
AWS
re:Invent

API301

Application integration patterns for microservices

Dirk Fröhner

Principal Solutions Architect

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

API301

Application integration patterns (not only) for microservices

Dirk Fröhner

Principal Solutions Architect

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Introduction

Application integration patterns

Concrete use cases



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Introduction



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



"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

General-purpose internet-software geek

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Potential drawbacks of synchronous systems

Synchronous systems are tightly coupled.

Problems in downstream dependencies can have immediate impact on upstream callers.

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



Photo: Dirk Fröhner

Potential drawbacks of synchronous systems

Synchronous sy
coupled.

Problems in do
dependencies c
impact on upst

Retries from up
all-too easily fan-out and amplify
problems.

…and some things simply
take too much time to wait,
or are asynchronous by nature.



Photo: Dirk Fröhner

Application integration patterns



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

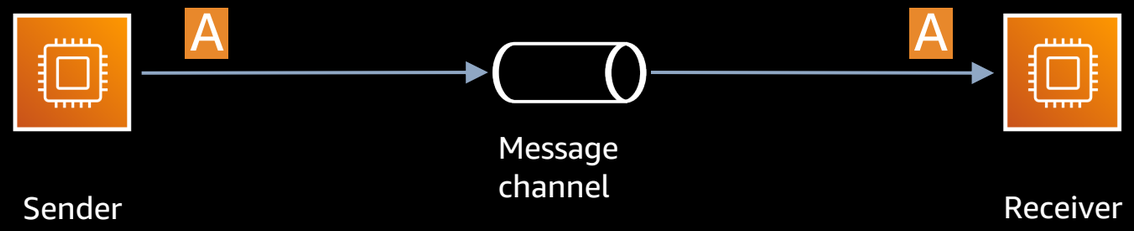
Message exchange

One-way

Request-response

Message exchange

One-way

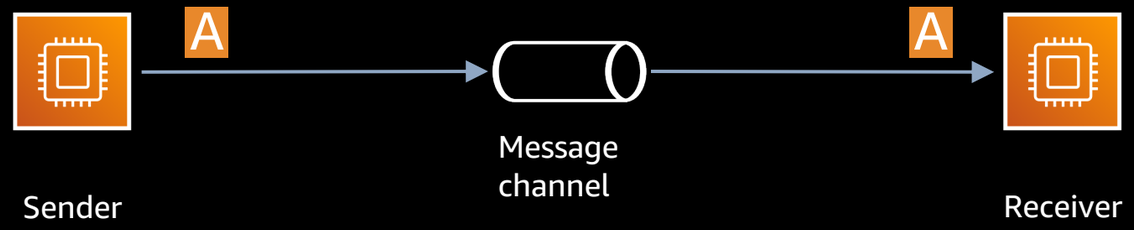


No response expected

Request-response

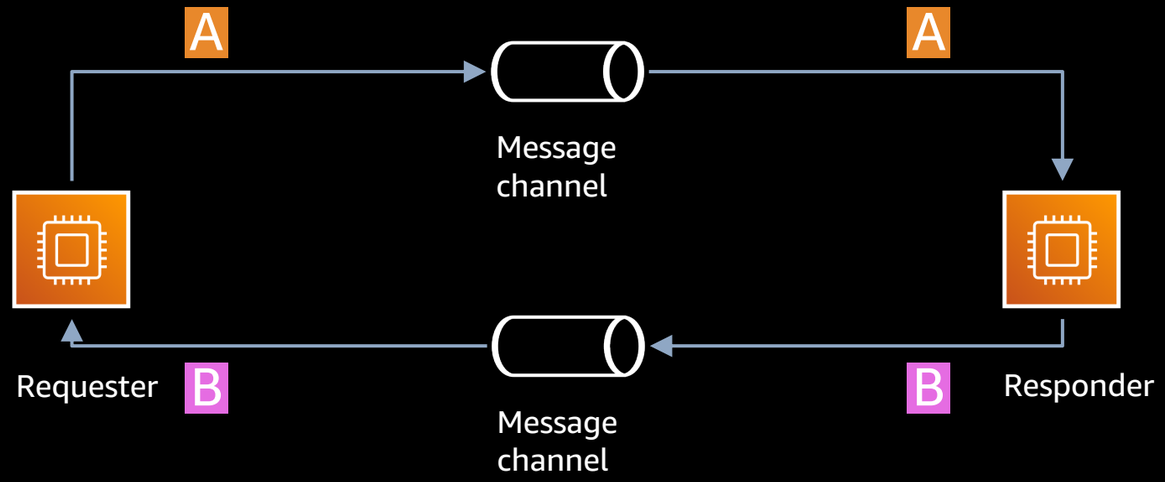
Message exchange

One-way



No response expected

Request-response



Response expected

Return address

Correlation ID

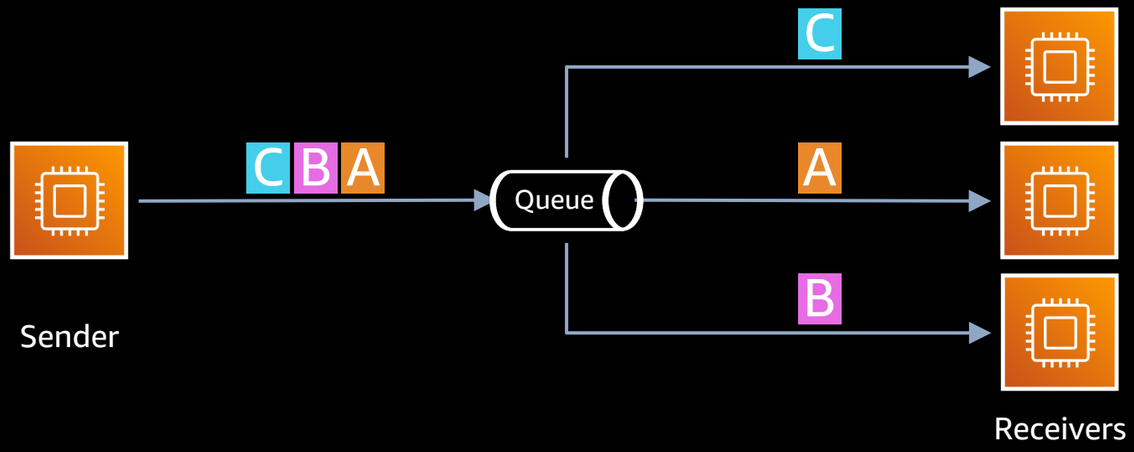
Message channels

Point-to-point (queue)

Publish-subscribe (topic)

Message channels

Point-to-point (queue)



Consumed by one receiver

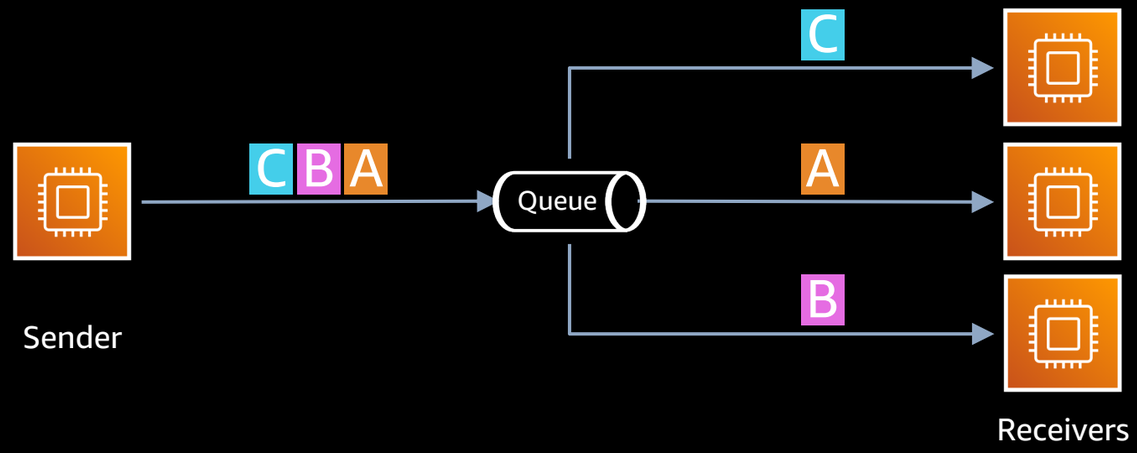
Easy to scale

Flatten peak loads

Publish-subscribe (topic)

Message channels

Point-to-point (queue)

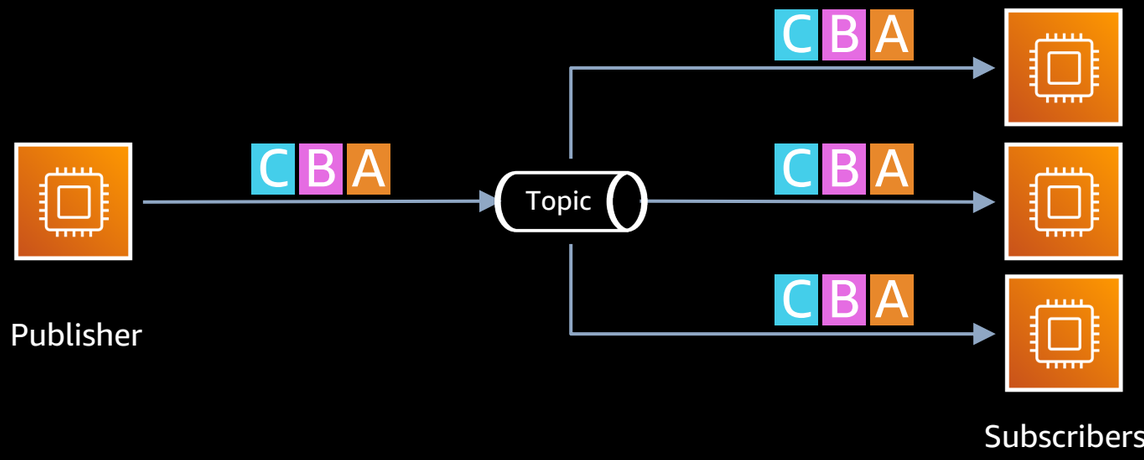


Consumed by one receiver

Easy to scale

Flatten peak loads

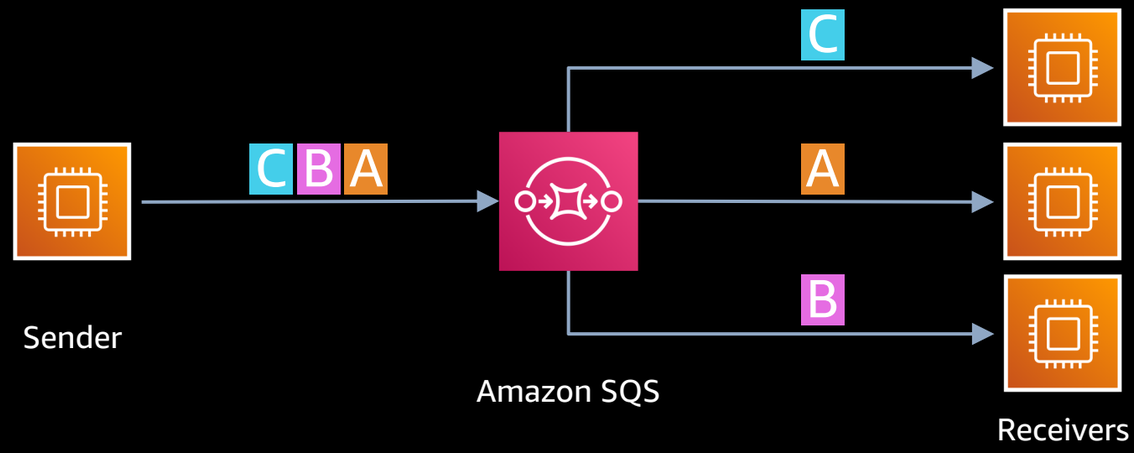
Publish-subscribe (topic)



Consumed by all subscribers

Message channels

Point-to-point (queue)

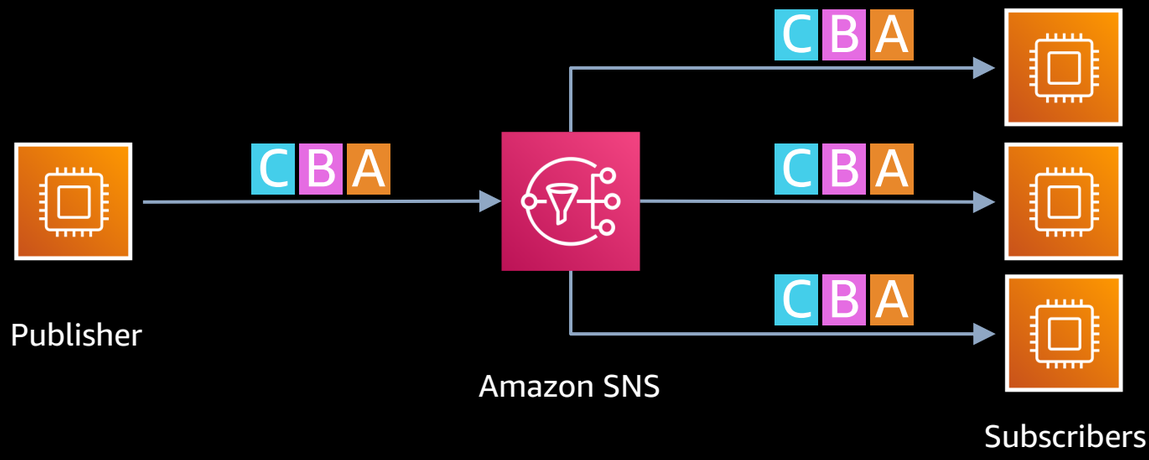


AWS service for queue functionality:

Amazon Simple Queue Service (SQS)

Serverless and cloud native

Publish-subscribe (topic)



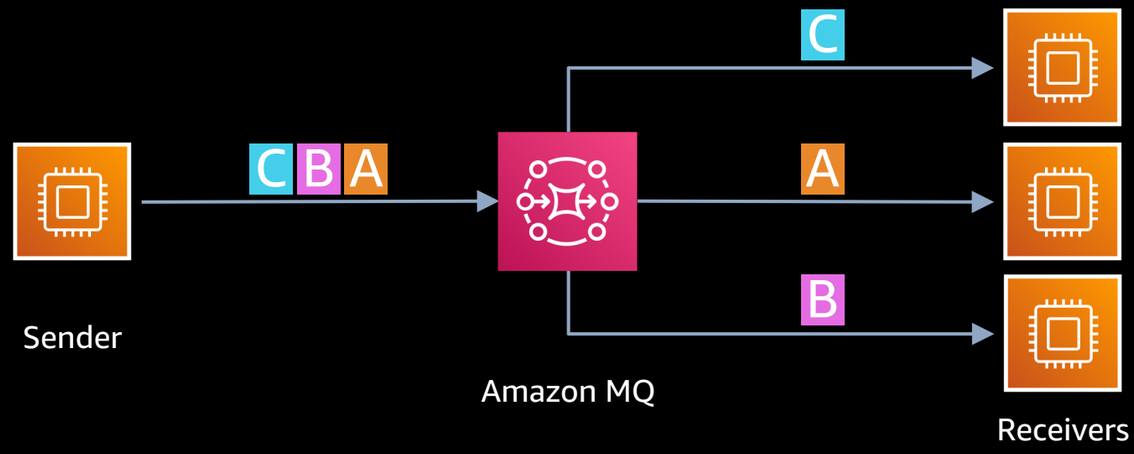
AWS service for topic functionality:

Amazon Simple Notification Service (SNS)

Serverless and cloud native

Message channels

Point-to-point (queue)

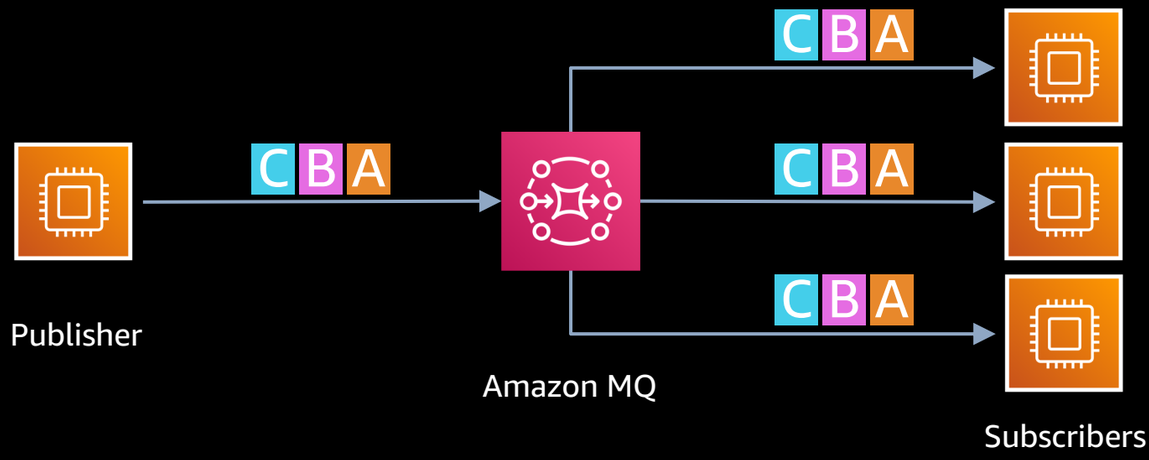


AWS service for queue functionality (non-serverless):

Amazon MQ (managed Apache Active MQ or RabbitMQ)

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 or RabbitMQ)

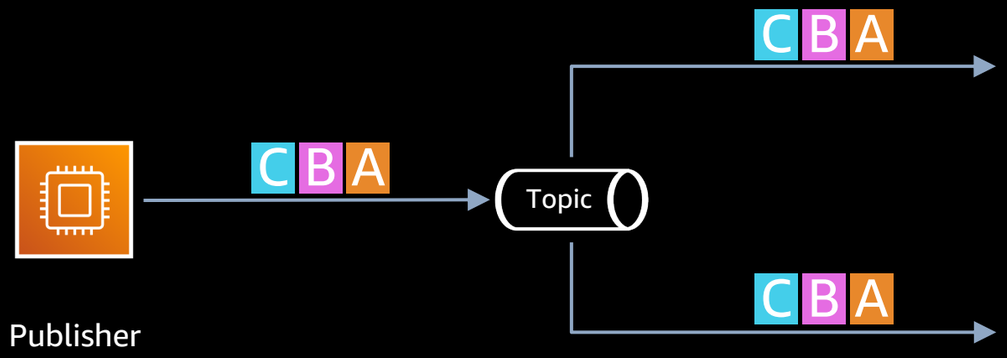
For apps constrained to protocols like JMS, AMQP, etc.

Message channels

Topic-queue-chaining

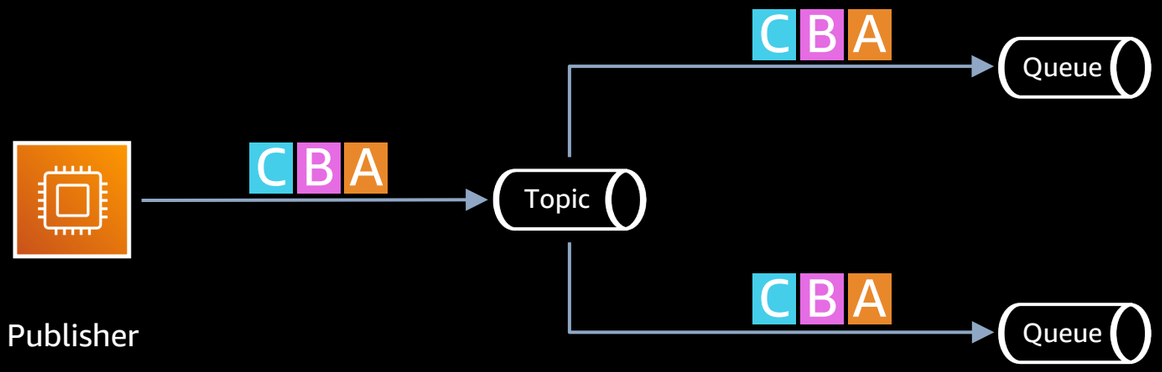
Message channels

Topic-queue-chaining

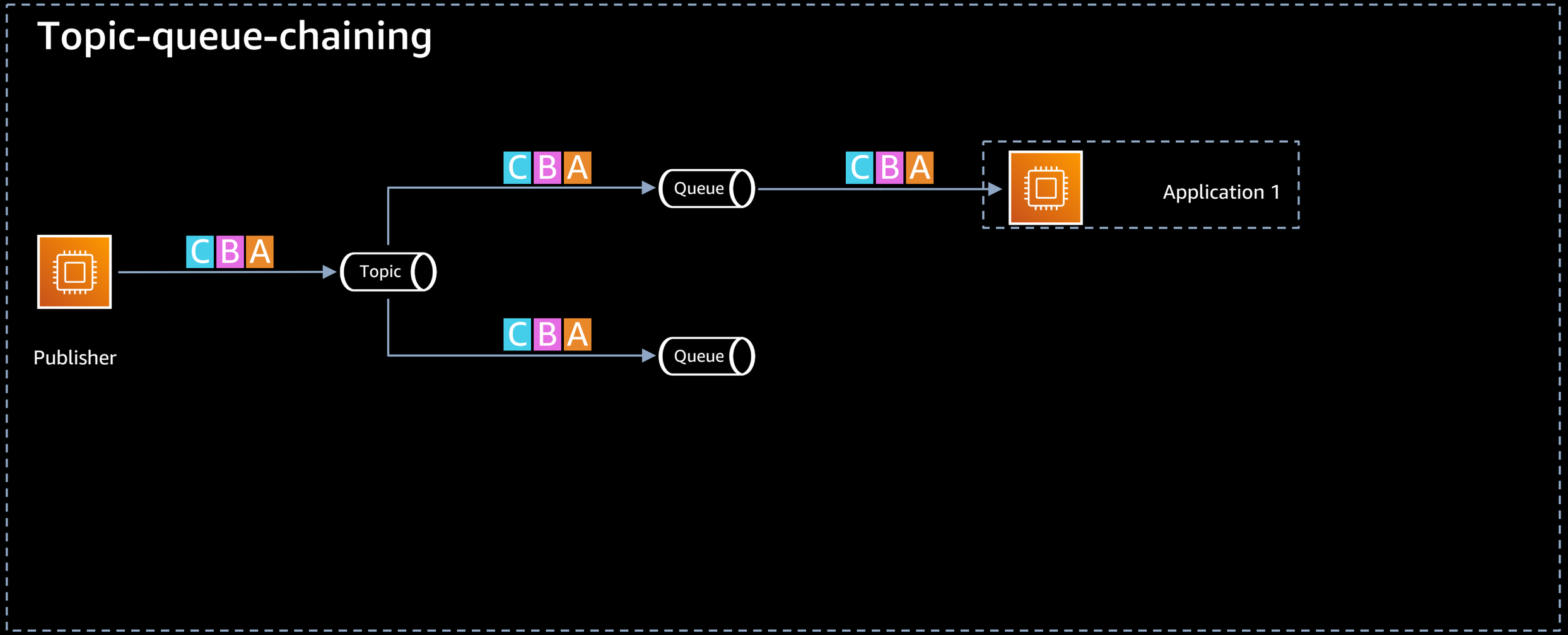


Message channels

Topic-queue-chaining

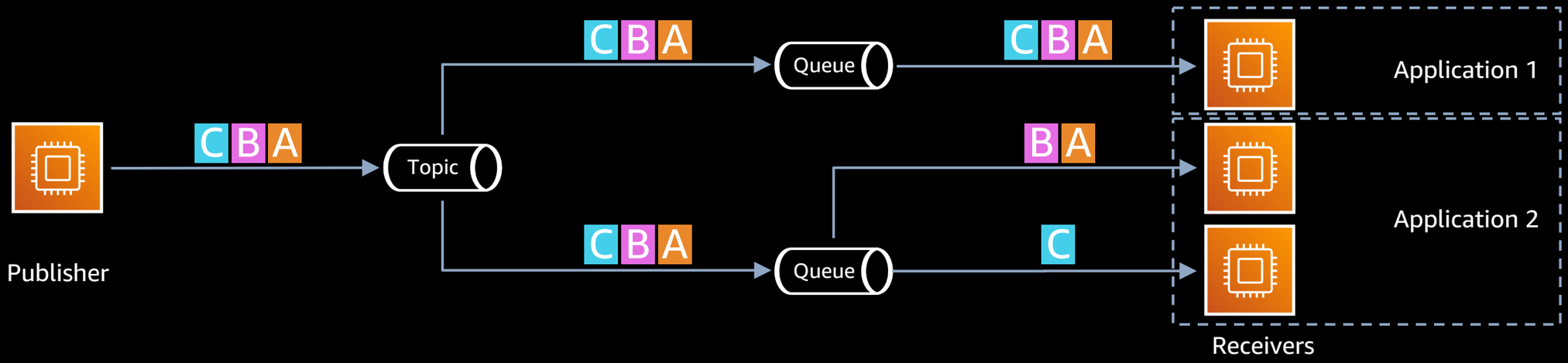


Message channels



Message channels

Topic-queue-chaining



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

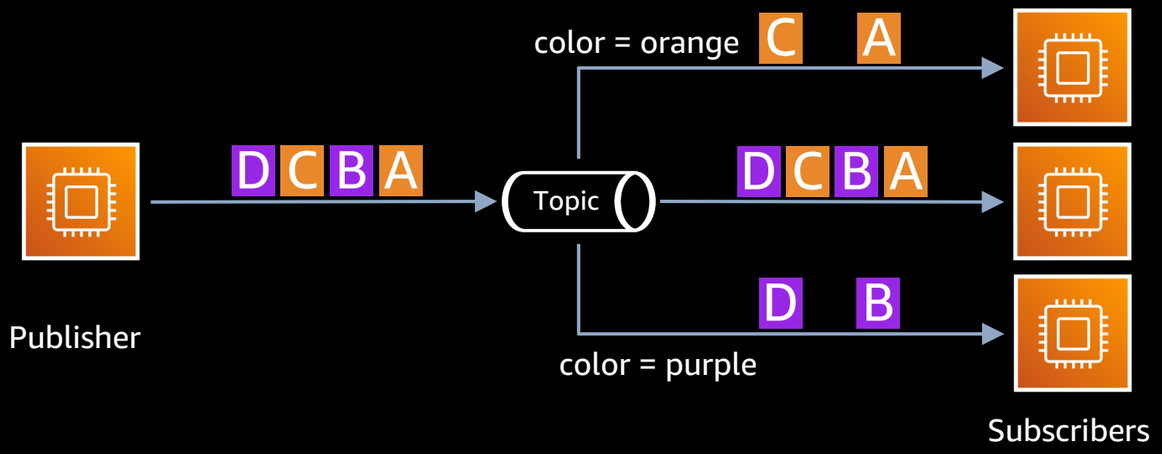
Message routing

Message filter

Recipient list

Message routing

Message filter



Receive only a relevant subset of messages

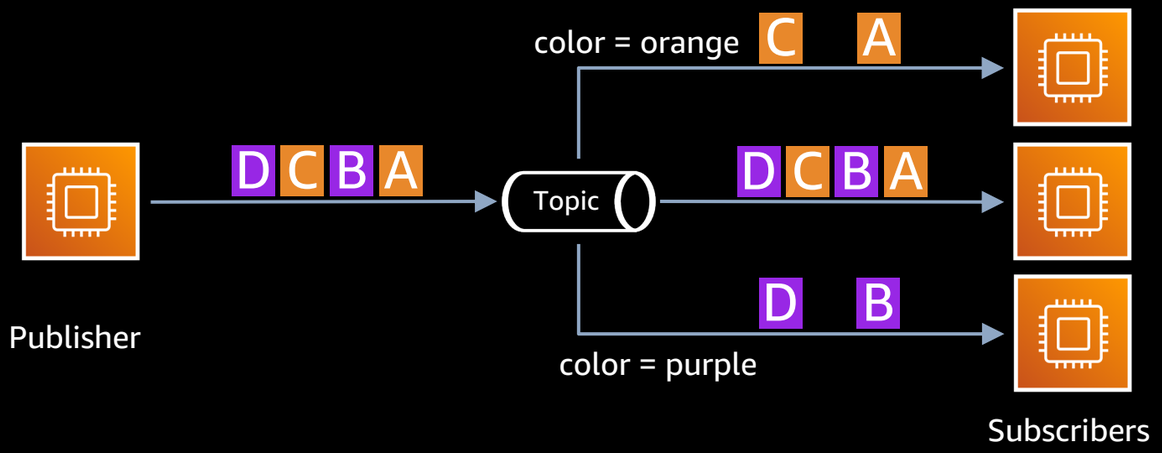
Controlled by subscriber

Publisher remains completely unaware

Recipient list

Message routing

Message filter

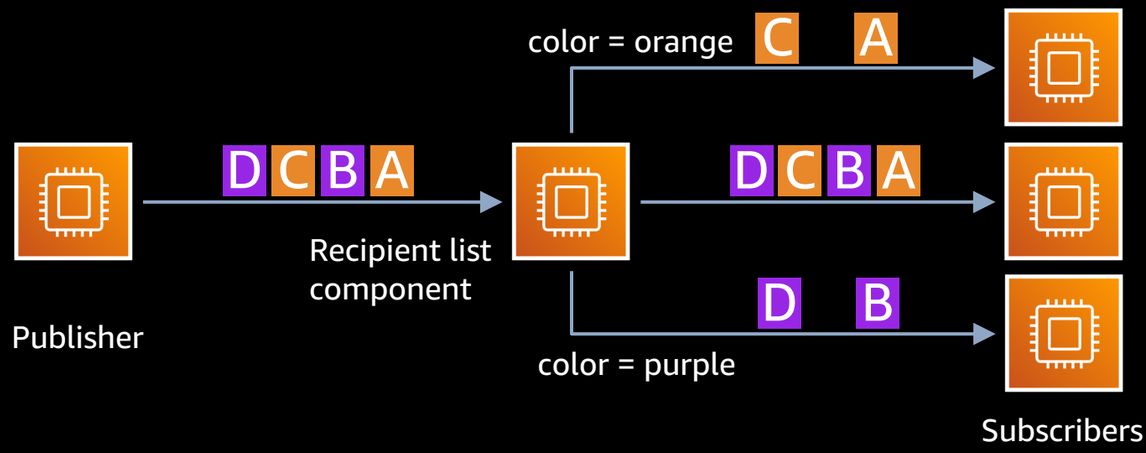


Receive only a relevant subset of messages

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

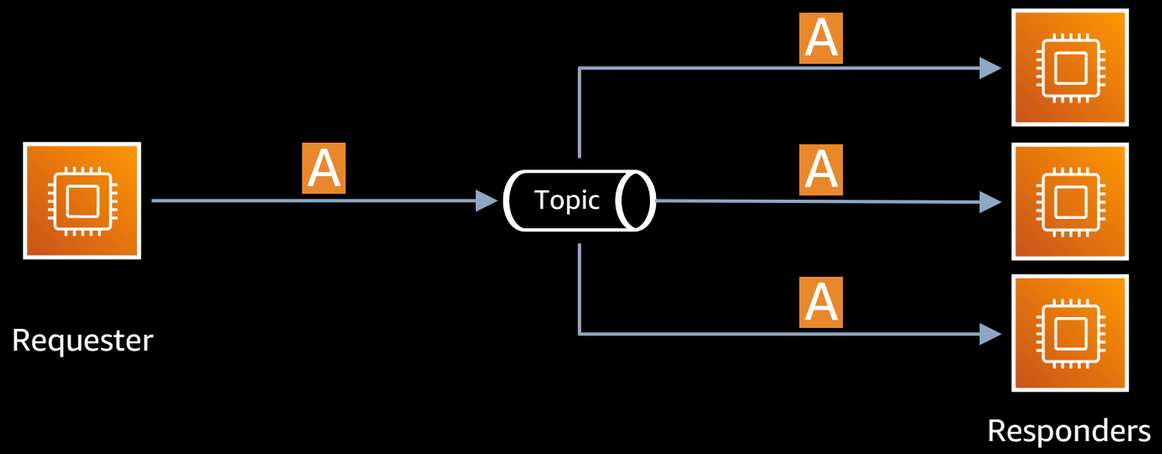
Scatter-gather

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

Election or parallel processing scenarios, i.e., search for best response or accumulate responses

Message routing

Scatter-gather

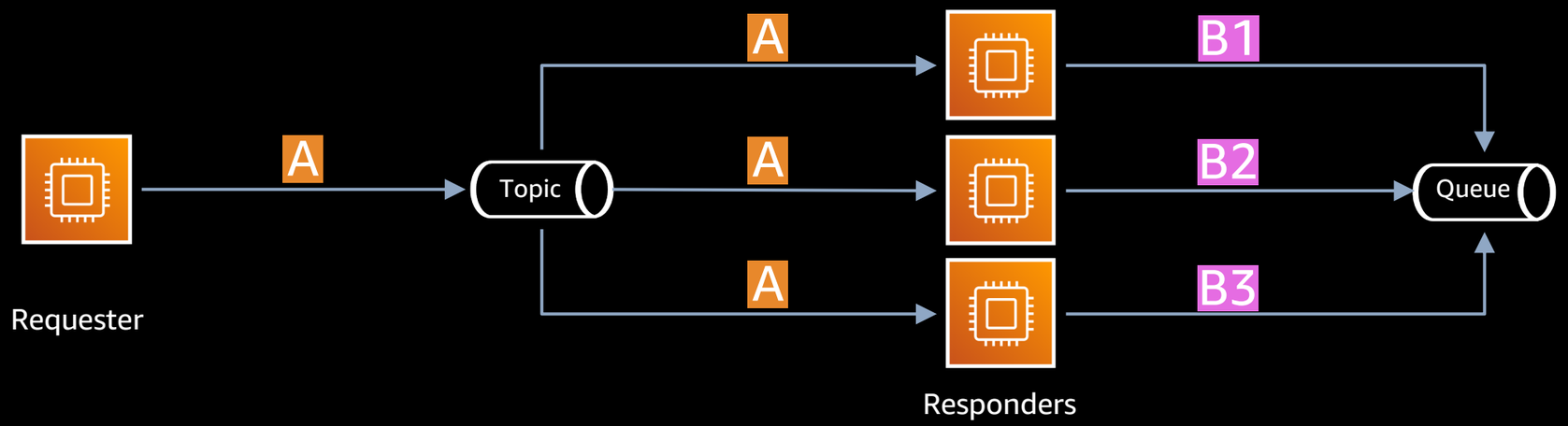


How to distribute a request across potentially interested/relevant parties and capture their individual responses?

Election or parallel processing scenarios, i.e., search for best response or accumulate responses

Message routing

Scatter-gather

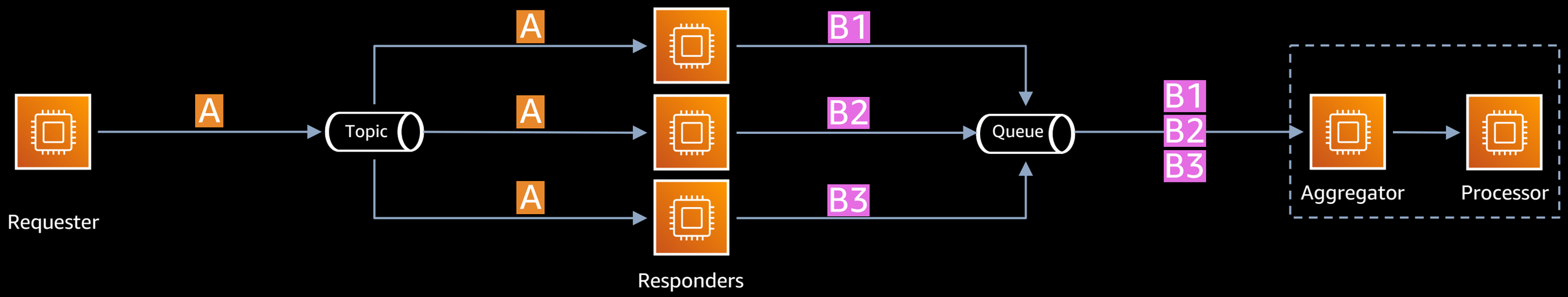


How to distribute a request across potentially interested/relevant parties and capture their individual responses?

Election or parallel processing scenarios, i.e., search for best response or accumulate responses

Message routing

Scatter-gather



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

Election or parallel processing scenarios, i.e., search for best response or accumulate responses

Concrete use cases



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Context: Wild Rydes, Inc.

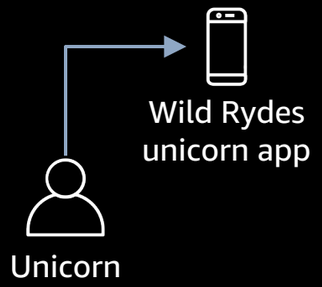


Use case: Submit a ride completion

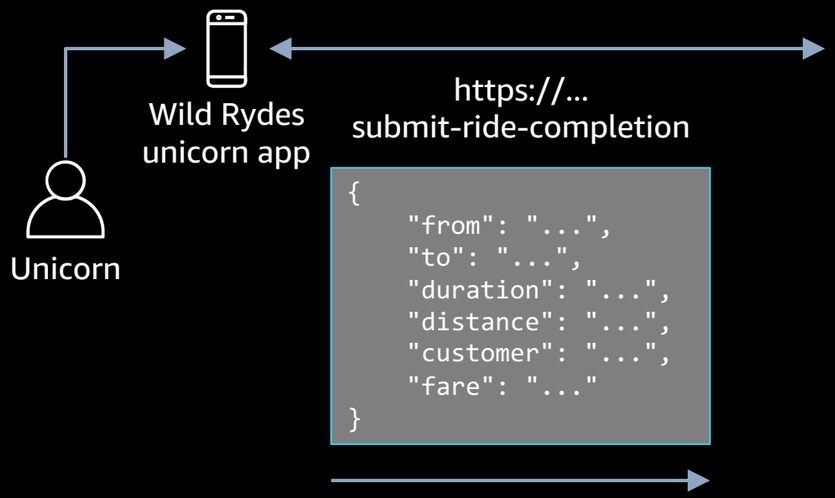


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

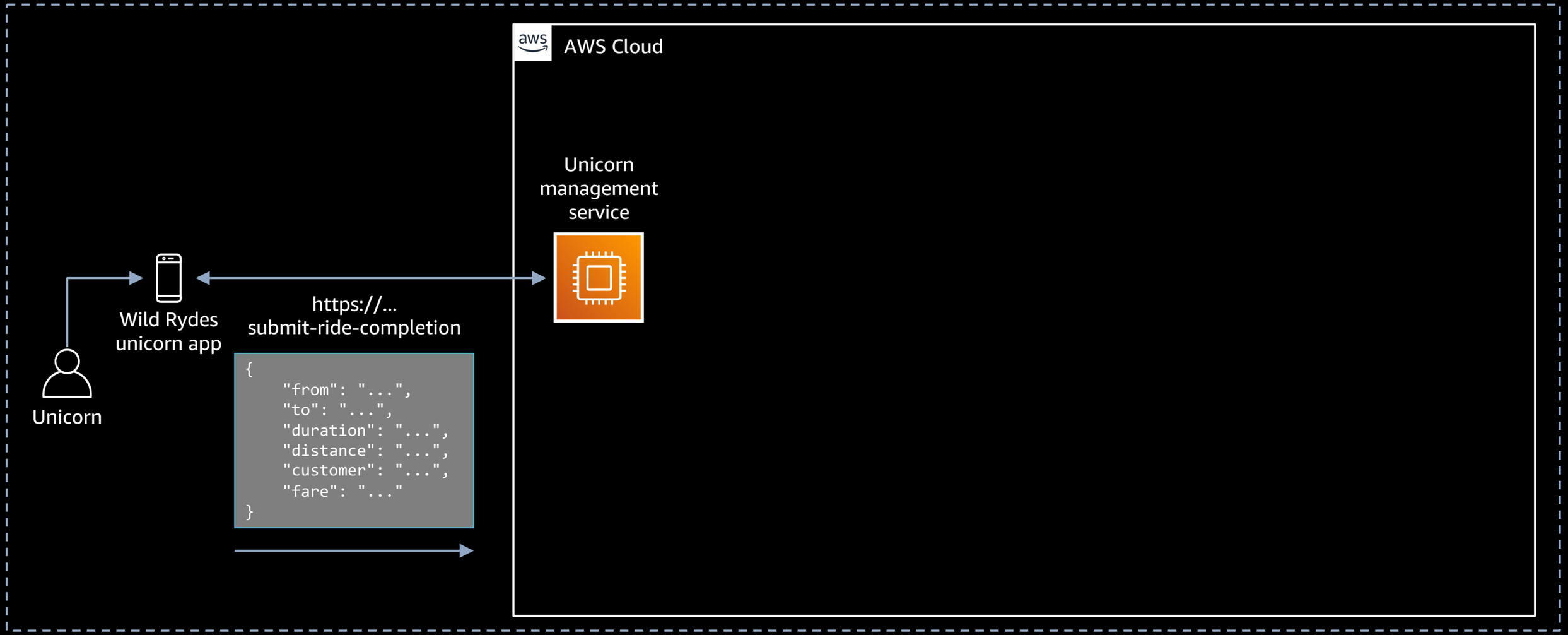
Use case: Submit a ride completion



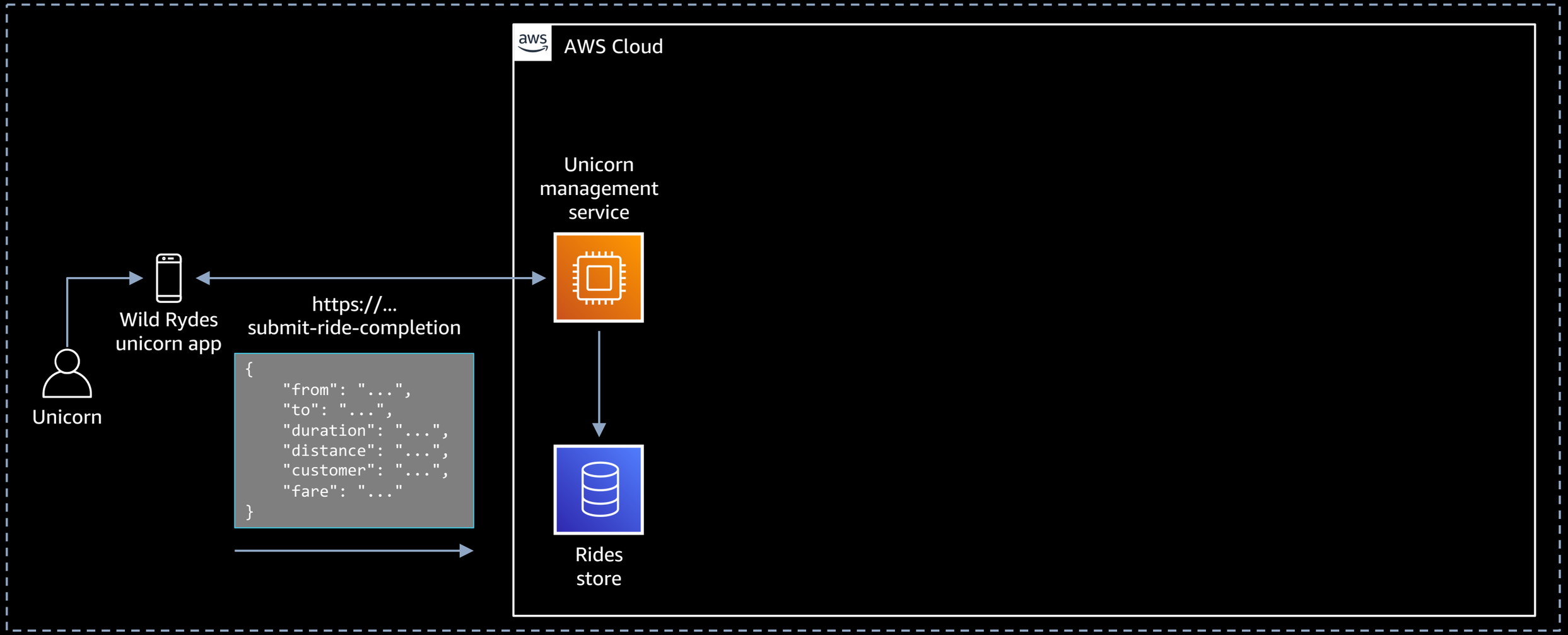
Use case: Submit a ride completion



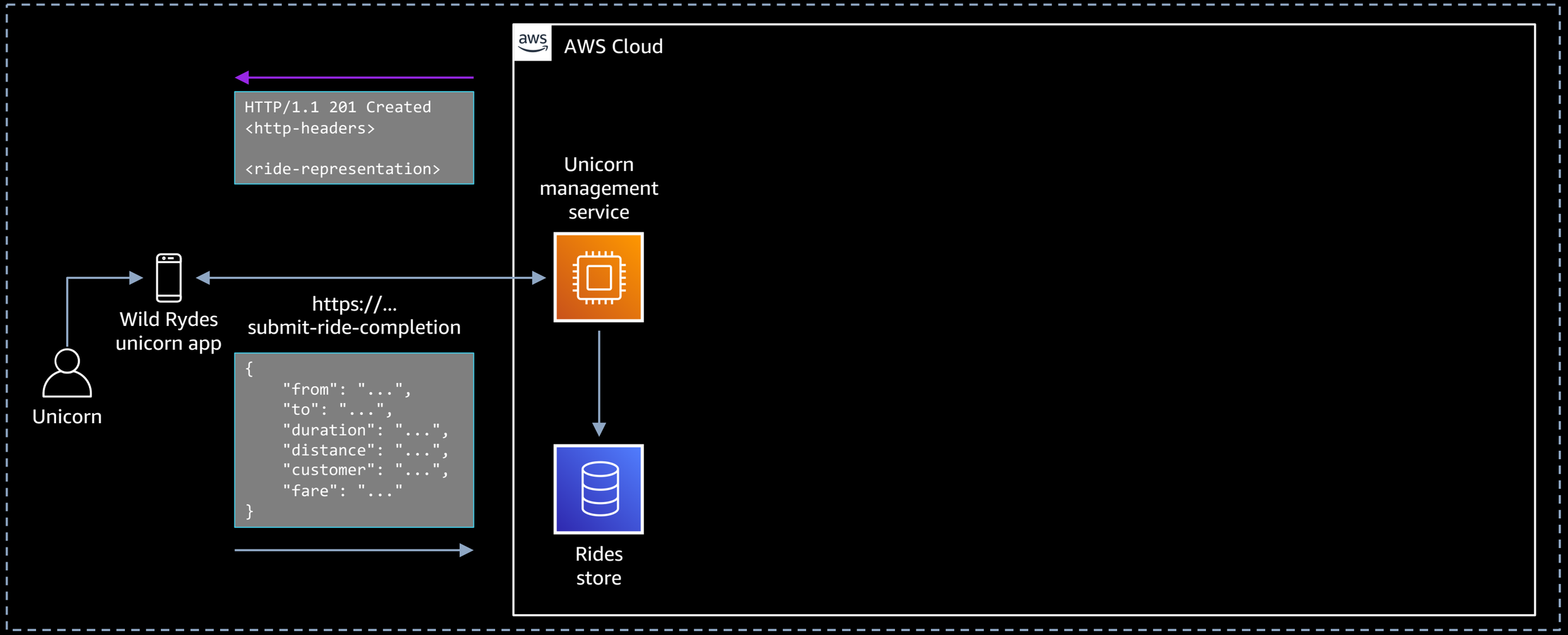
Use case: Submit a ride completion



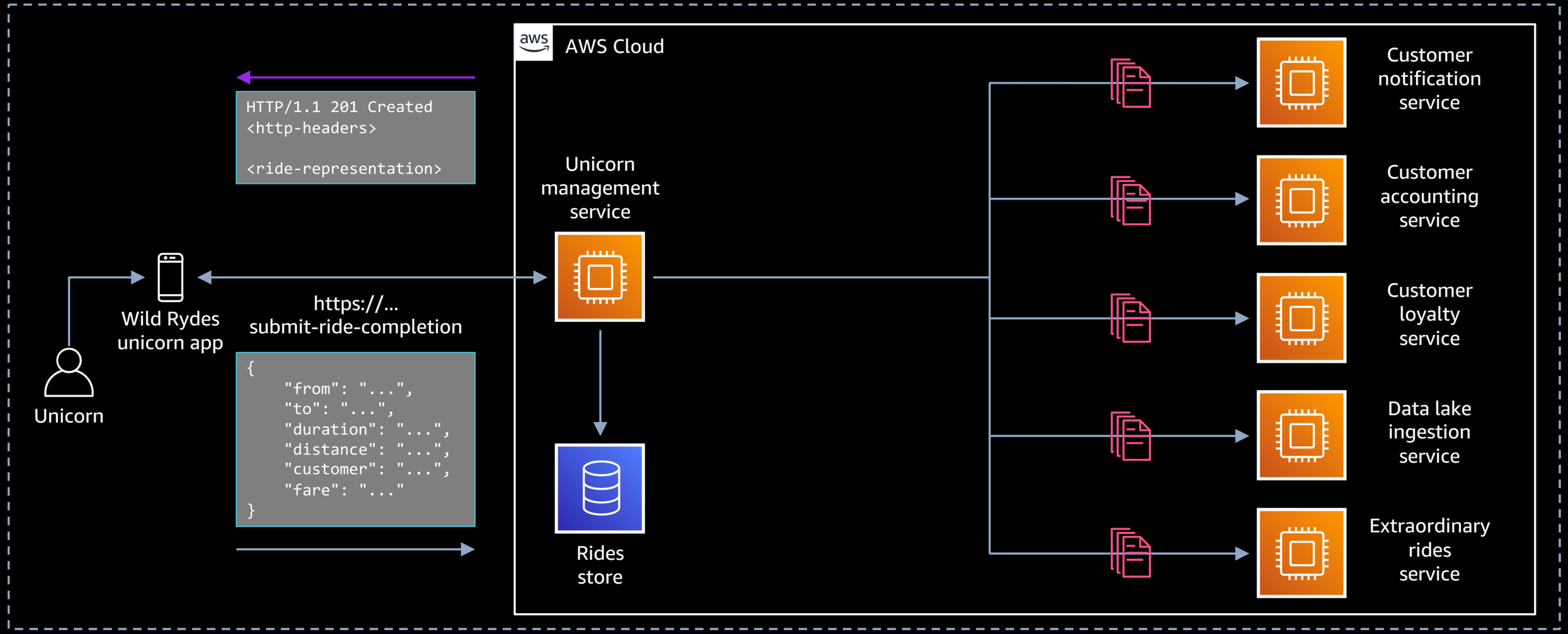
Use case: Submit a ride completion



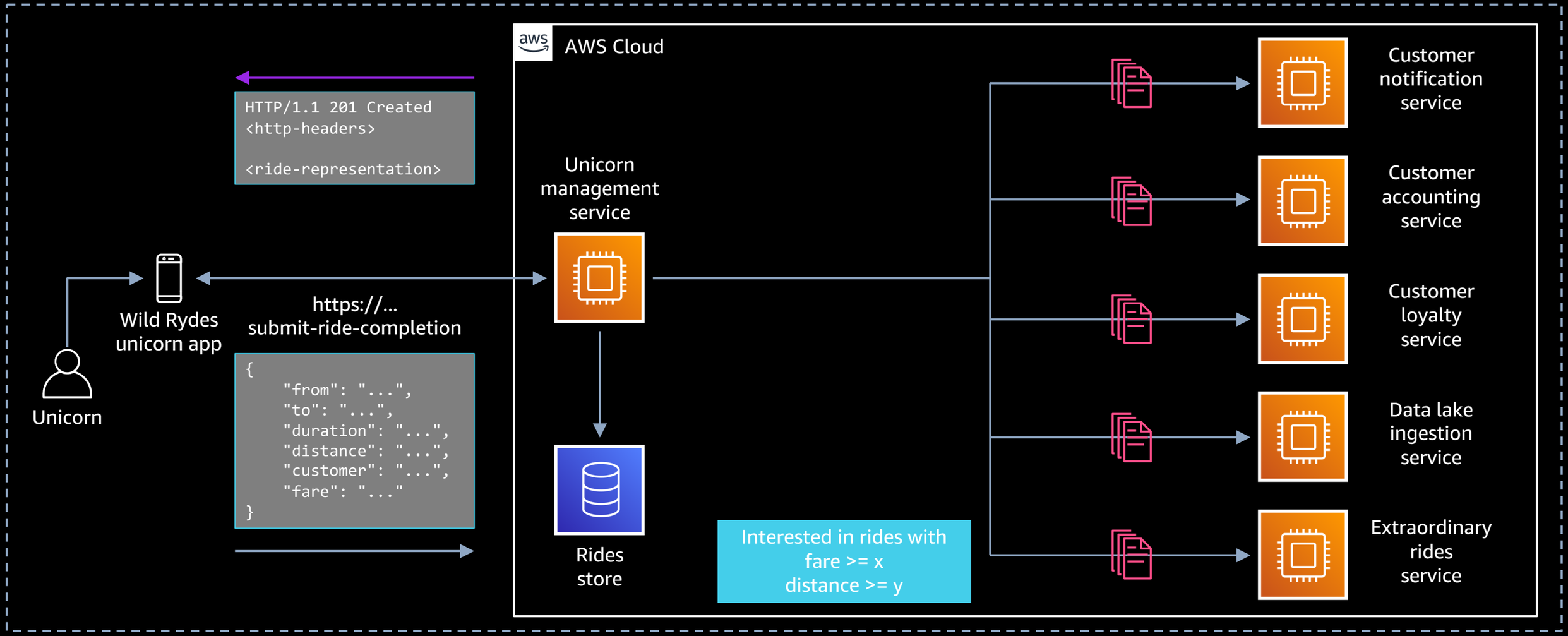
Use case: Submit a ride completion



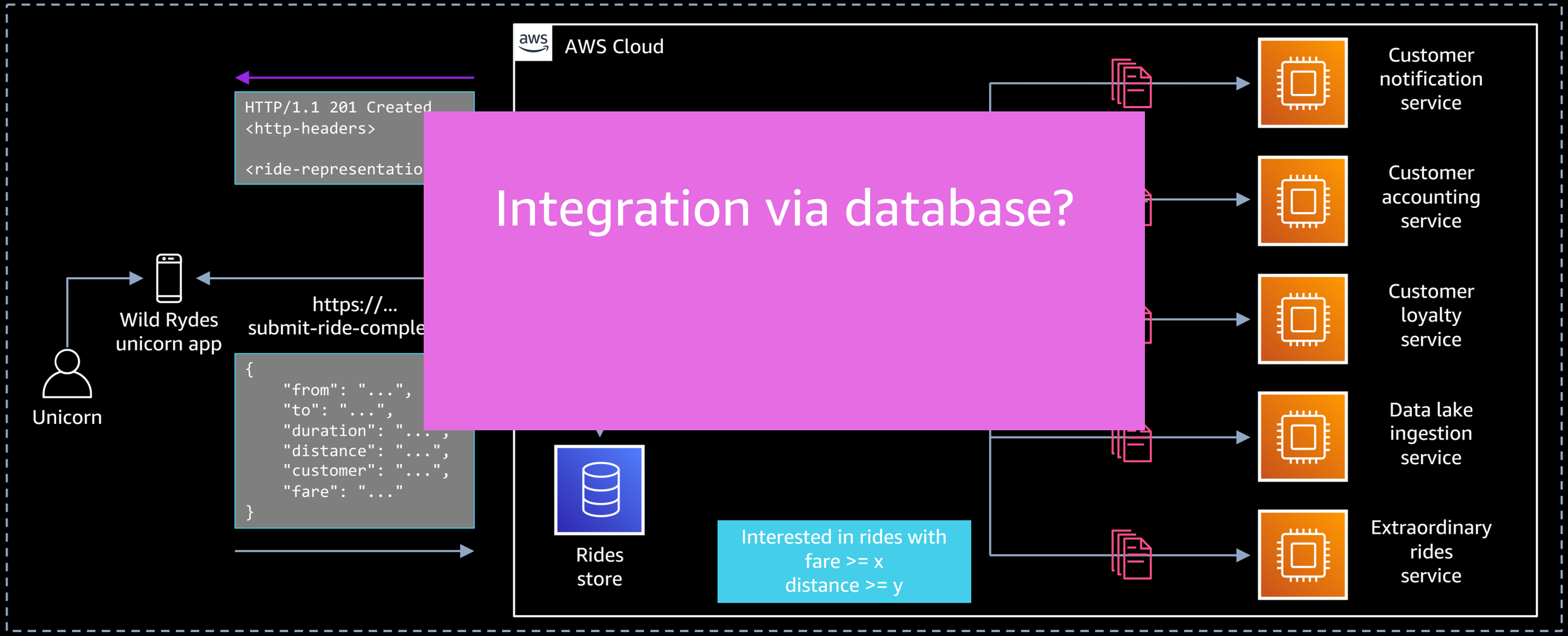
Use case: Submit a ride completion



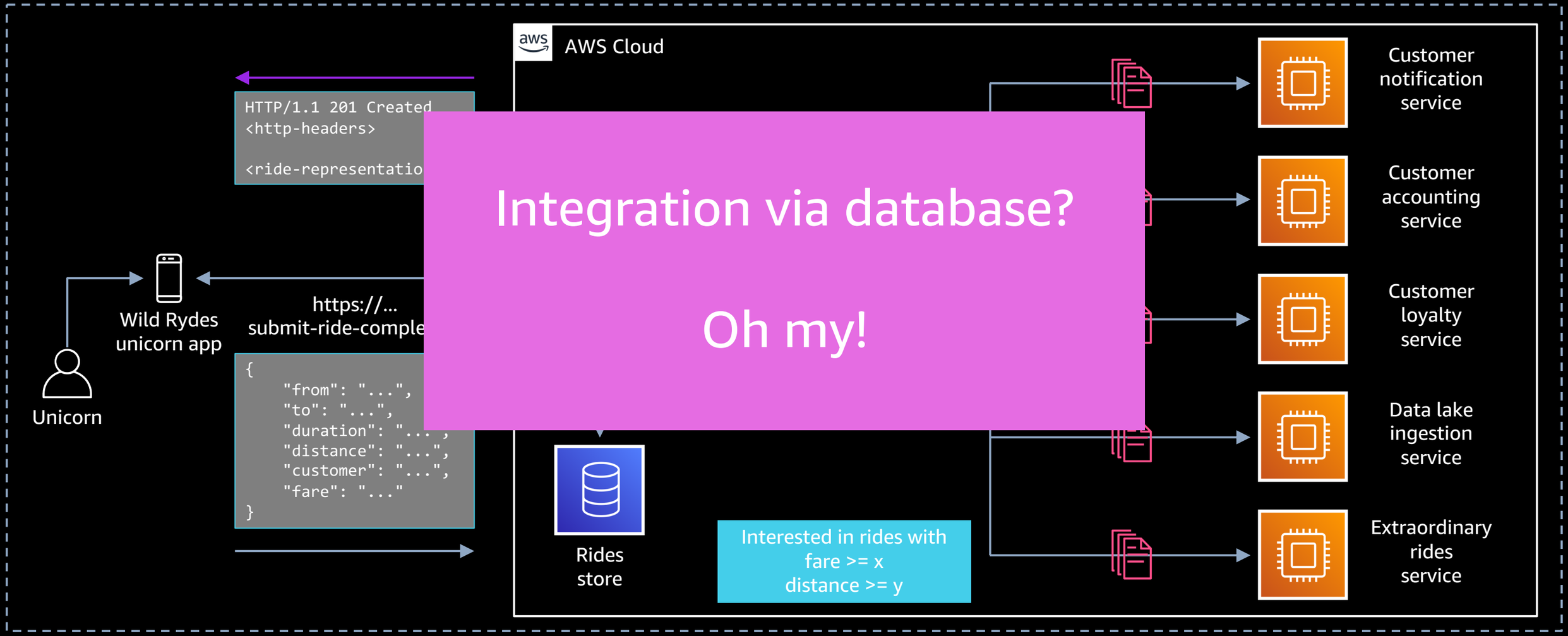
Use case: Submit a ride completion



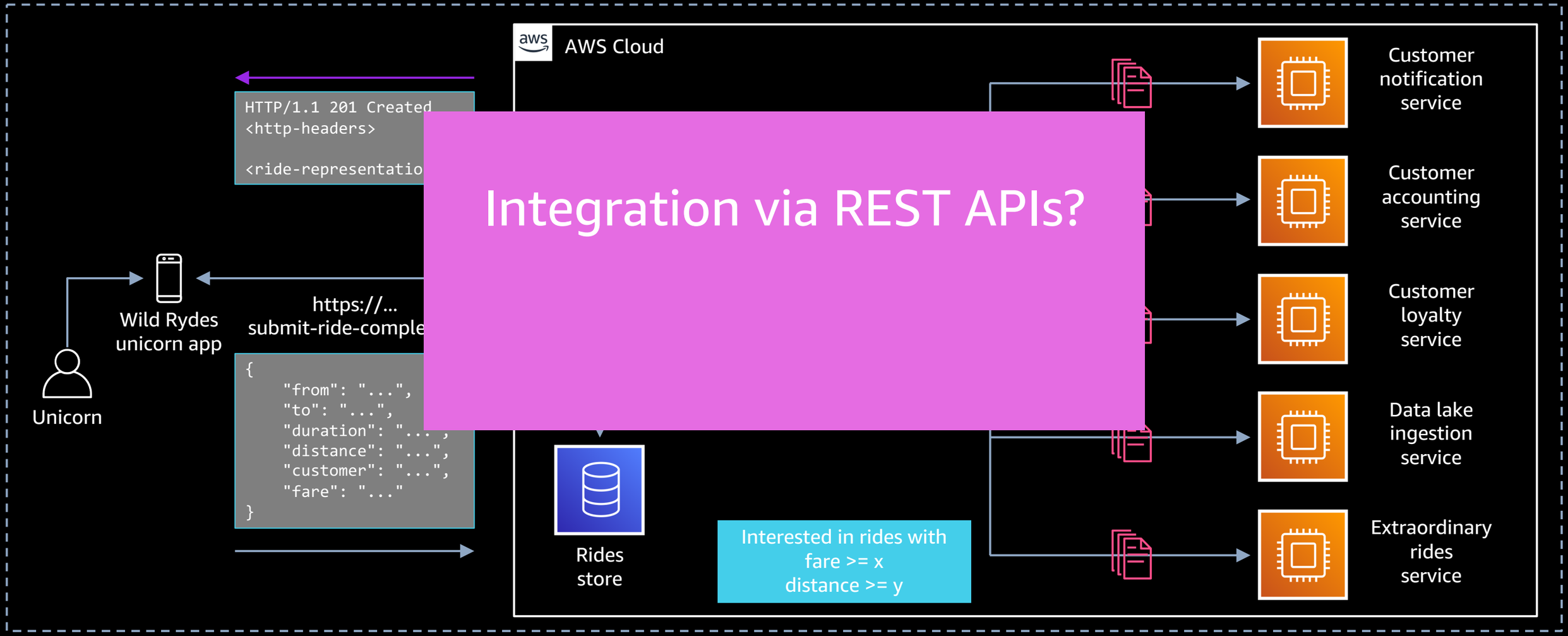
Use case: Submit a ride completion



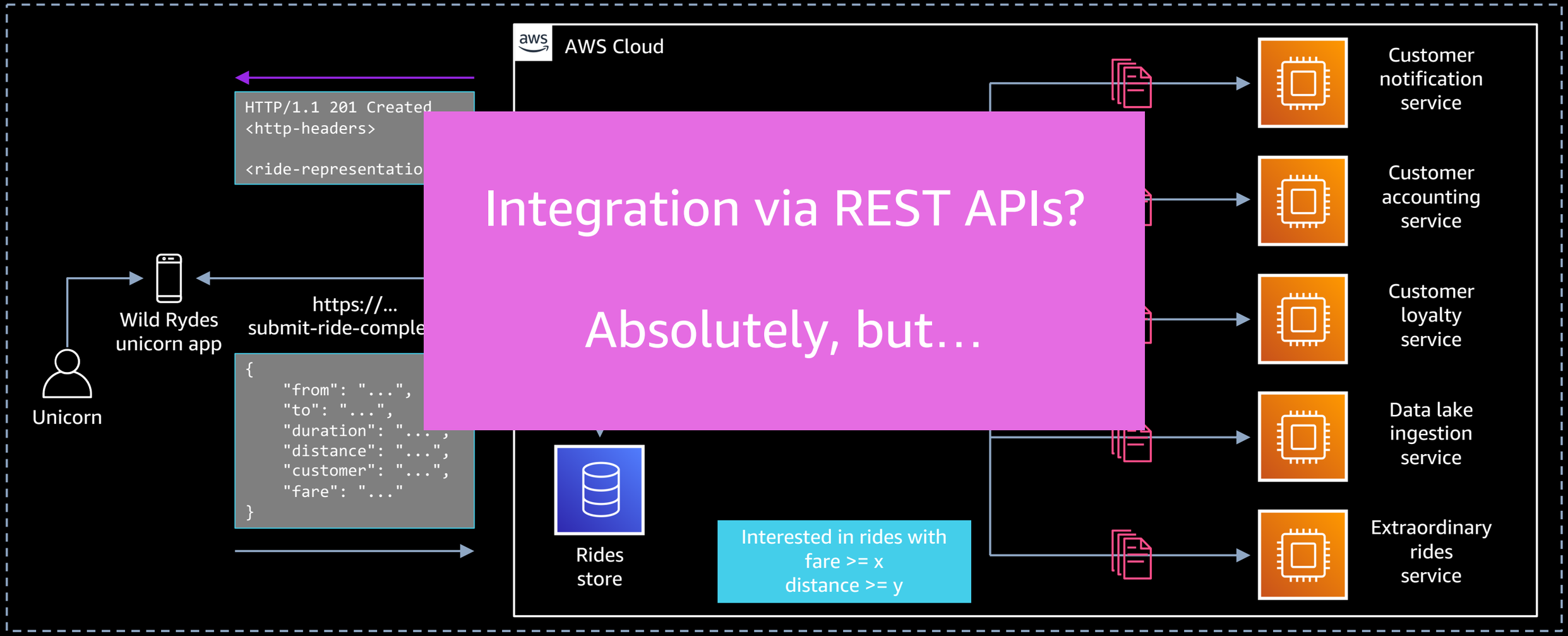
Use case: Submit a ride completion



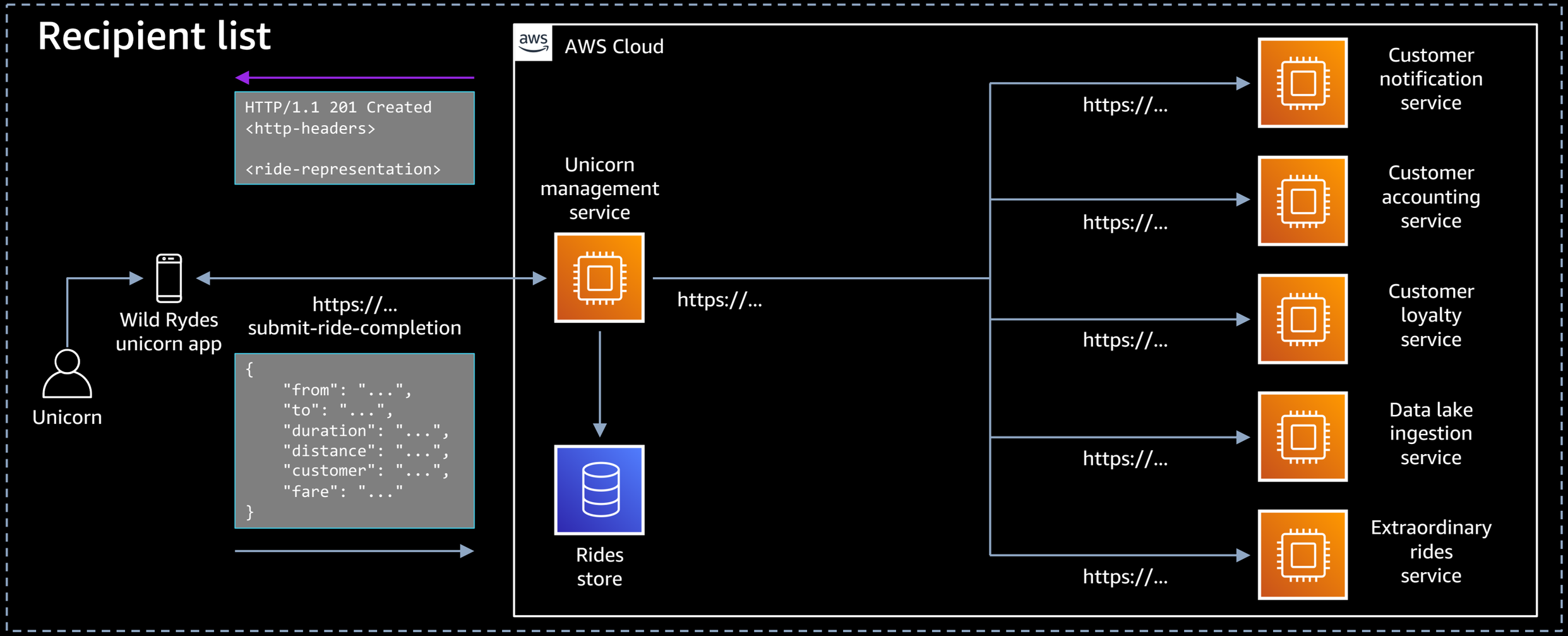
Use case: Submit a ride completion



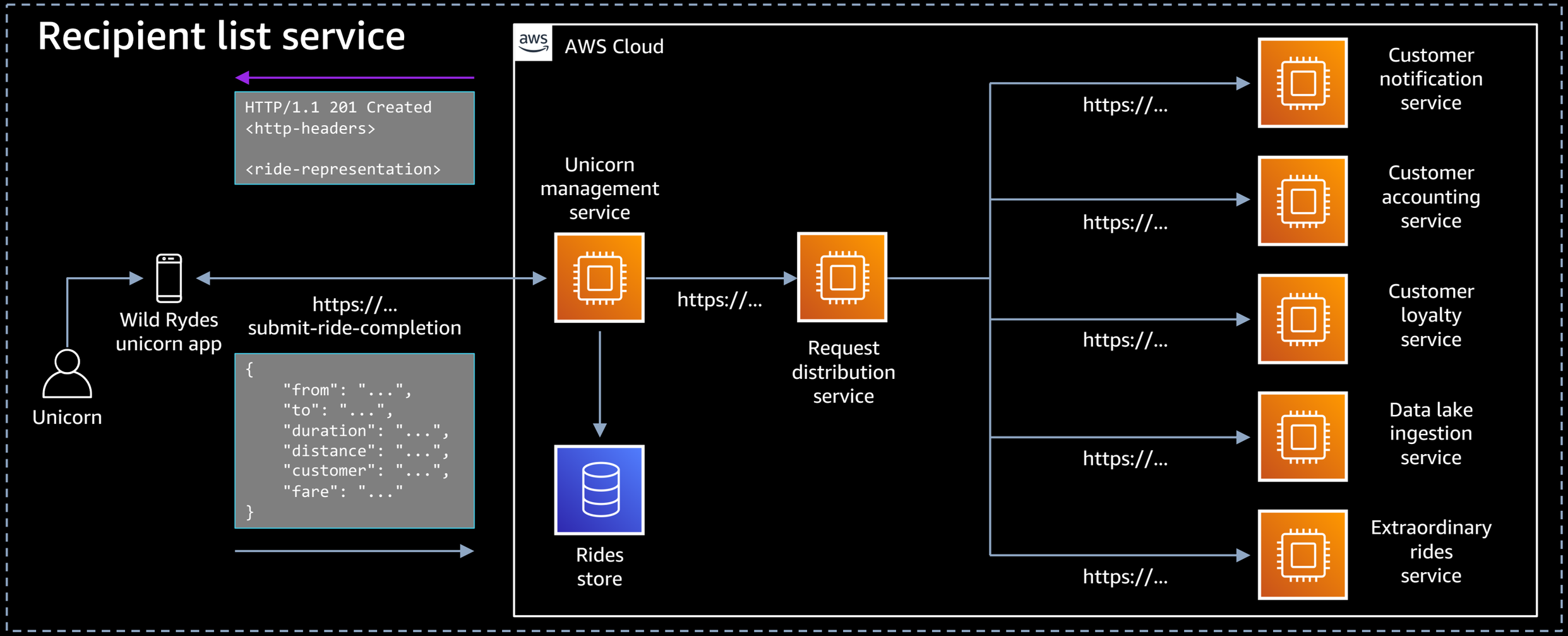
Use case: Submit a ride completion



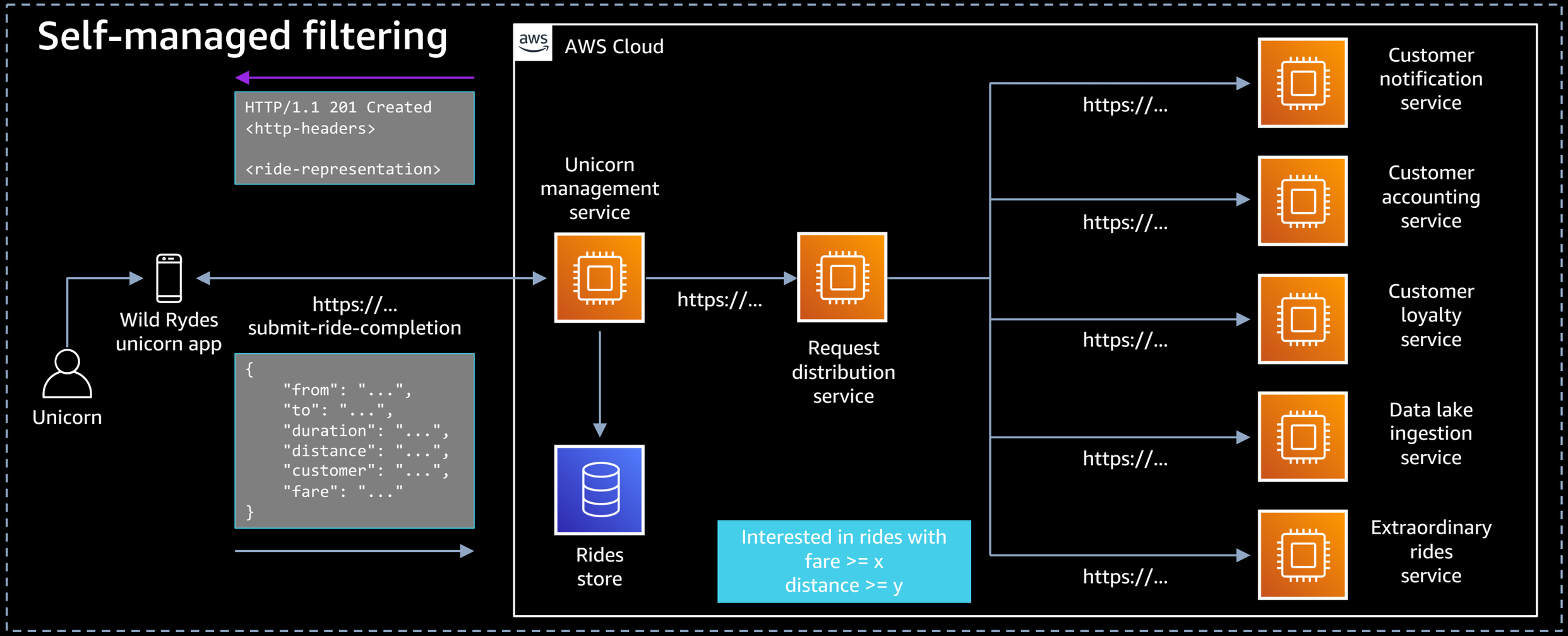
Use case: Submit a ride completion



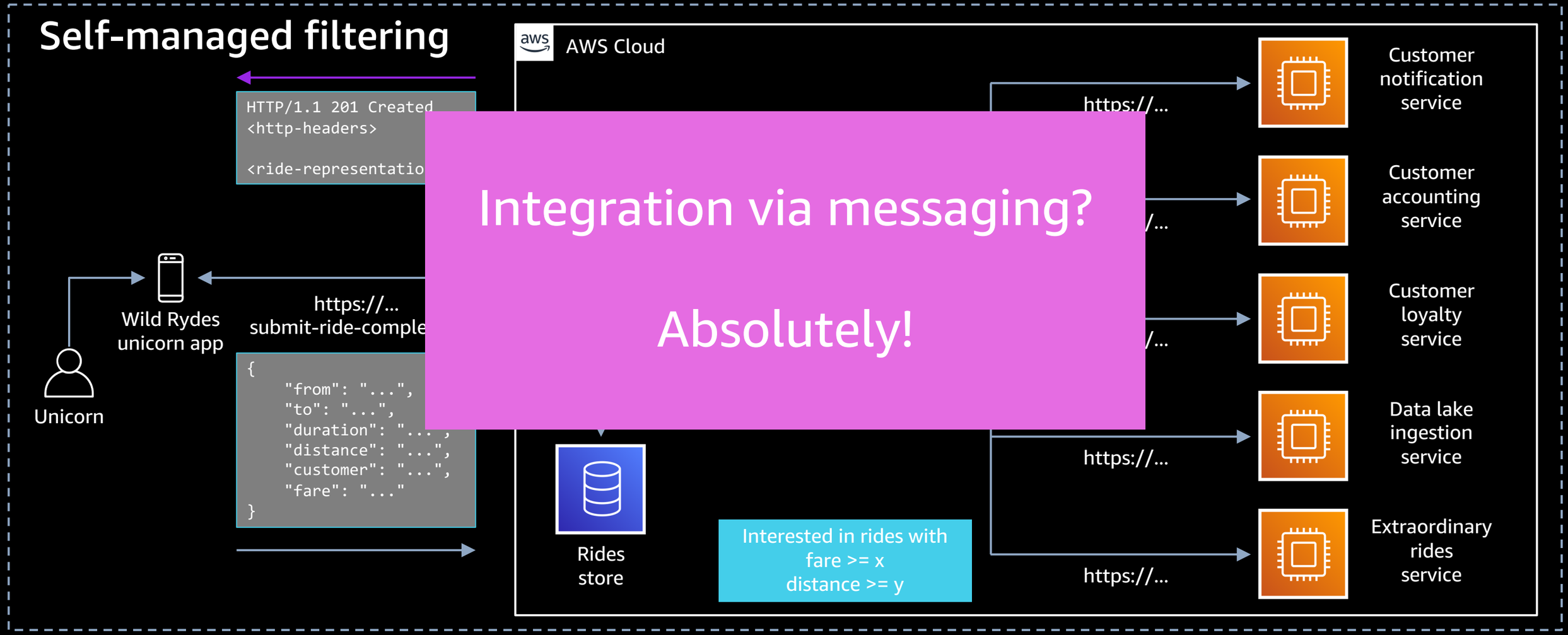
Use case: Submit a ride completion



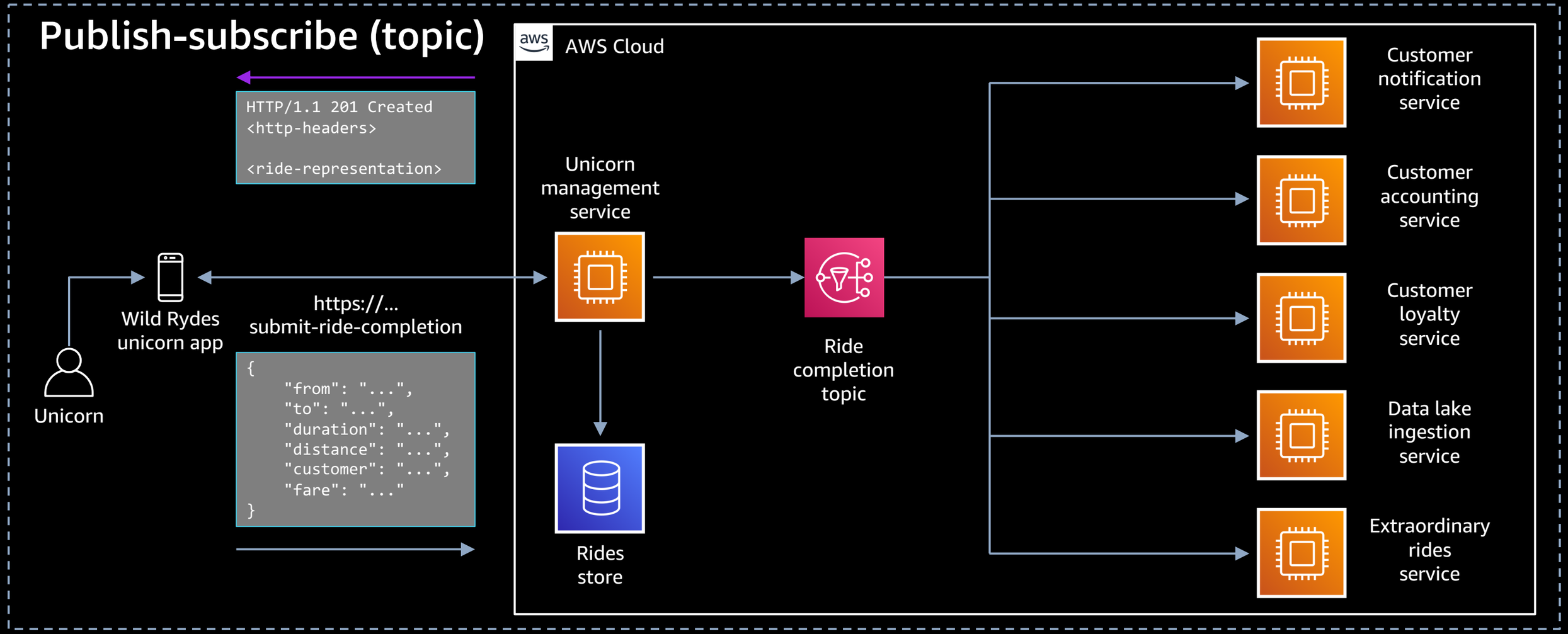
Use case: Submit a ride completion



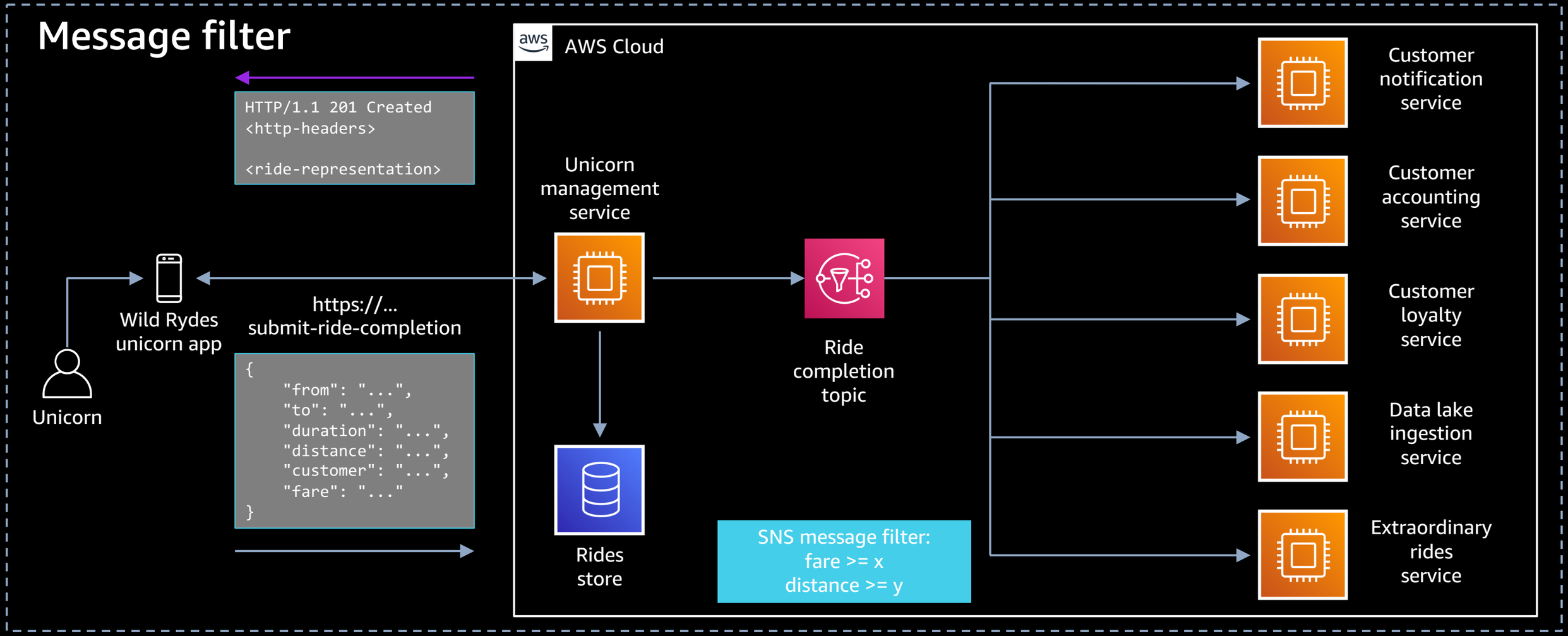
Use case: Submit a ride completion



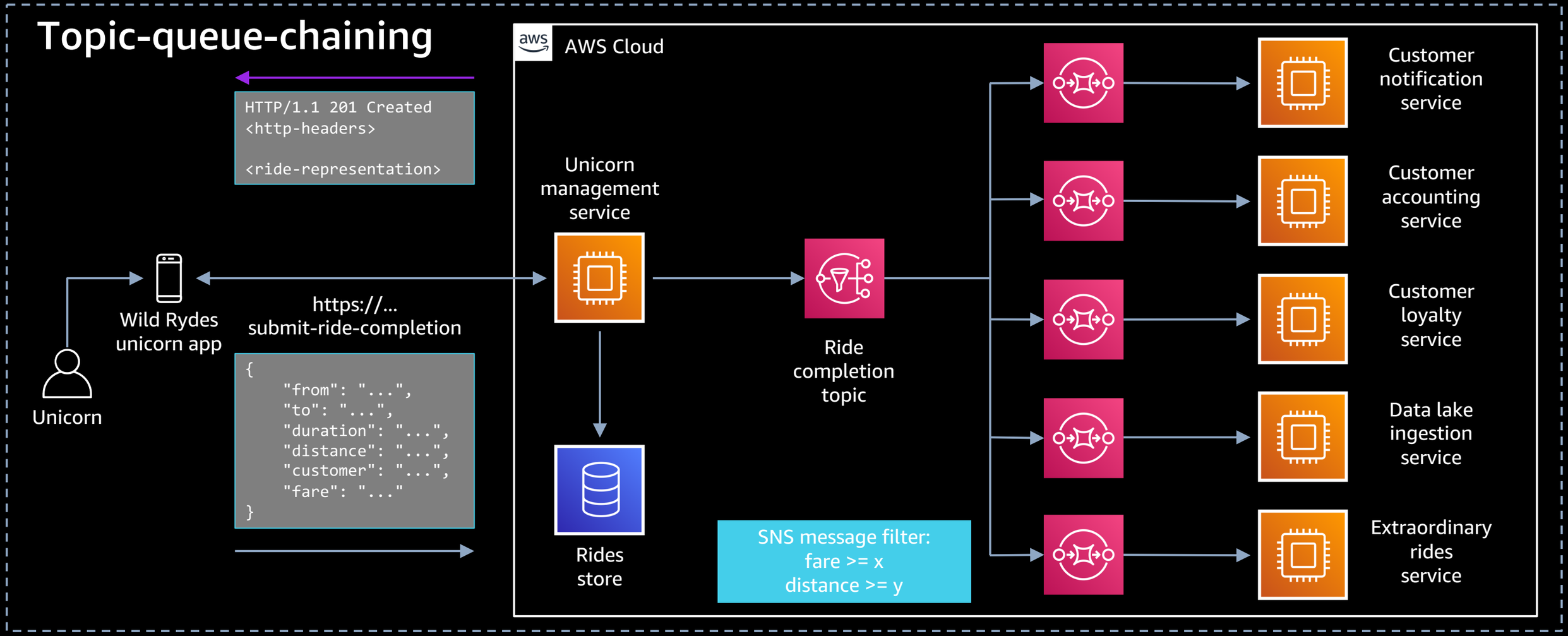
Use case: Submit a ride completion



Use case: Submit a ride completion



Use case: Submit a ride completion



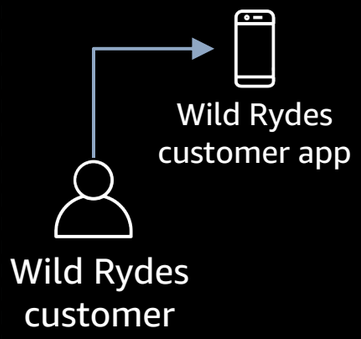
Use case: Instant ride RFQ



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

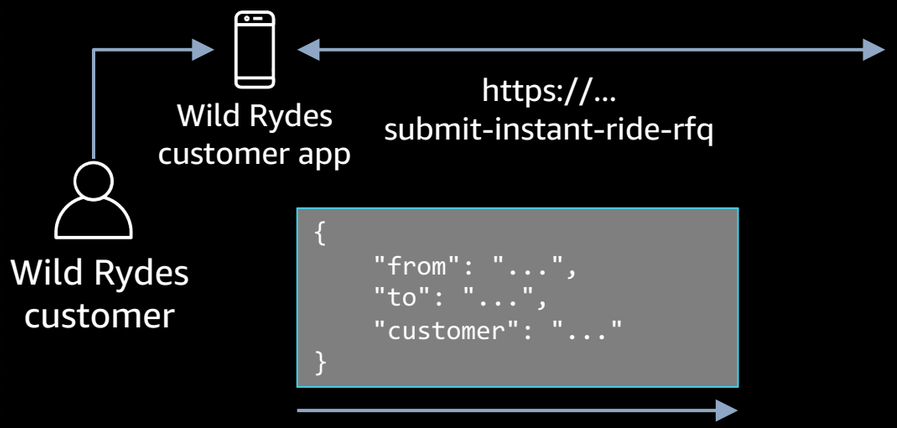
Use case: Instant ride RFQ

Scatter-gather

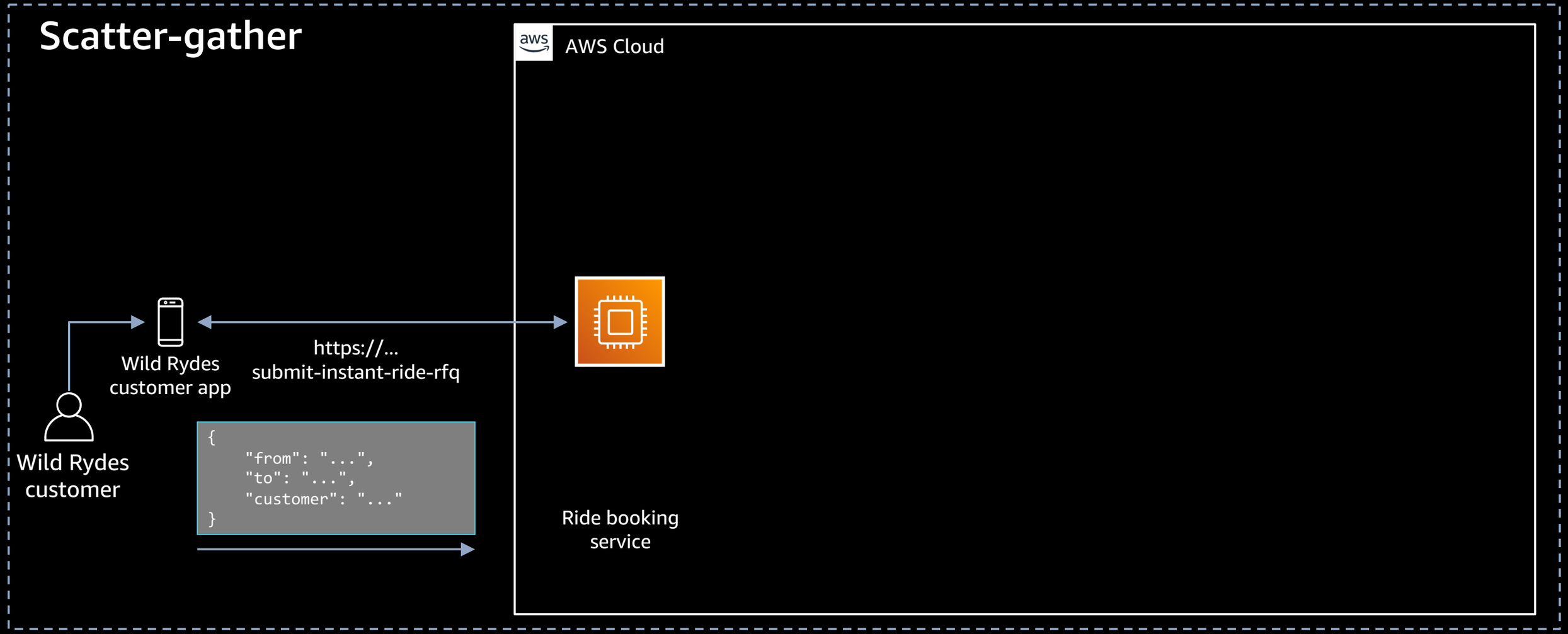


Use case: Instant ride RFQ

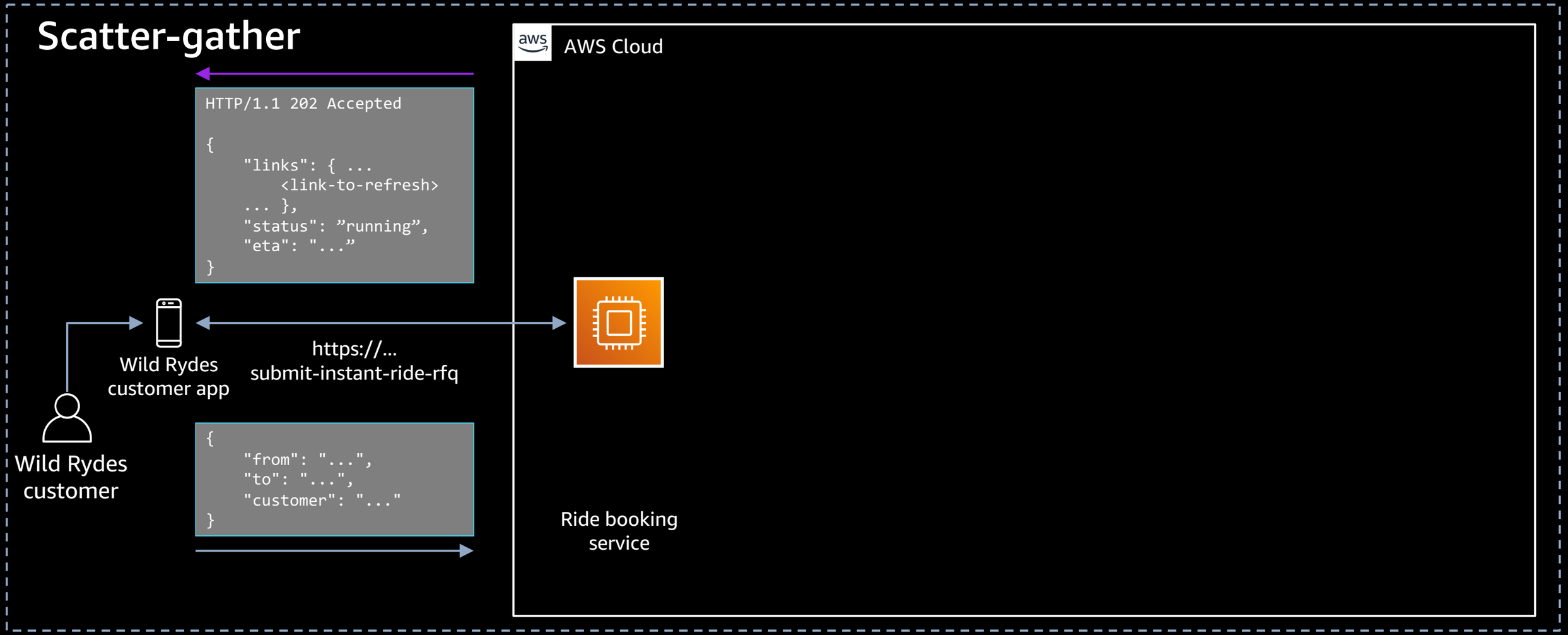
Scatter-gather



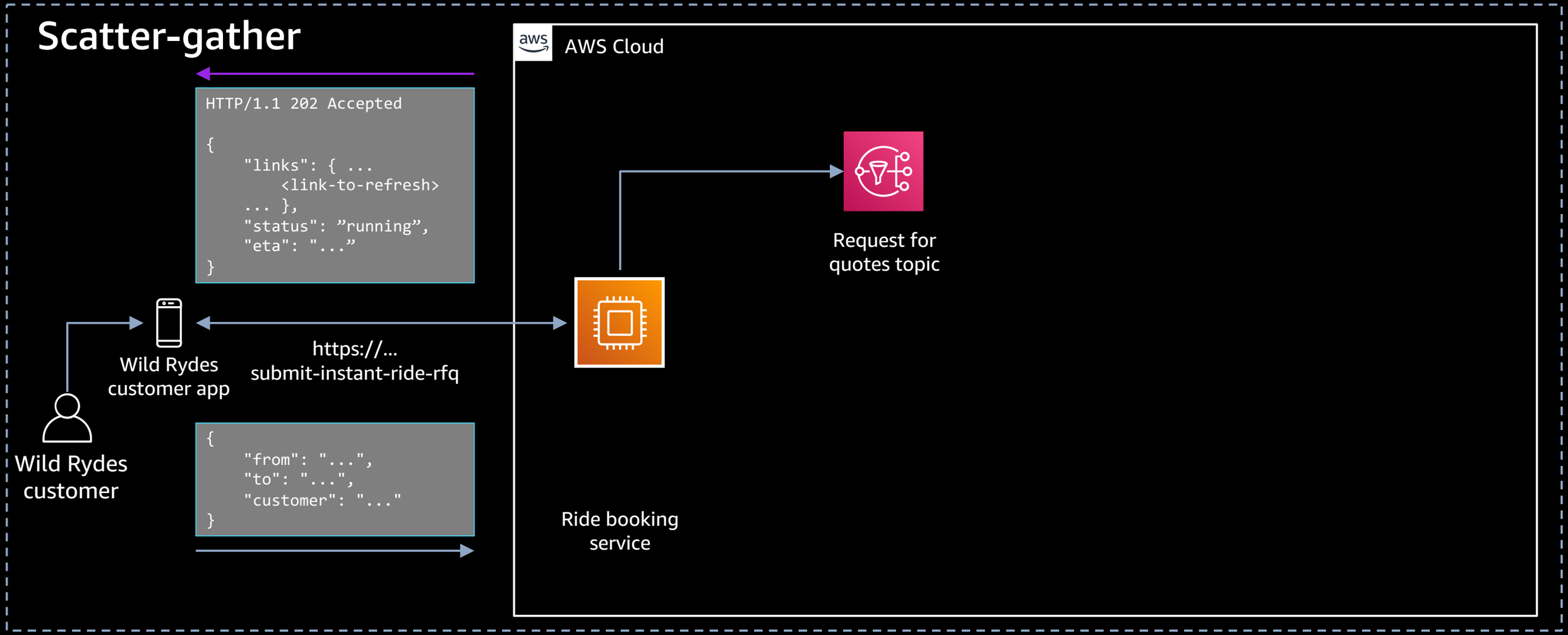
Use case: Instant ride RFQ



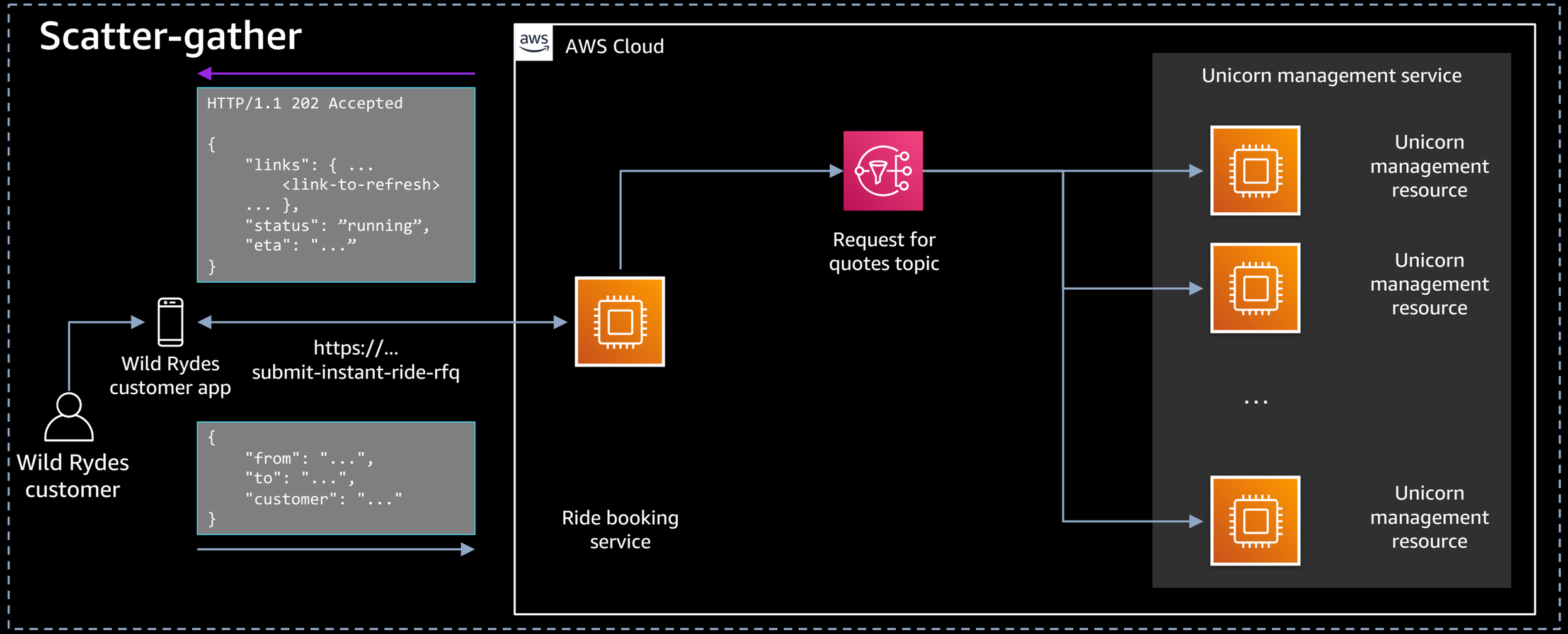
Use case: Instant ride RFQ



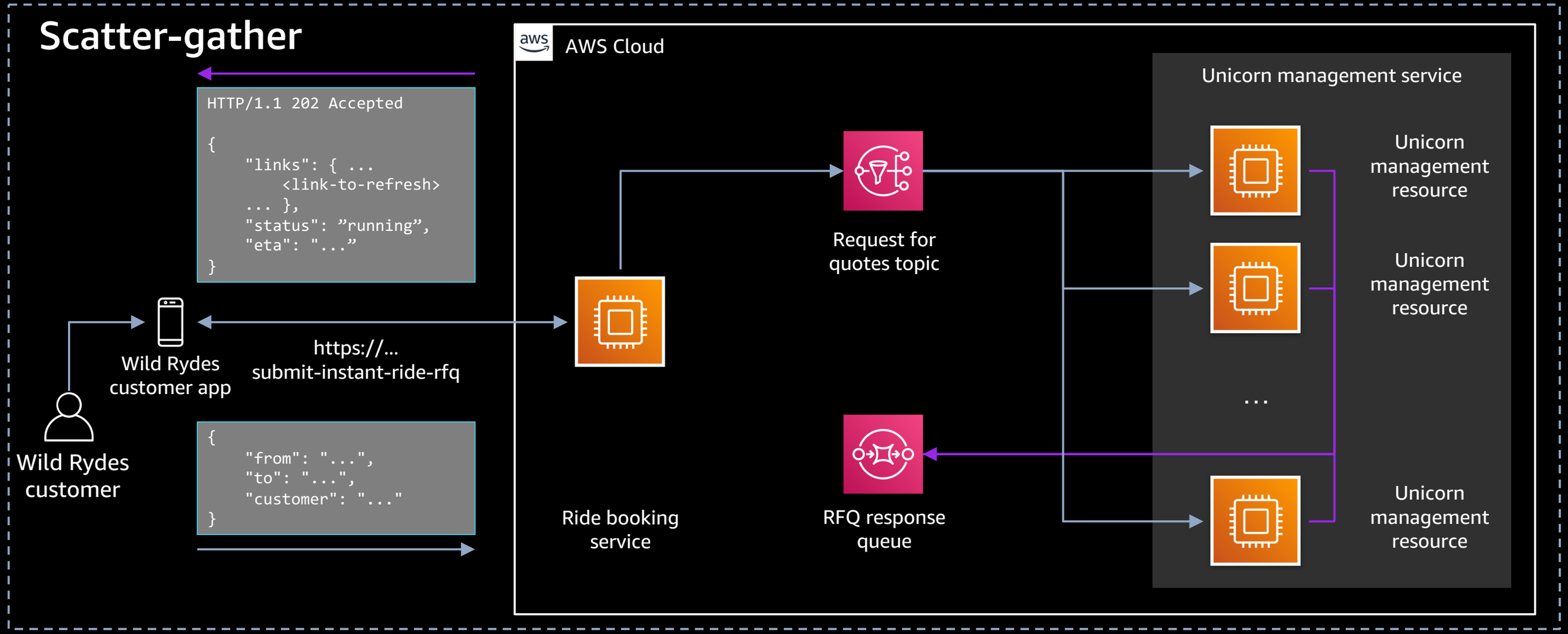
Use case: Instant ride RFQ



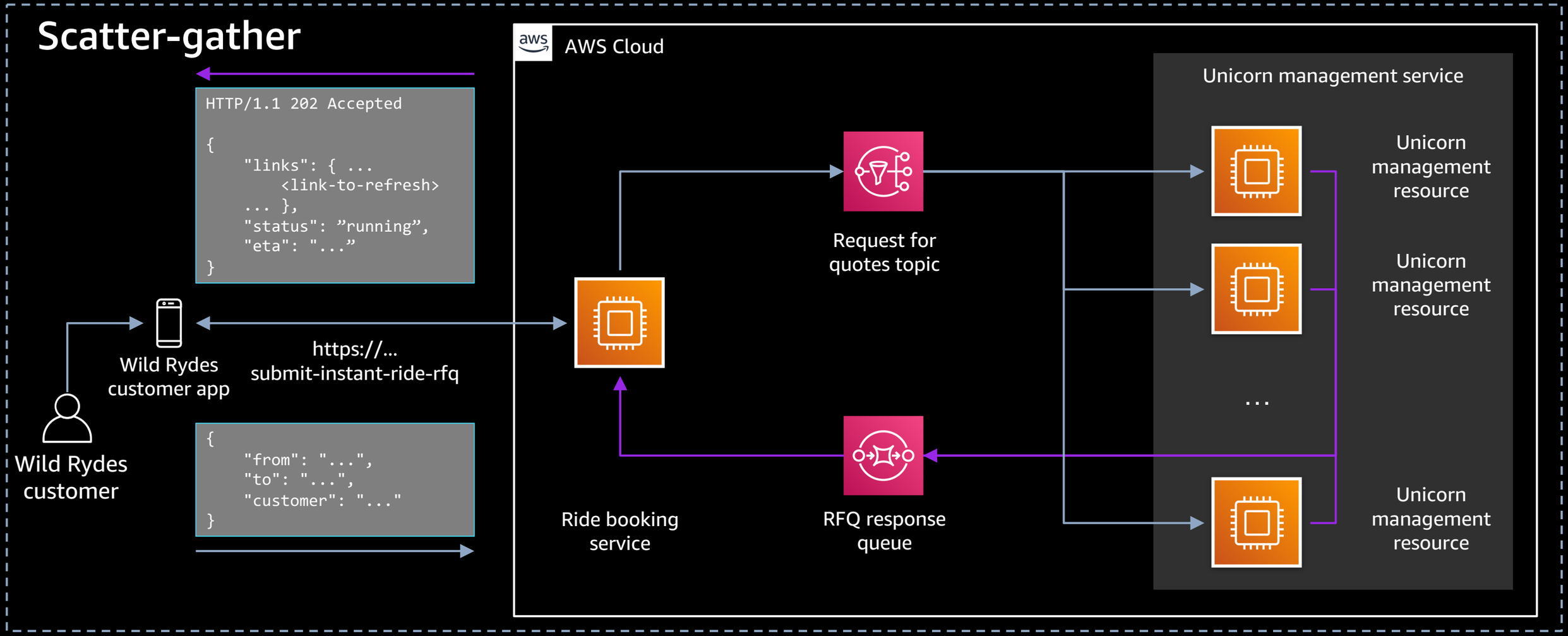
Use case: Instant ride RFQ



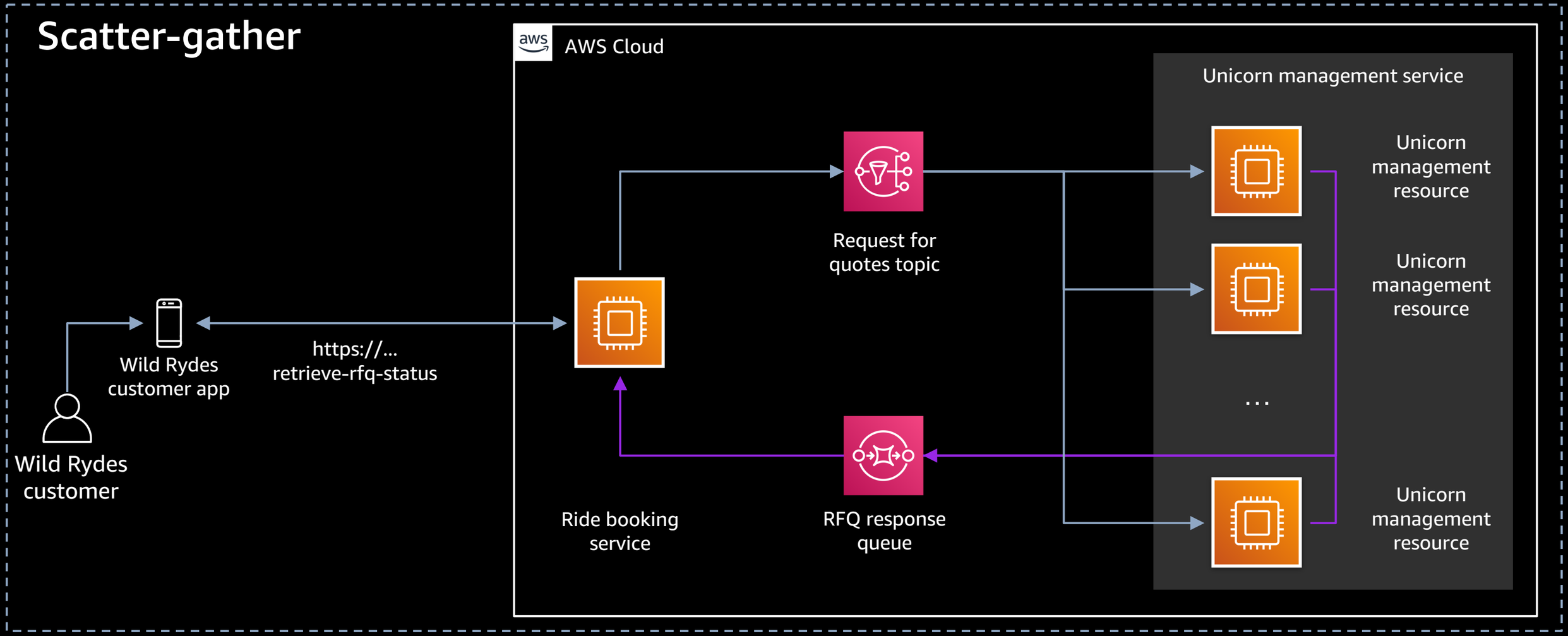
Use case: Instant ride RFQ



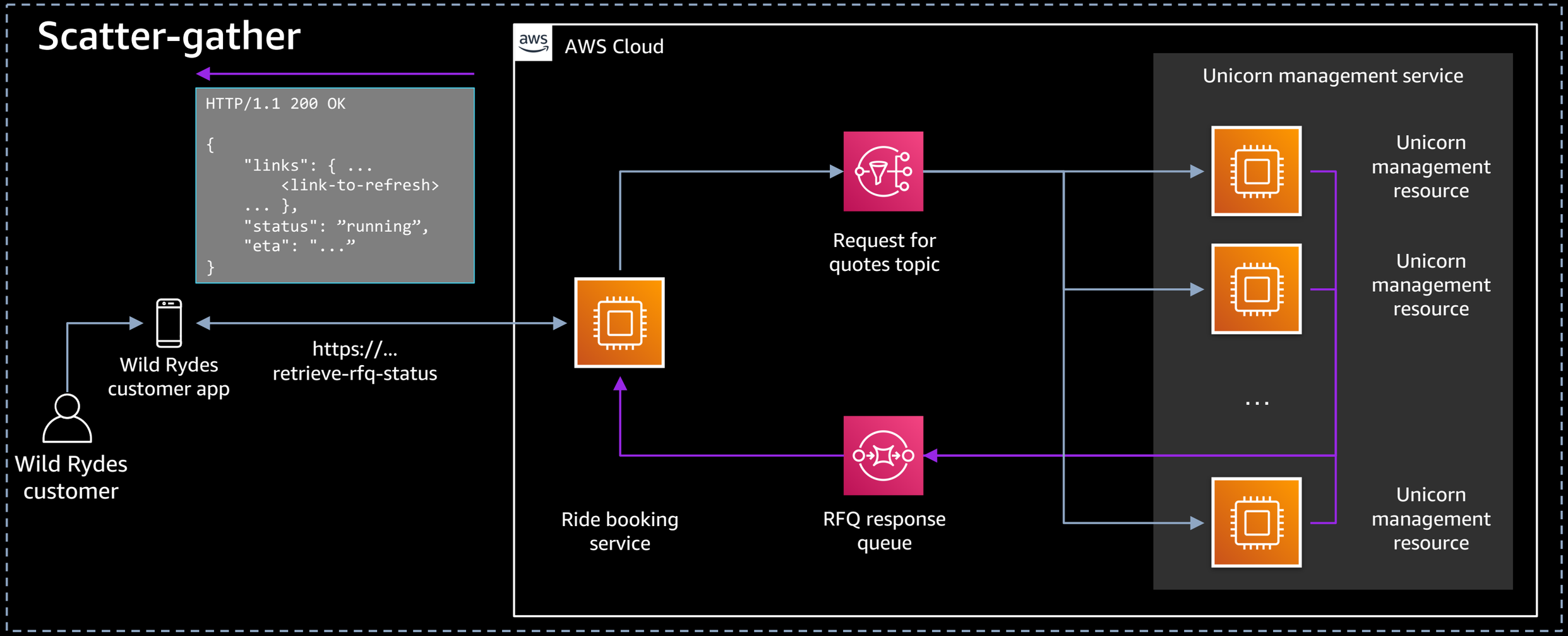
Use case: Instant ride RFQ



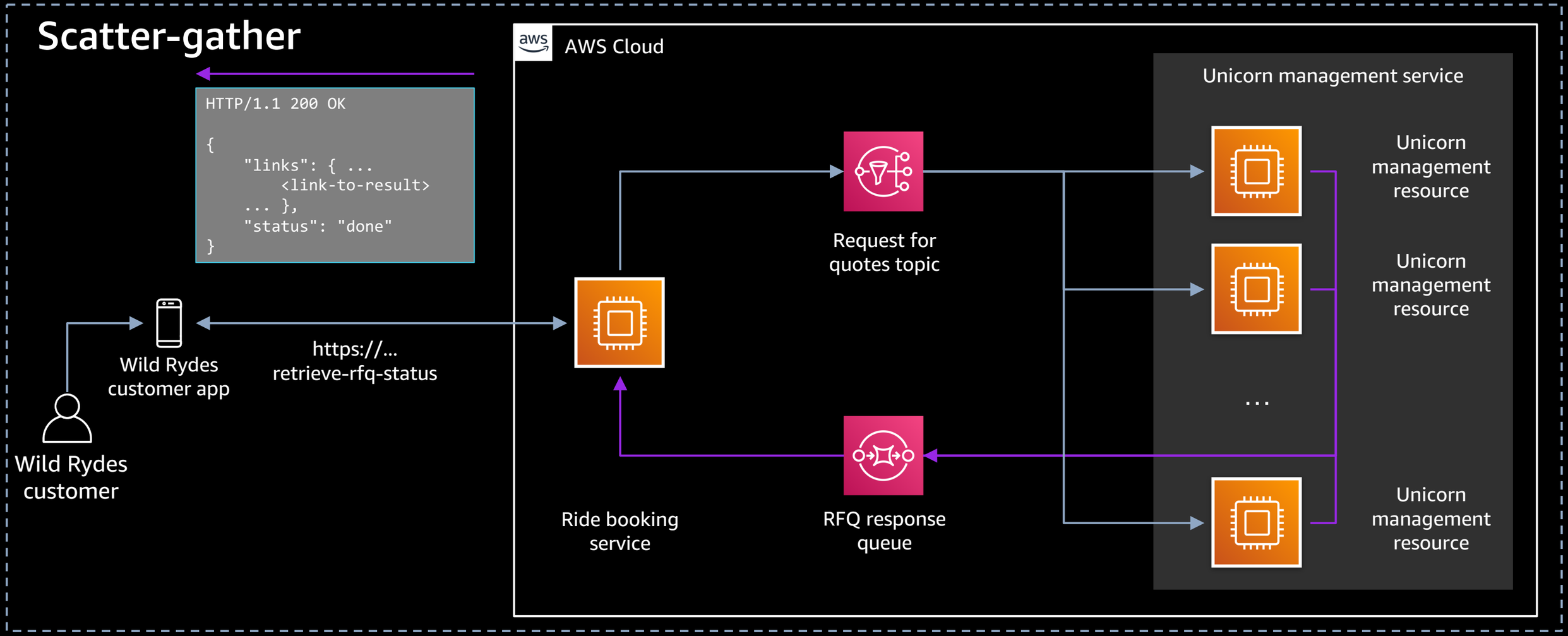
Use case: Instant ride RFQ



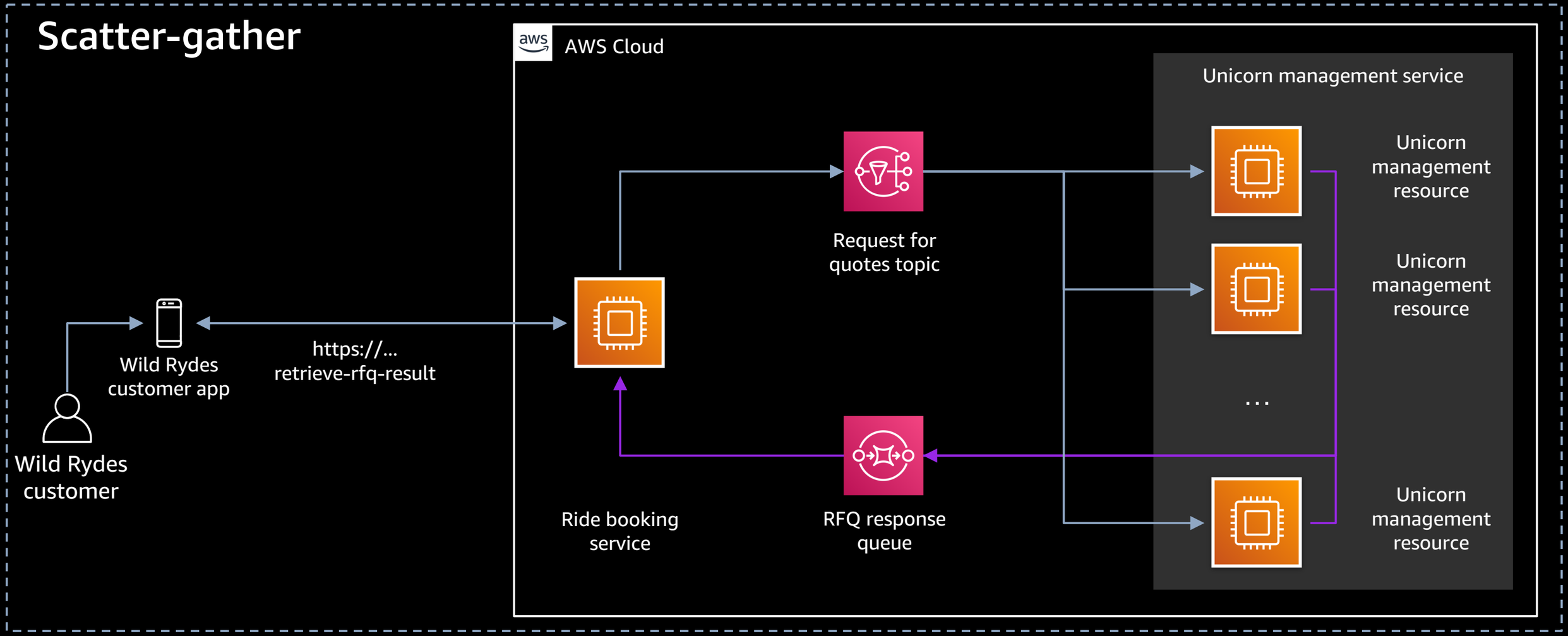
Use case: Instant ride RFQ



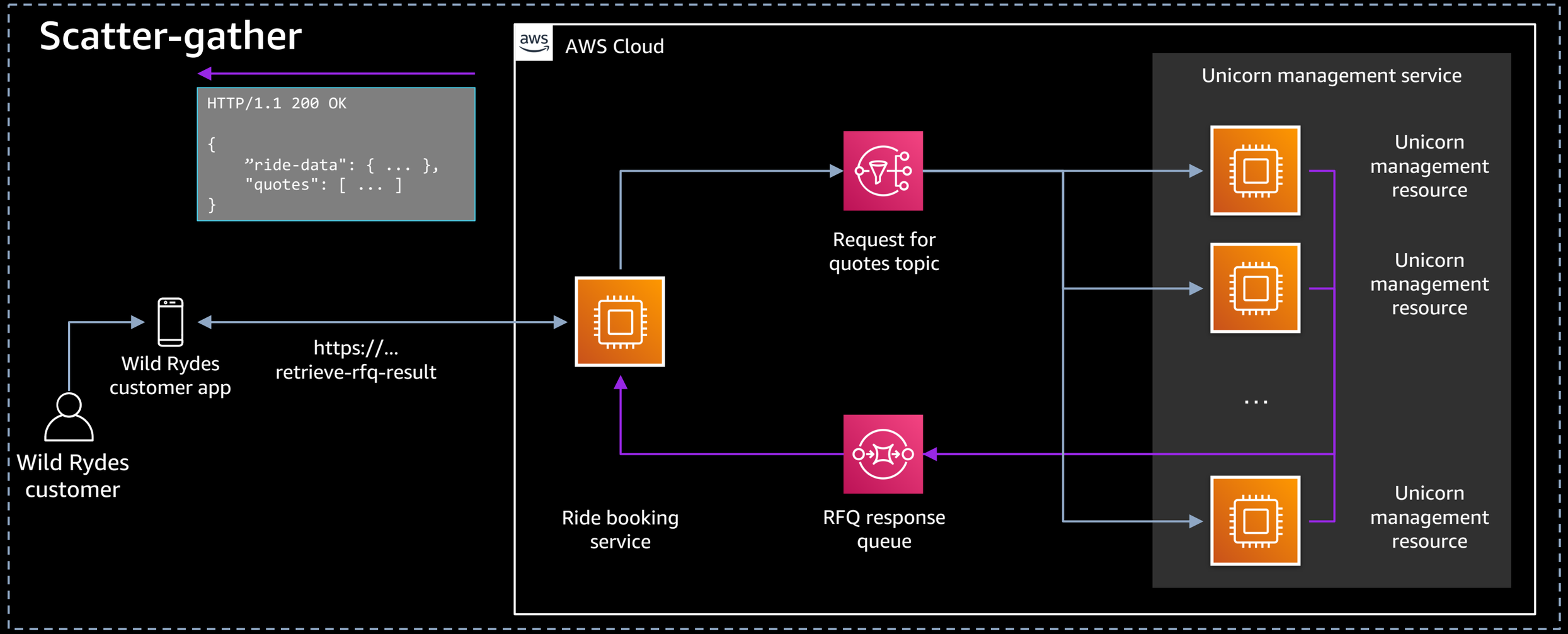
Use case: Instant ride RFQ



Use case: Instant ride RFQ



Use case: Instant ride RFQ



Resources and call to action



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Resources and call to action

Resources and call to action

AWS blogs about messaging

aws.amazon.com/blogs/compute/tag/messaging

Resources and call to action

AWS blogs about messaging

aws.amazon.com/blogs/compute/tag/messaging

Hands-on workshop on implementing the patterns from this talk

async-messaging.workshop.aws

Ask your AWS SA for an application integration immersion day

Resources and call to action

AWS blogs about messaging

aws.amazon.com/blogs/compute/tag/messaging

Hands-on workshop on implementing the patterns from this talk

async-messaging.workshop.aws

Ask your AWS SA for an application integration immersion day

Keep in mind

Loose coupling is always better than lousy coupling!

Thank you!

Dirk Fröhner

Principal Solutions Architect
AWS

@dirk_f5r

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.