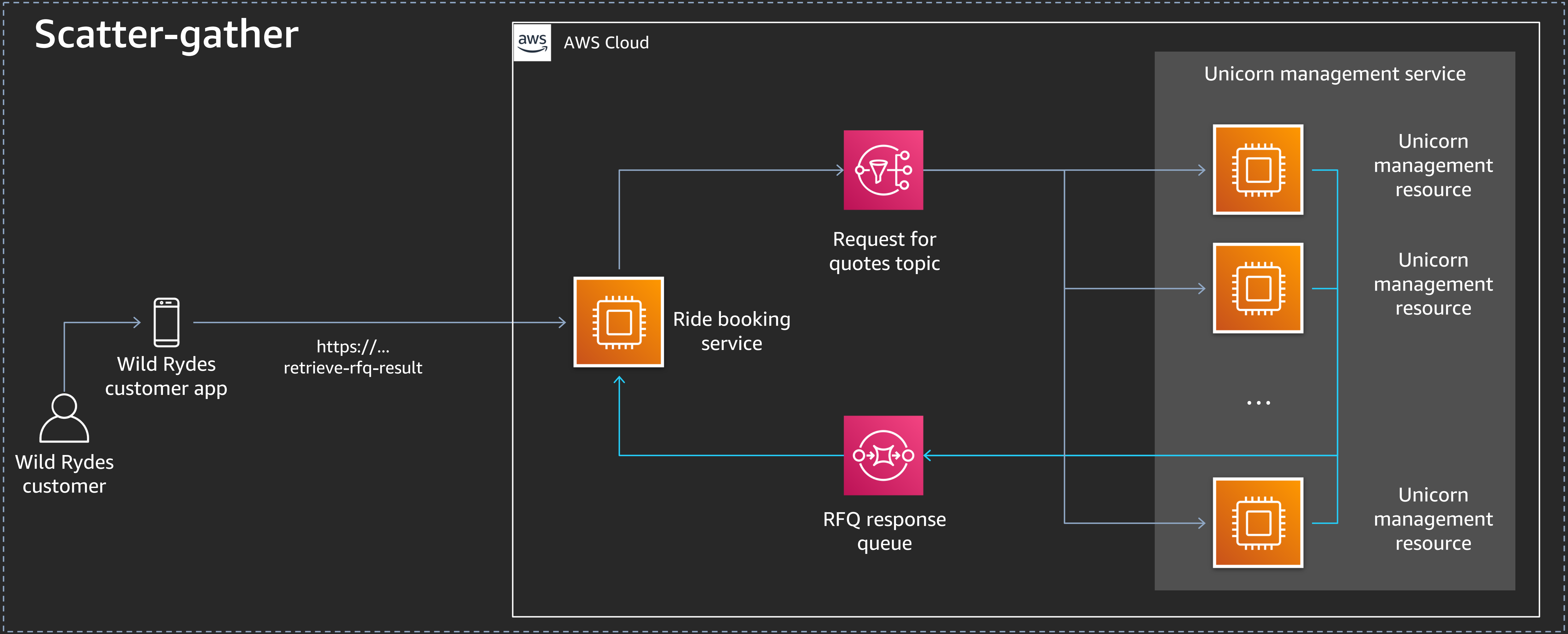
```
200 OK

{
    "links": { ... },
    "status": "running",
    "eta": "..."
}
```

Use case: Instant ride RFQ

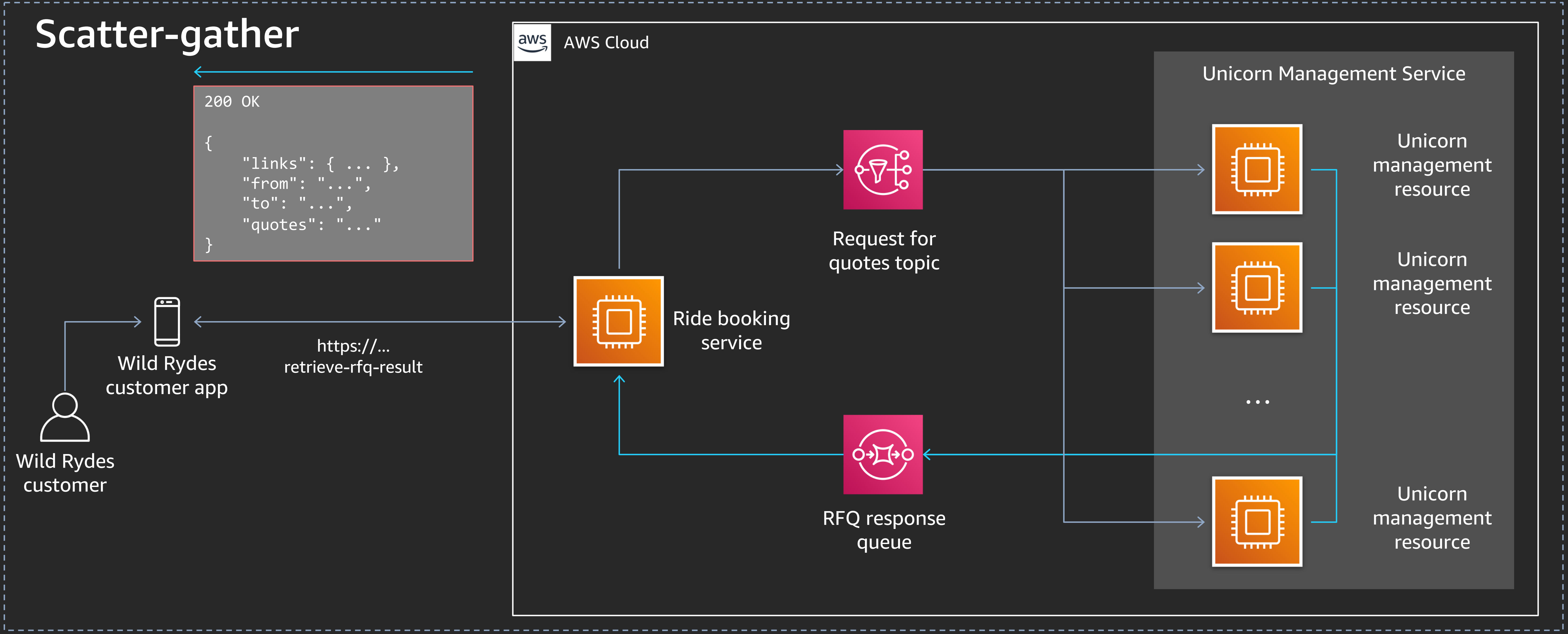


```
200 OK

{
    "links": { ...
        <link-to-result>
    ... },
    "status": "done"
}
```

Use case: Instant ride RFQ


Scatter-gather
aws
AWS Cloud
Unicorn management service
Request for quotes topic
Unicorn management resource
Unicorn management resource
...
Unicorn management resource
Wild Rydes customer app
https://... retrieve-rfq-result
Ride booking service
Wild Rydes customer
RFQ response queue

Use case: Instant ride RFQ

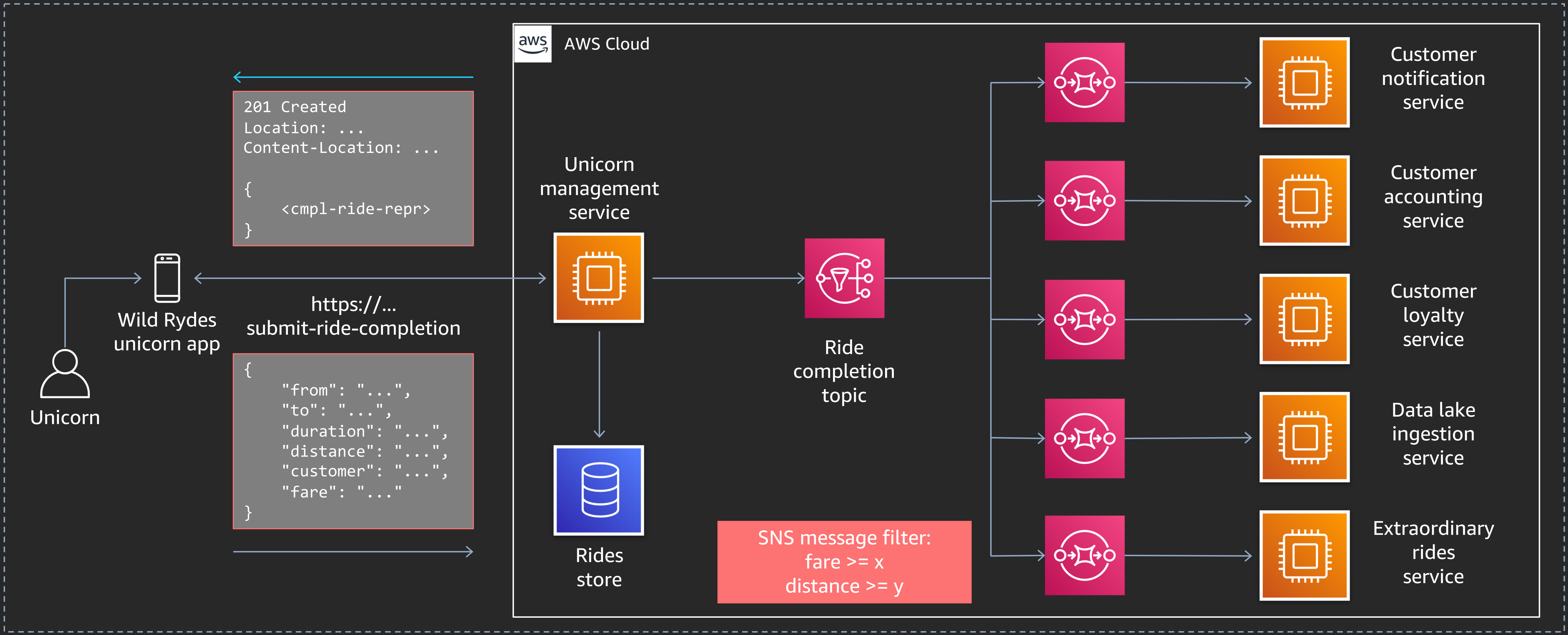


Scatter-gather

```
200 OK

{
    "links": { ... },
    "from": "...",
    "to": "...",
    "quotes": "..."
}
```

Use case: Fare collection
Context for lab 4

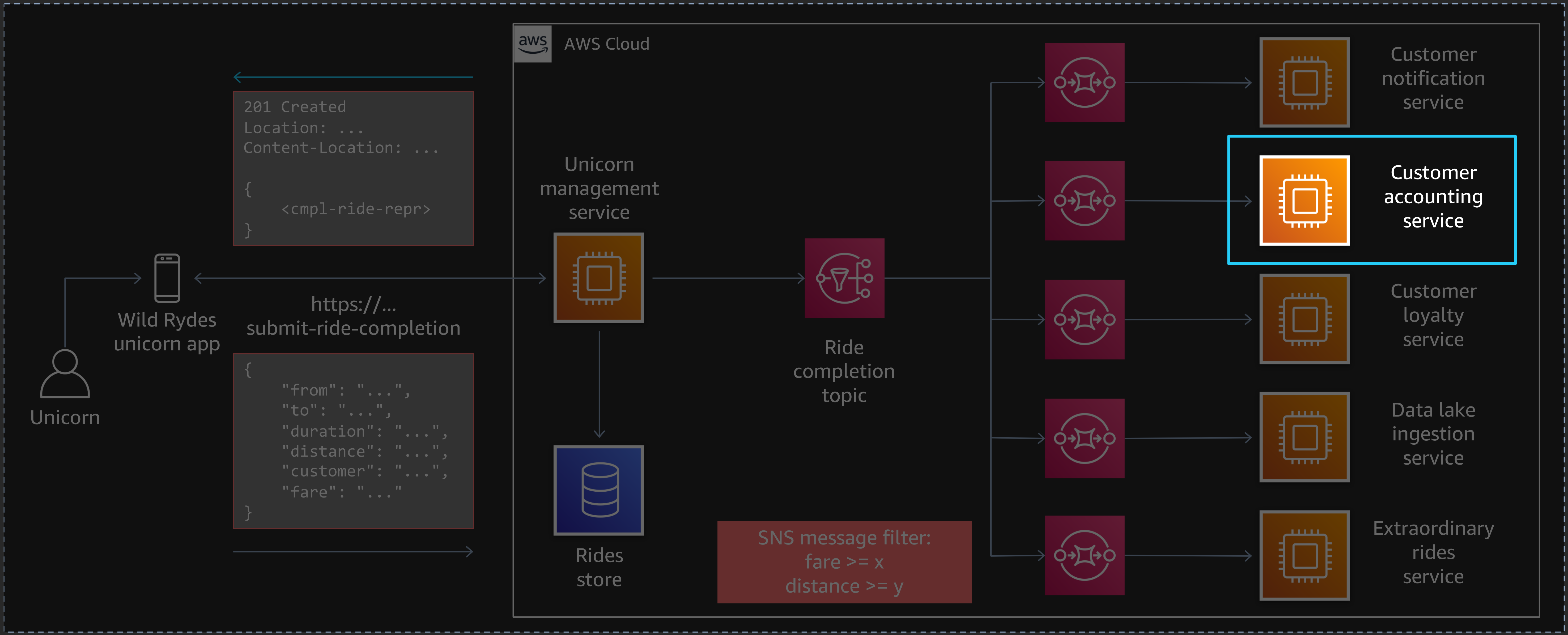
Use case: Submit a ride completion



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

Use case: Submit a ride completion



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

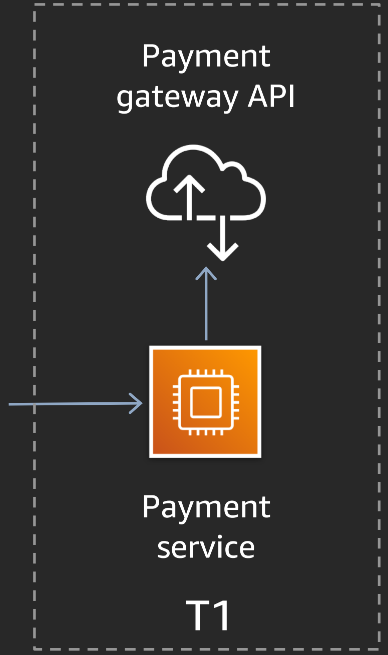
AWS Cloud
Unicorn
Wild Rydes unicorn app
https://... submit-ride-completion
Unicorn management service
Rides store
Ride completion topic
SNS message filter: fare >= x distance >= y
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

Use case: Fare collection

Saga orchestration

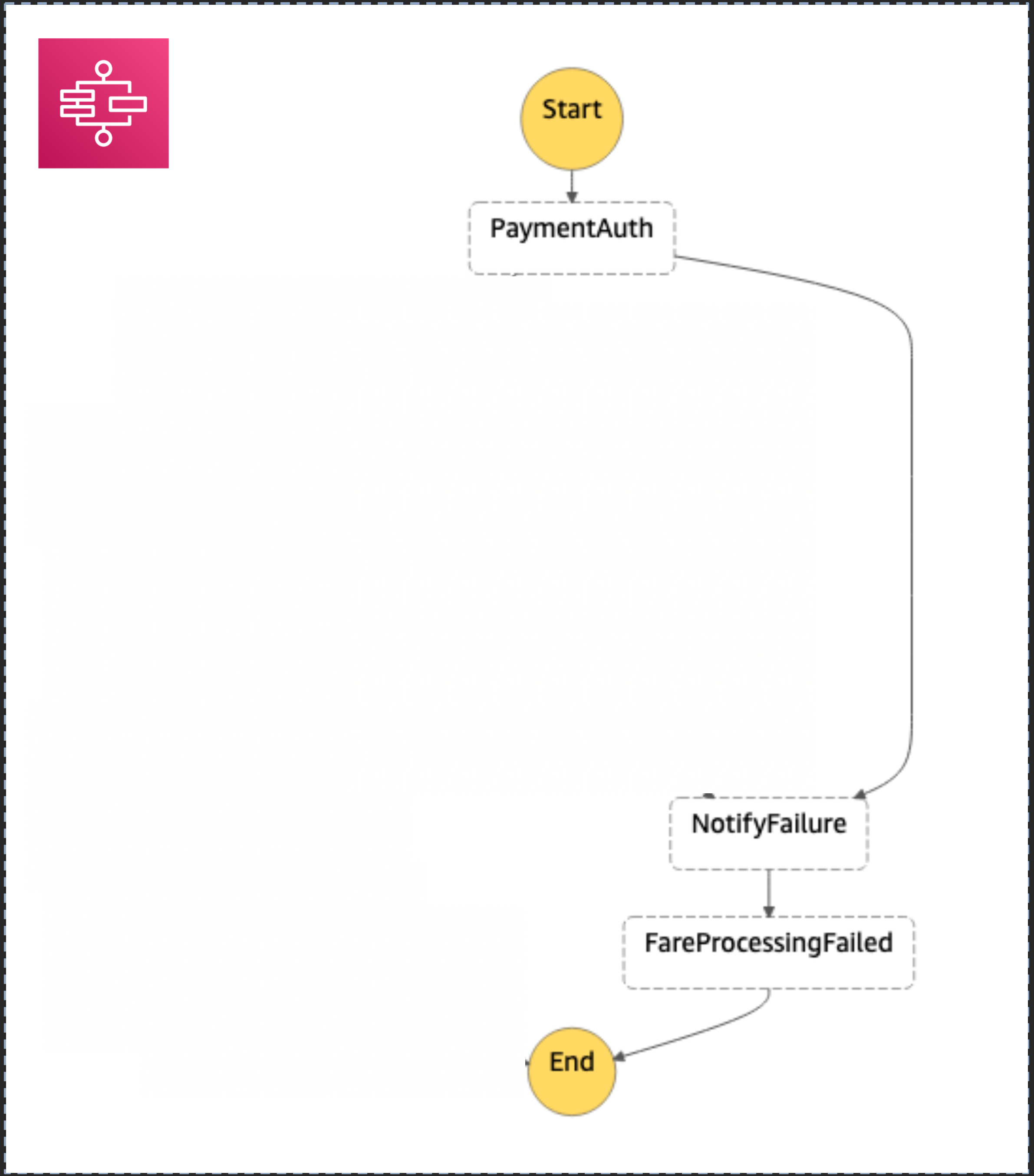
Use case: Fare collection

Saga orchestration



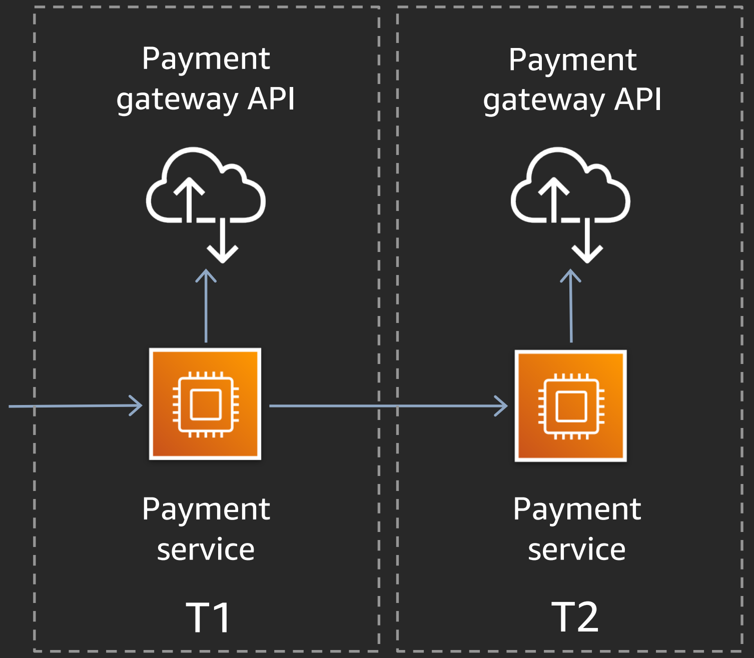
Discrete transactions:

1. Credit card pre-auth



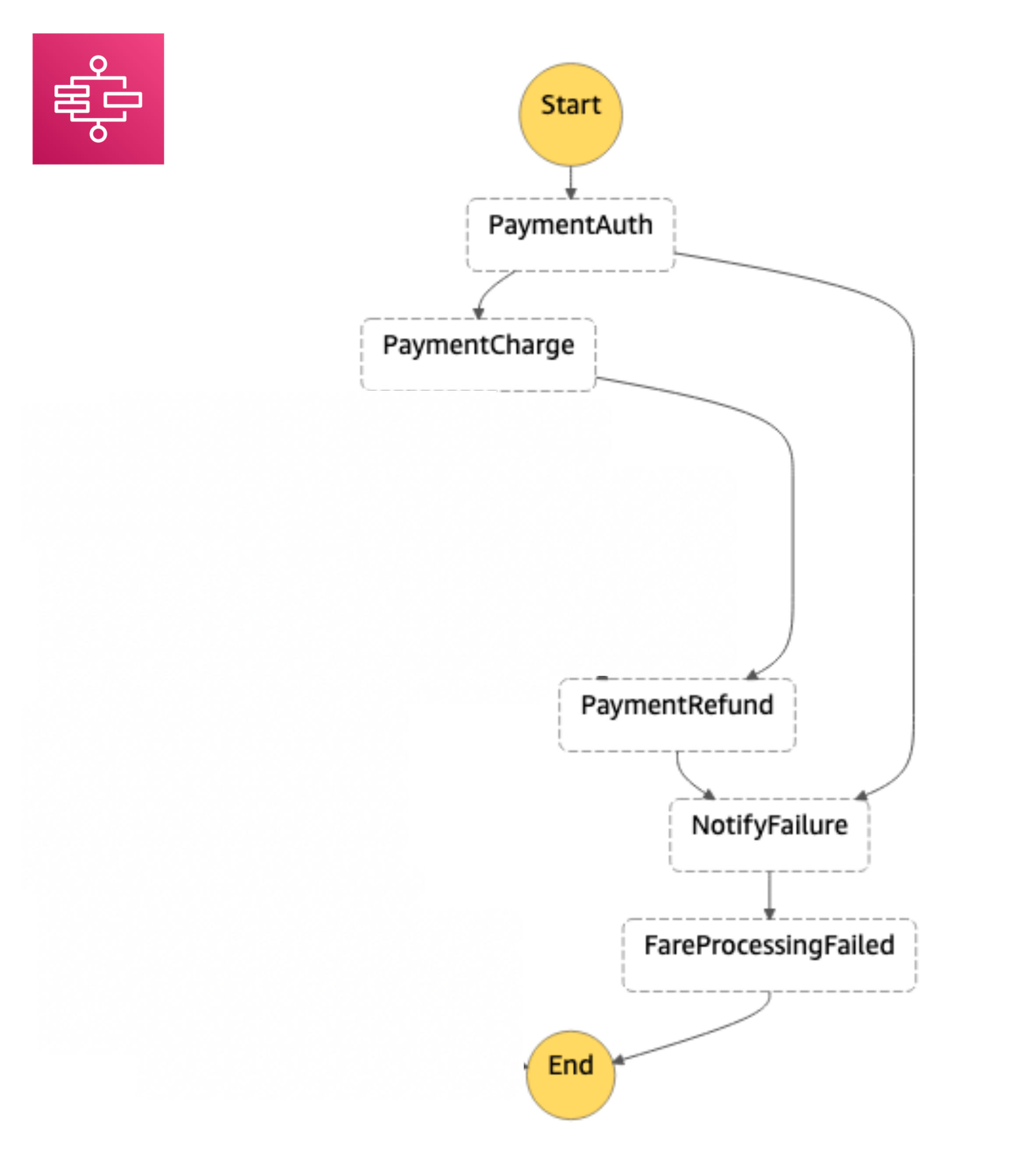
Use case: Fare collection

Saga orchestration



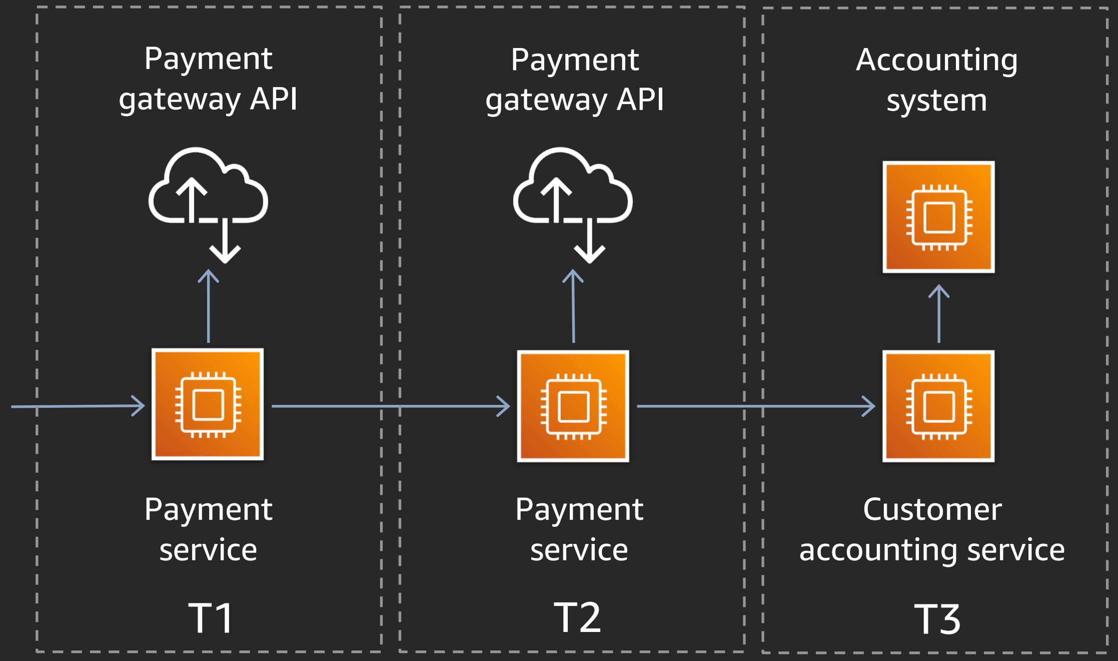
Discrete transactions:

1. Credit card pre-auth
2. Charge card using pre-auth code



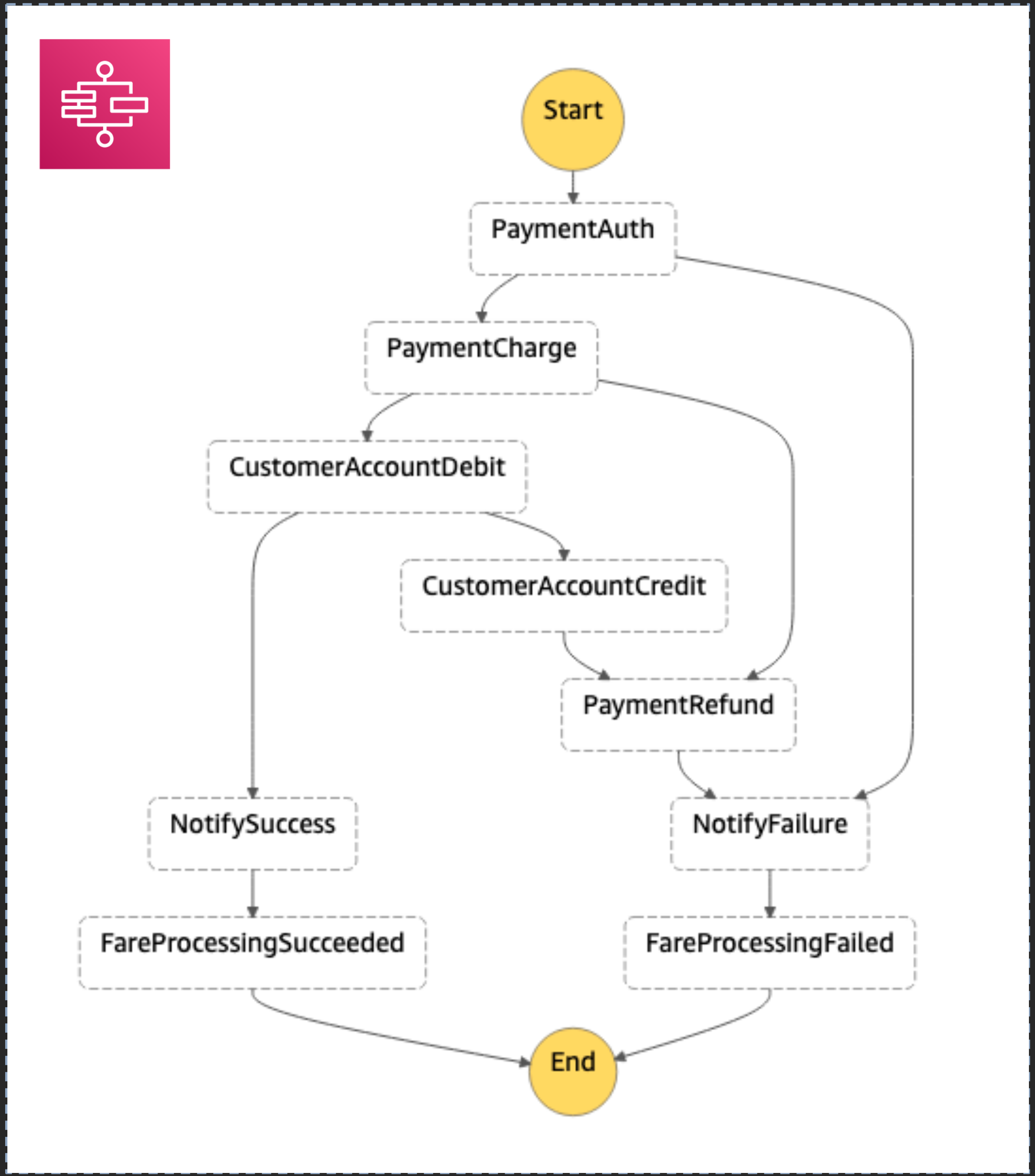
Use case: Fare collection

Saga orchestration



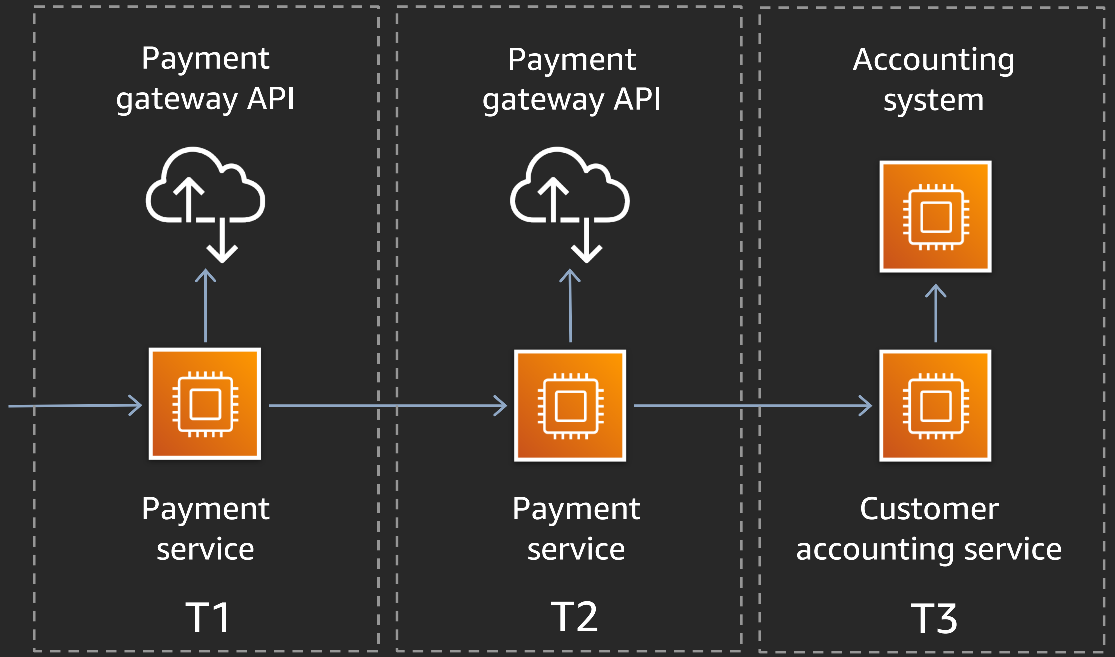
Discrete transactions:

1. Credit card pre-auth
2. Charge card using pre-auth code
3. Update customer account



Use case: Fare collection

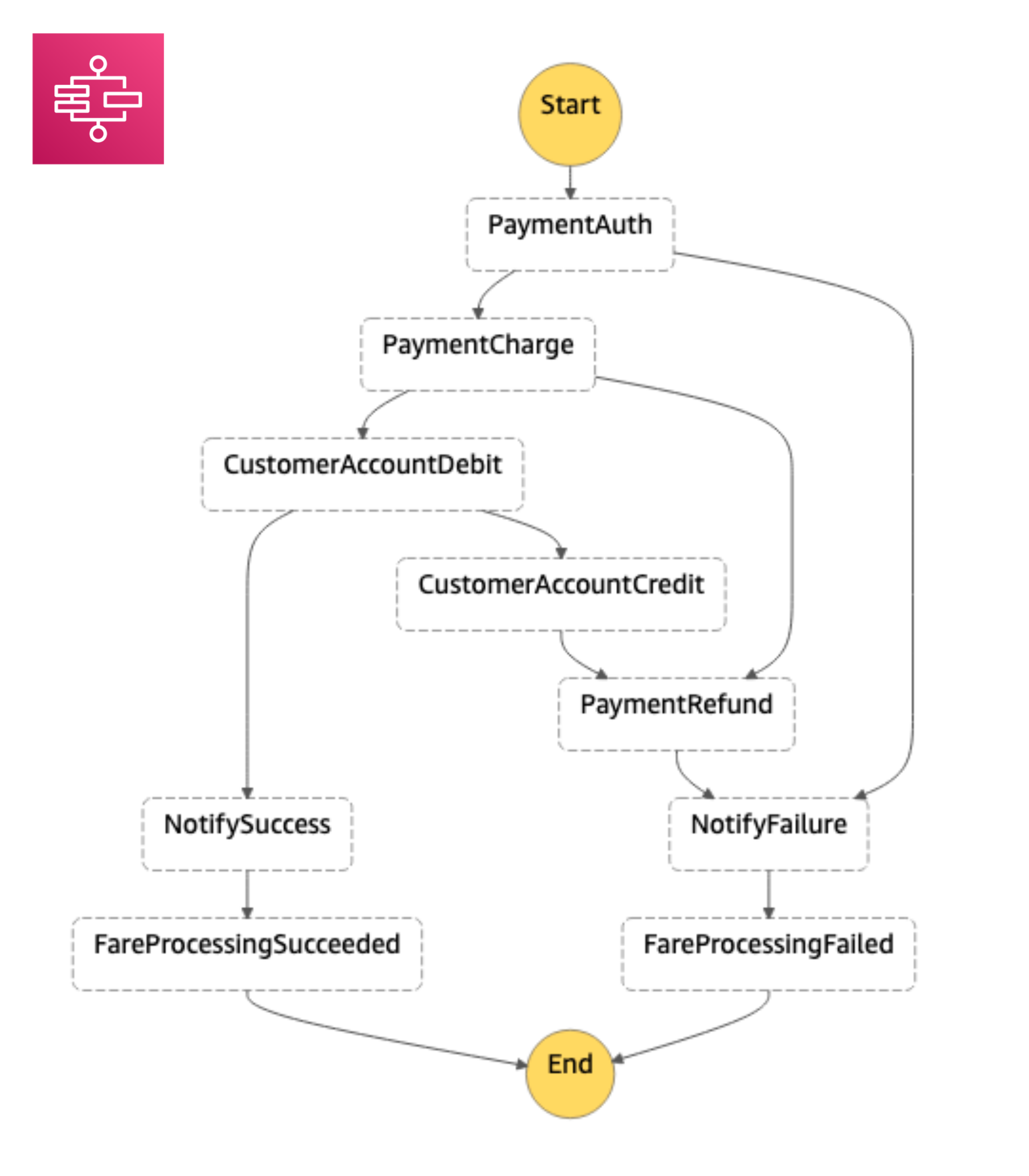
Saga orchestration



Discrete transactions:

1. Credit card pre-auth
2. Charge card using pre-auth code
3. Update customer account

To be treated as one distributed TA, to leave the systems in a semantically consistent state

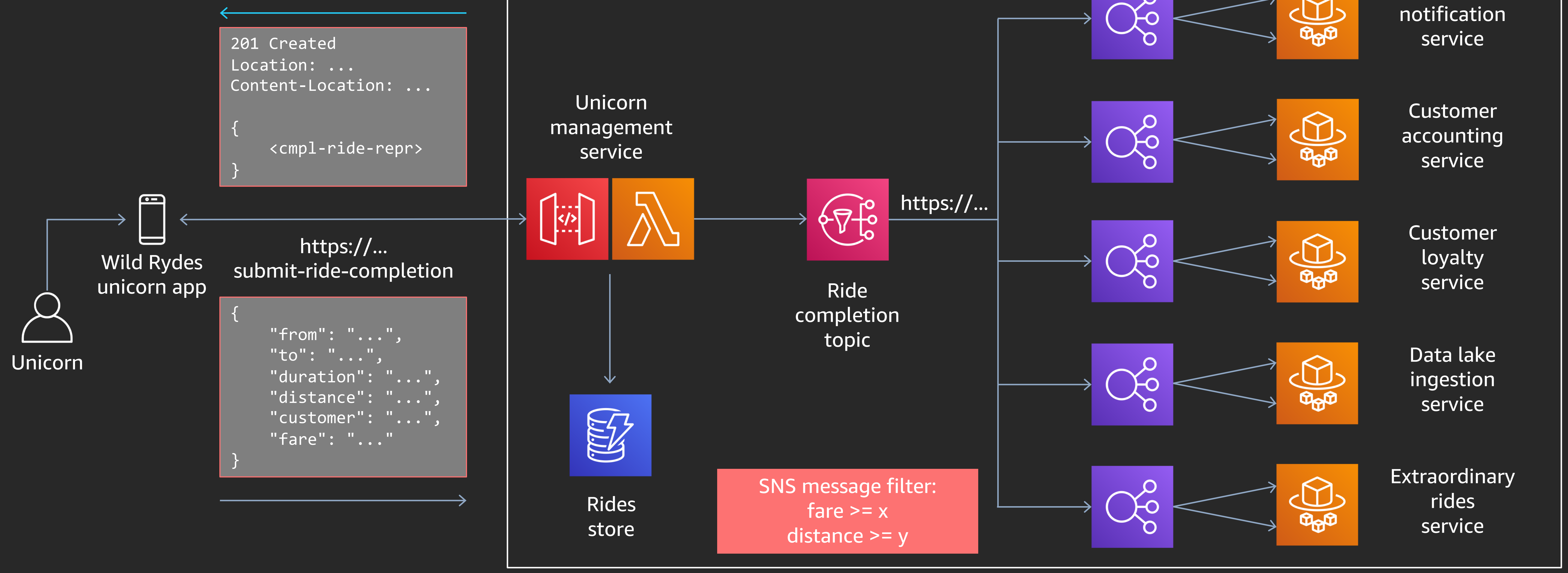


Labs summary

Use case: Submit a ride completion

Fan-out, message-filtering

Lab 1 architecture



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

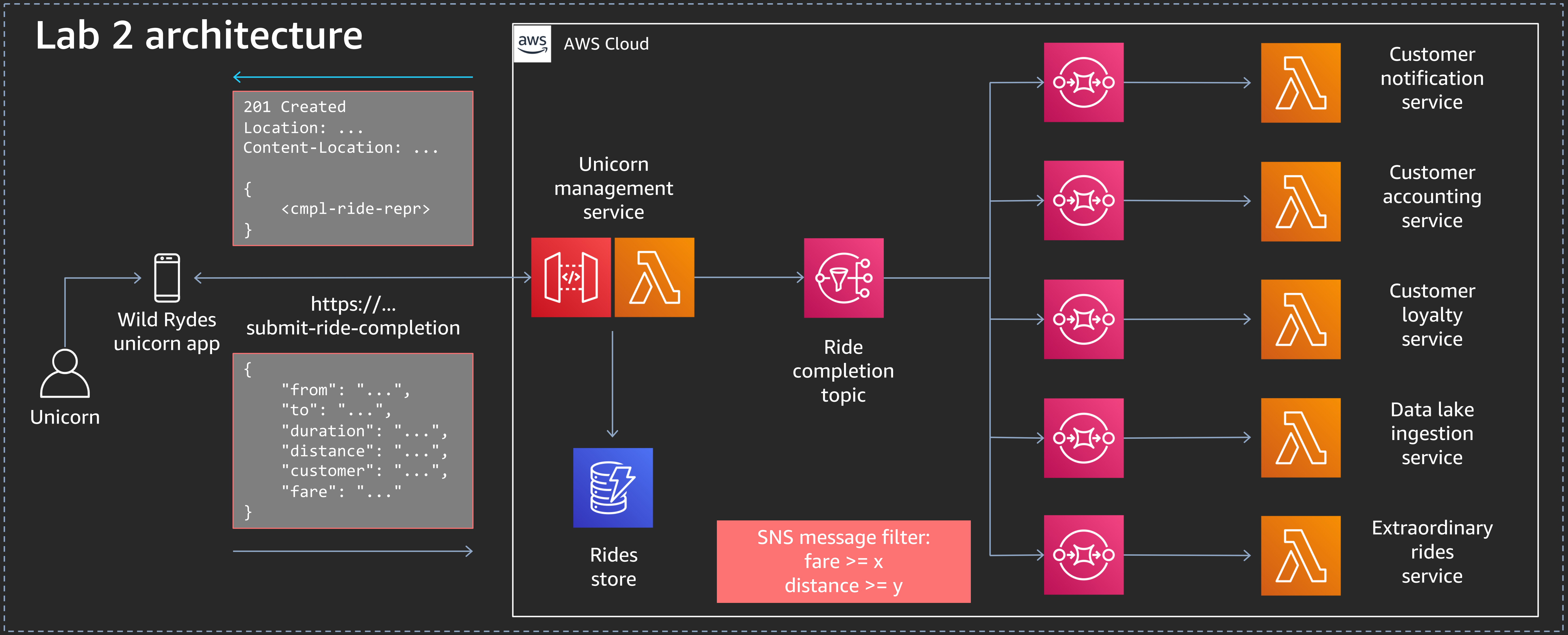
SNS message filter:
fare >= x
distance >= y

AWS Cloud
Unicorn
Wild Rydes unicorn app
https://... submit-ride-completion
Unicorn management service
Rides store
Ride completion topic
https://...
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

Use case: Submit a ride completion

Topic-queue-chaining, queues as buffering LBs

Lab 2 architecture



```
201 Created
Location: ...
Content-Location: ...

{
    <cmpl-ride-repr>
}
```

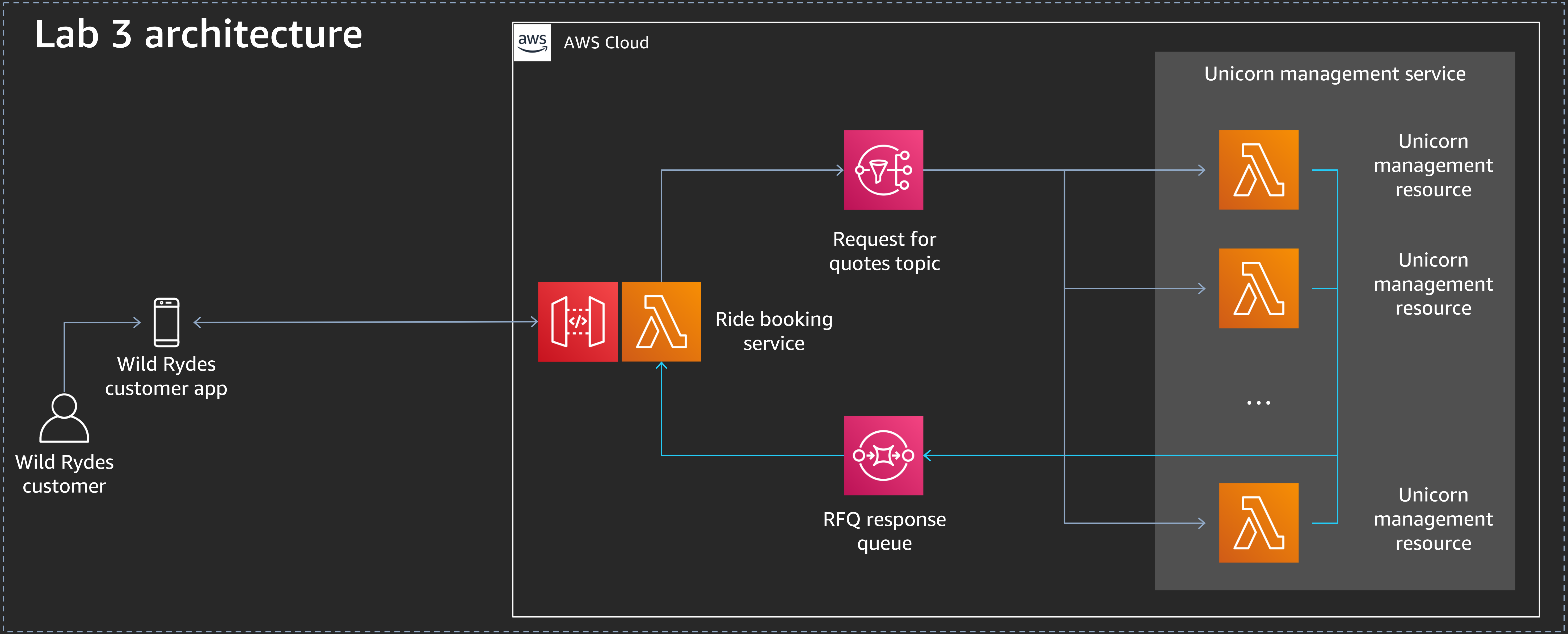
```
{
    "from": "...",
    "to": "...",
    "duration": "...",
    "distance": "...",
    "customer": "...",
    "fare": "..."
}
```

SNS message filter:
fare >= x
distance >= y

AWS Cloud
Unicorn
Wild Rydes unicorn app
https://...
submit-ride-completion
Unicorn management service
Rides store
Ride completion topic
Customer notification service
Customer accounting service
Customer loyalty service
Data lake ingestion service
Extraordinary rides service

Use case: Instant ride RFQ

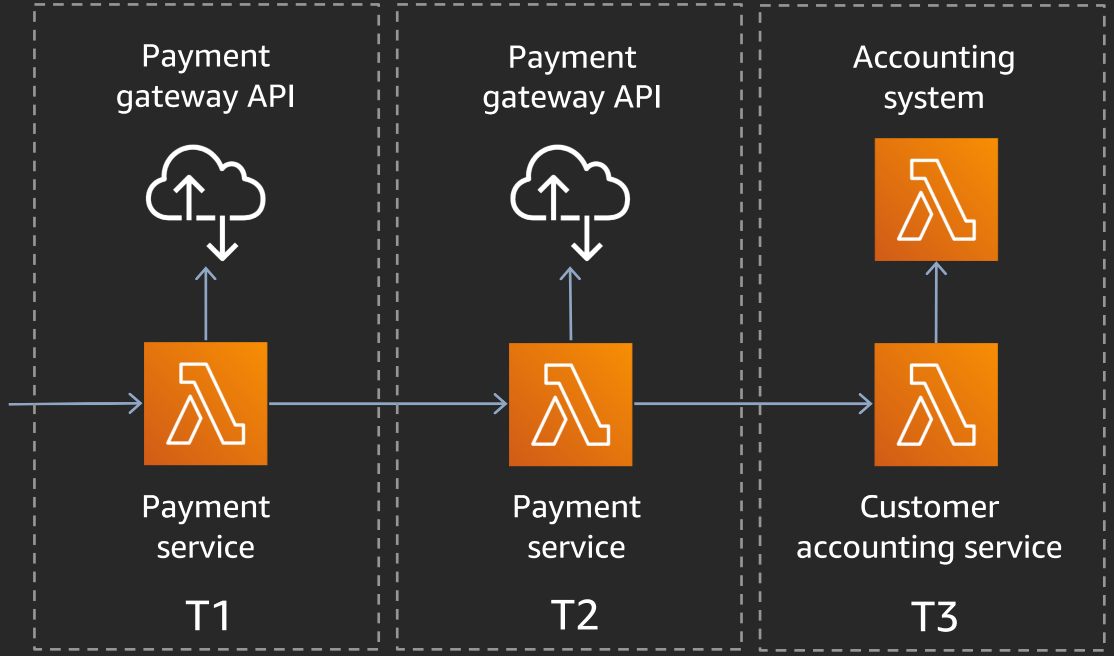
Scatter-gather



Use case: Fare collection

Saga orchestration

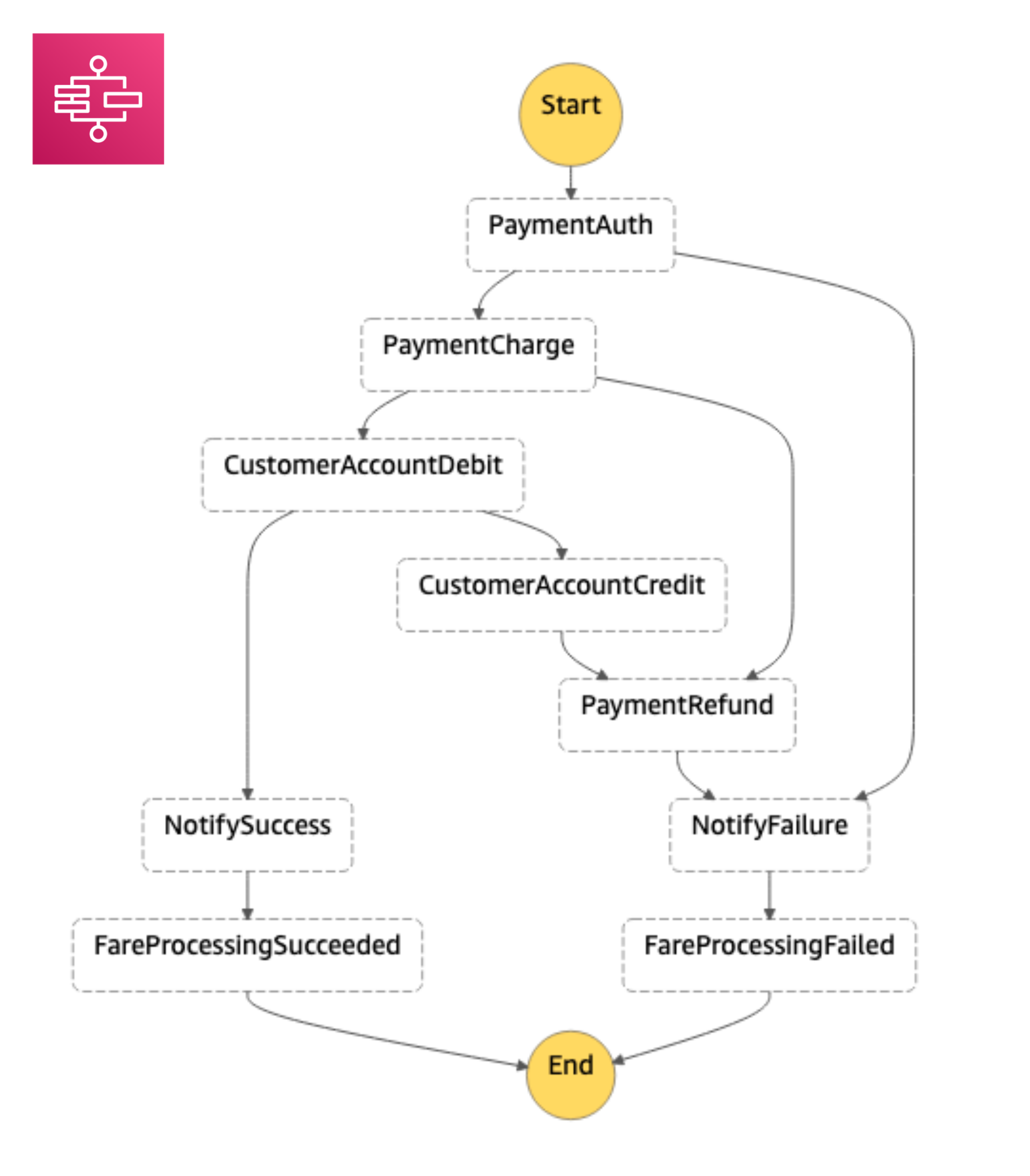
Saga orchestration



Discrete transactions:

1. Credit card pre-auth
2. Charge card using pre-auth code
3. Update customer account

To be treated as one distributed TA, to leave the systems in a semantically consistent state



Lab 0 / AWS Cloud9 overview

Your AWS accounts for today:

dashboard.eventengine.run

Now, choose your own adventure!

http://async-messaging.workshop.aws

Go build!

Resources/Call to action

Resources/Call-to-action

Resources/Call-to-action

AWS blogs and other content about application integration

- Find these resources linked from the lab guide website

Resources/Call-to-action

AWS blogs and other content about application integration

- Find these resources linked from the lab guide website

Hands-on workshop on implementing the patterns from this talk

- API315 during this re:Invent (Monday + Tuesday + Wednesday + Thursday)
- Ask your AWS SA for an application integration immersion day

Resources/Call-to-action

- AWS blogs and other content about application integration
  - Find these resources linked from the lab guide website
- Hands-on workshop on implementing the patterns from this talk
  - API315 during this re:Invent (Monday + Tuesday + Wednesday + Thursday)
  - Ask your AWS SA for an application integration immersion day
- Chalk talk about how to select the right app-int service
  - API312 during this re:Invent (Wednesday)

Resources/Call-to-action

- AWS blogs and other content about application integration
  - Find these resources linked from the lab guide website
- Hands-on workshop on implementing the patterns from this talk
  - API315 during this re:Invent (Monday + Tuesday + Wednesday + Thursday)
  - Ask your AWS SA for an application integration immersion day
- Chalk talk about how to select the right app-int service
  - API312 during this re:Invent (Wednesday)
- Keep in mind
  - Loose coupling is better than lousy coupling

Learn to architect with AWS Training and Certification

Resources created by the experts at AWS to propel your organization and career forward

Free foundational to advanced digital courses cover AWS services and teach architecting best practices

Classroom offerings, including Architecting on AWS, feature AWS expert instructors and hands-on labs

Validate expertise with the **AWS Certified Solutions Architect - Associate** or **AWS Certification Solutions Architect - Professional** exams

Visit aws.amazon.com/training/path-architecting/

Thank you!

Dirk Fröhner

froehner@amazon.de | @dirk_f5r

Christian Müller

cmr@amazon.de | @ChristianM

Please complete the session survey in the mobile app.

AWS re:Invent

aws