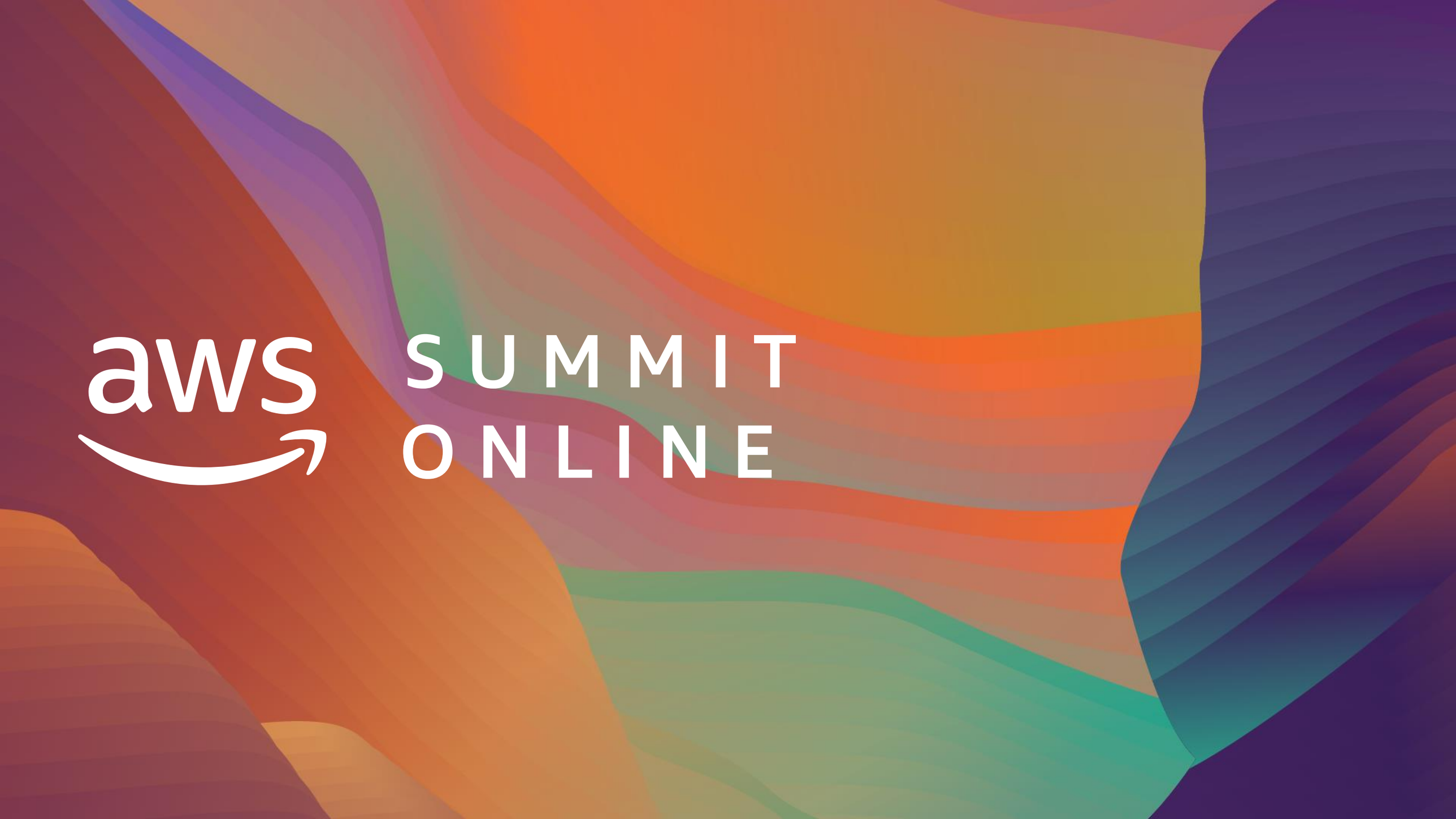

aws
SUMMIT
ONLINE

Application integration patterns for microservices

Anshul Sharma

Solutions Architect

Amazon Web Services

aws SUMMIT ONLINE

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

Agenda

Introduction

Application integration patterns

Relevant use cases

Introduction

© 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

Distinguished Engineer

AWS Messaging, Workflow Management

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

Potential drawbacks of synchronous systems

Synchronous systems are tightly coupled

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

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

Photo: Dirk Fröhner

Application integration patterns

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

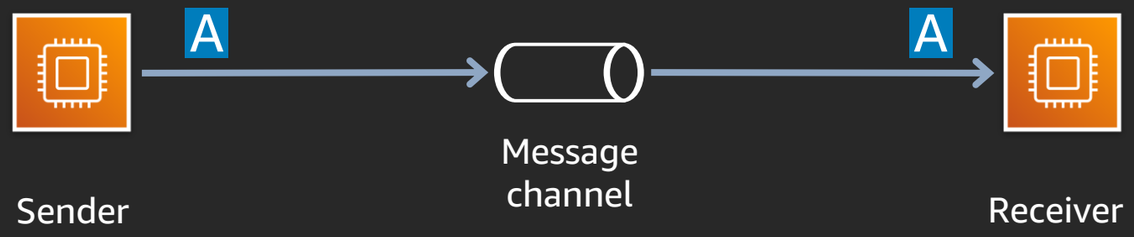
Message exchange

One-way

Request-response

Message exchange: One-way and request-response

One-way



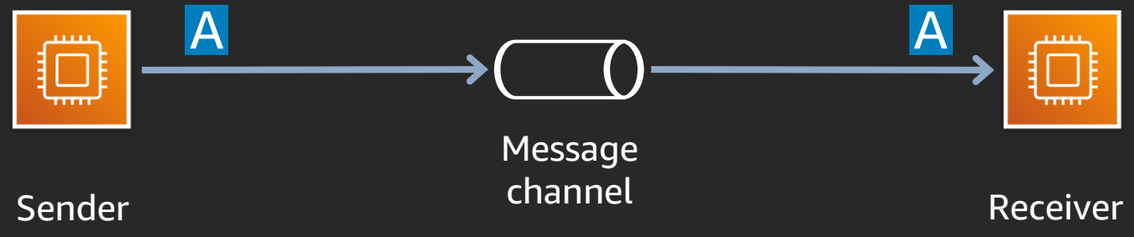
No response expected

Synchronous vs. fire-and-forget

Request-response

Message exchange: One-way and request-response

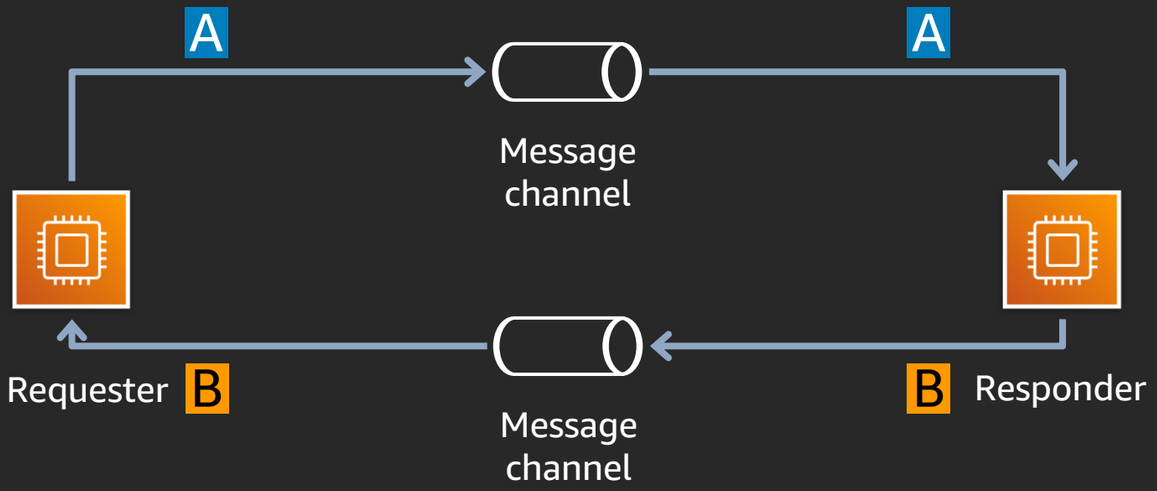
One-way



No response expected

Synchronous vs. fire-and-forget

Request-response



Response expected

Return address

Correlation ID

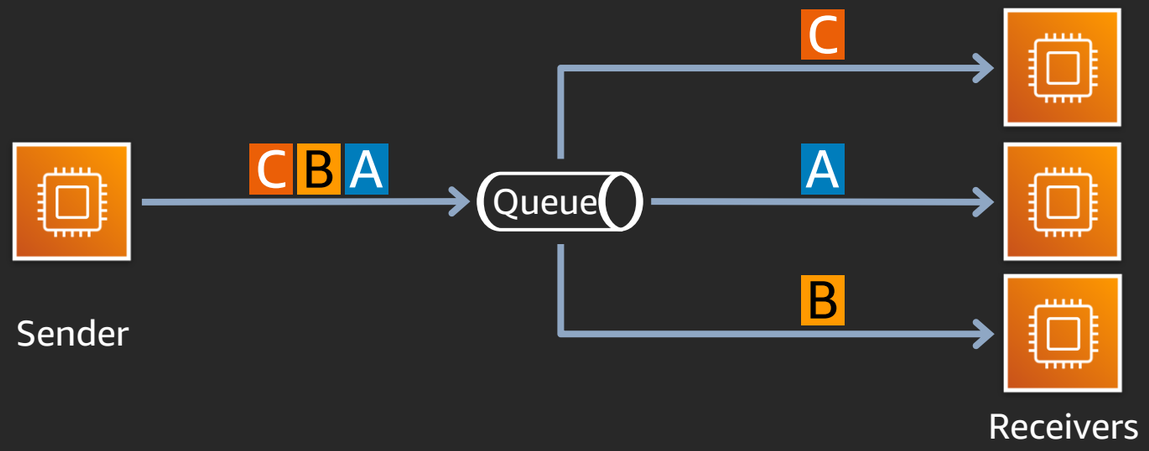
Message channels

Point-to-point (queue)

Publish-subscribe (topic)

Message channels: Queue and topic

Point-to-point (queue)



Consumed by only one receiver

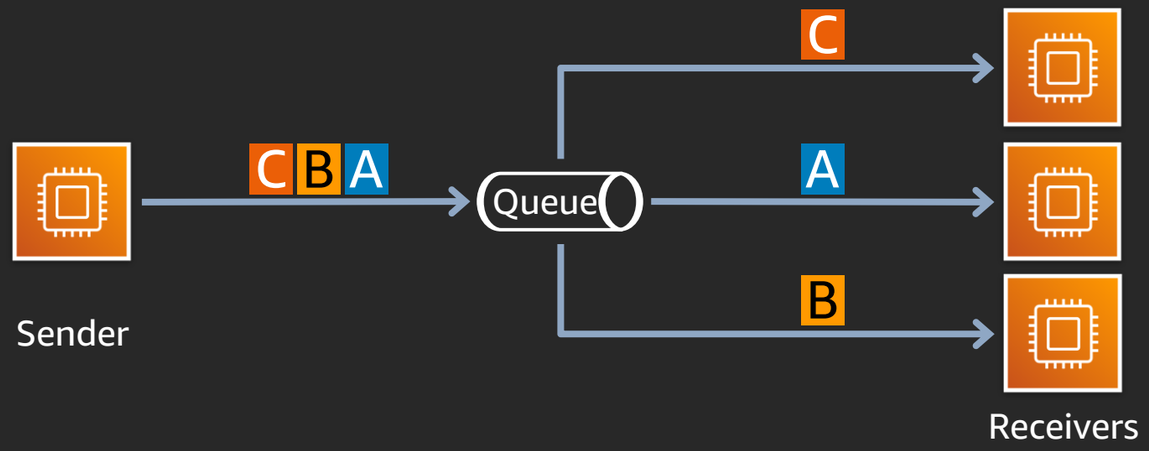
Easy to scale

Flatten peak loads

Publish-subscribe (topic)

Message channels: Queue and topic

Point-to-point (queue)

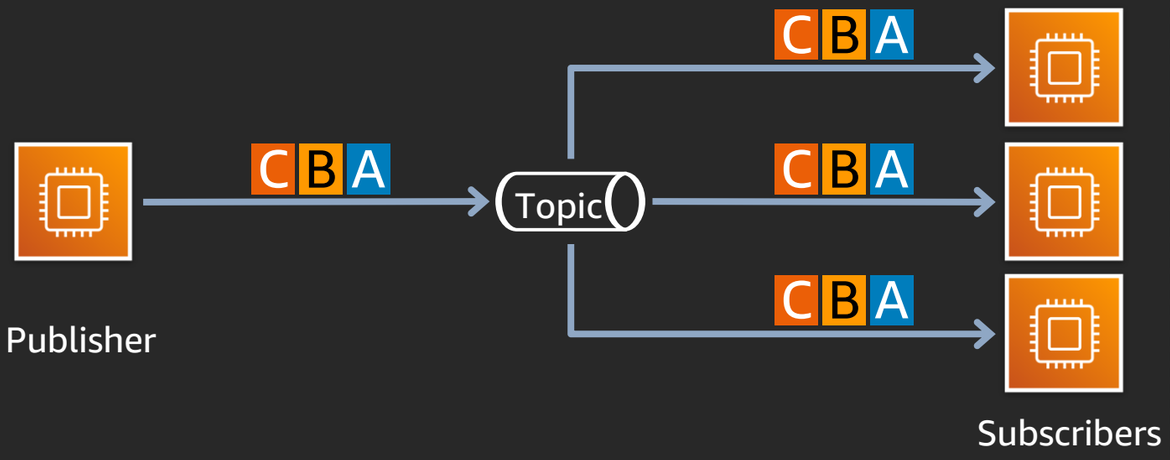


Consumed by only one receiver

Easy to scale

Flatten peak loads

Publish-subscribe (topic)

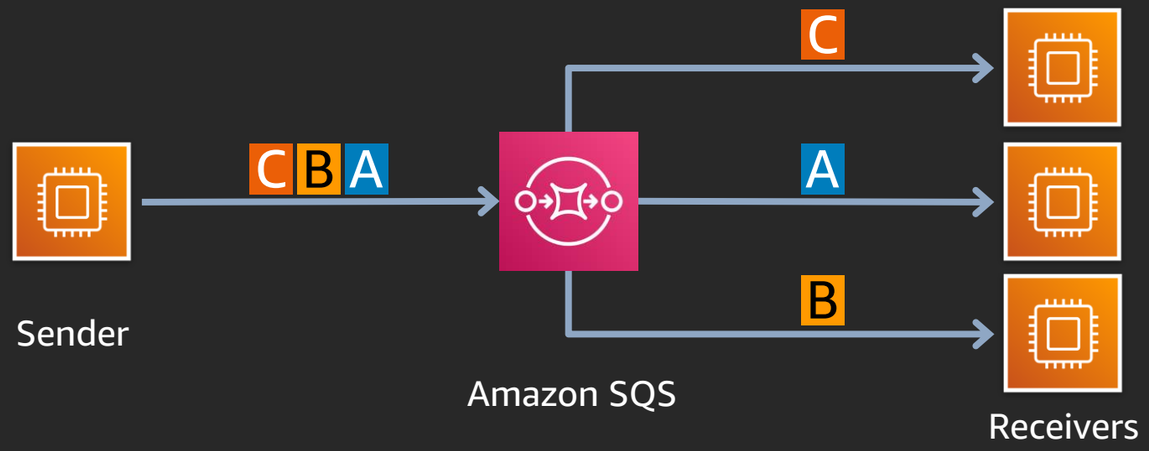


Consumed by all subscribers

Durable subscriber

Message channels: Amazon SNS and Amazon SQS

Point-to-point (queue)

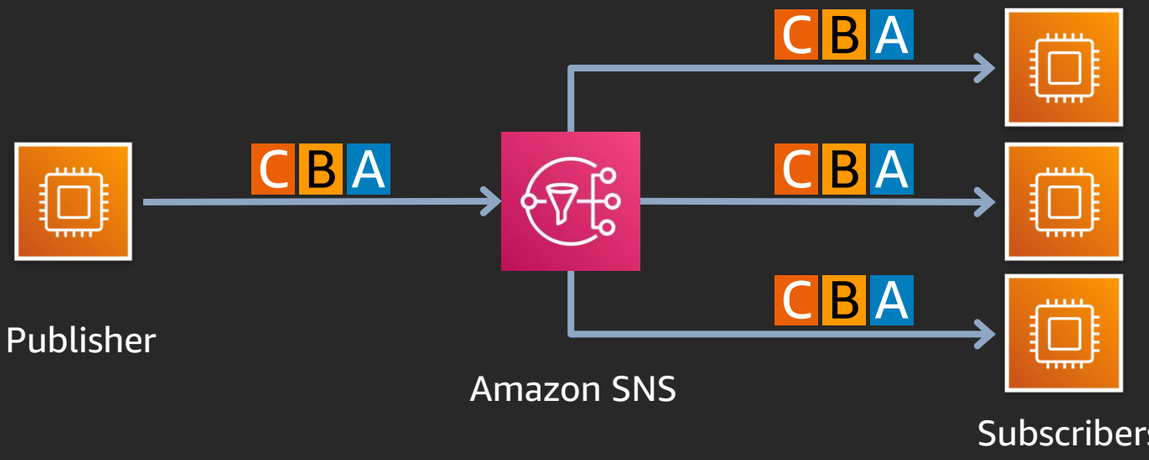


AWS service for queue functionality

Amazon Simple Queue Service (SQS)

Serverless and cloud-native

Publish-subscribe (topic)



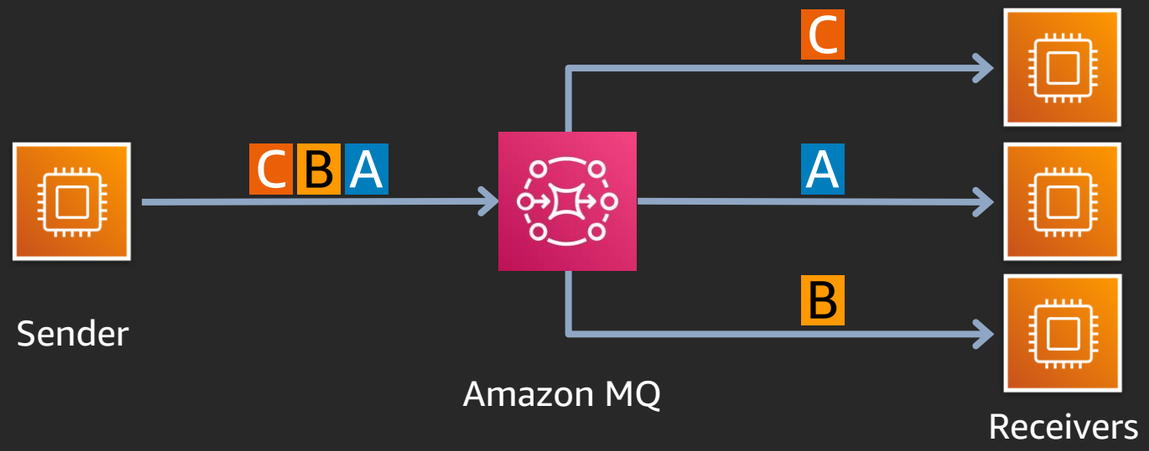
AWS service for topic functionality

Amazon Simple Notification Service (Amazon SNS)

Serverless and cloud-native

Message channels: Amazon MQ

Point-to-point (queue)

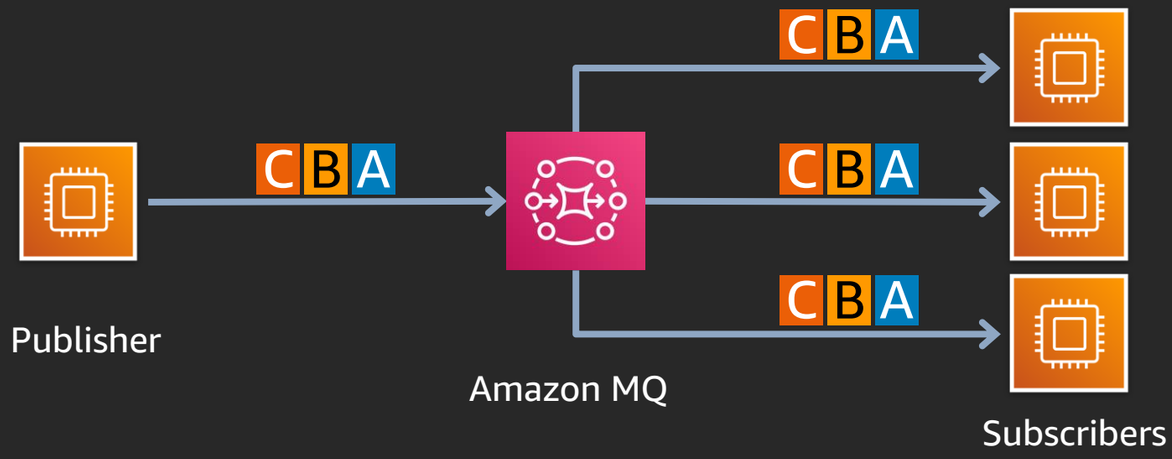


AWS service for queue functionality (non-serverless)

Amazon MQ (managed Apache ActiveMQ)

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

Publish-subscribe (topic)



AWS service for topic functionality (non-serverless)

Amazon MQ (managed Apache ActiveMQ)

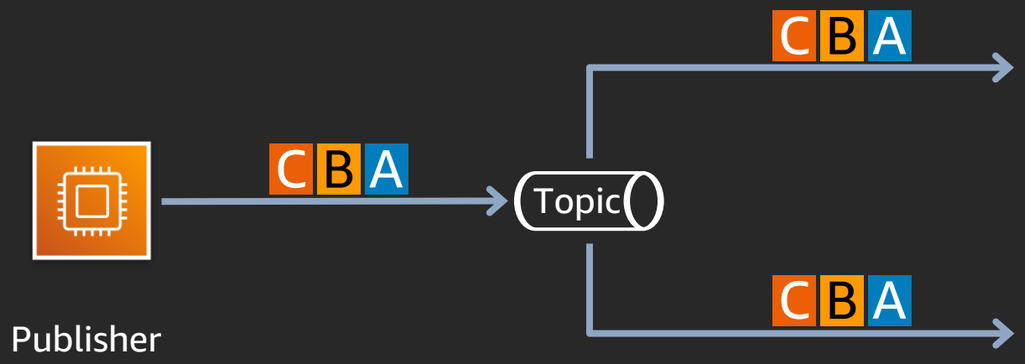
For applications constrained to protocols like JMS, AMQP, etc.

Message channels: Topic-queue-chaining

Topic-queue-chaining

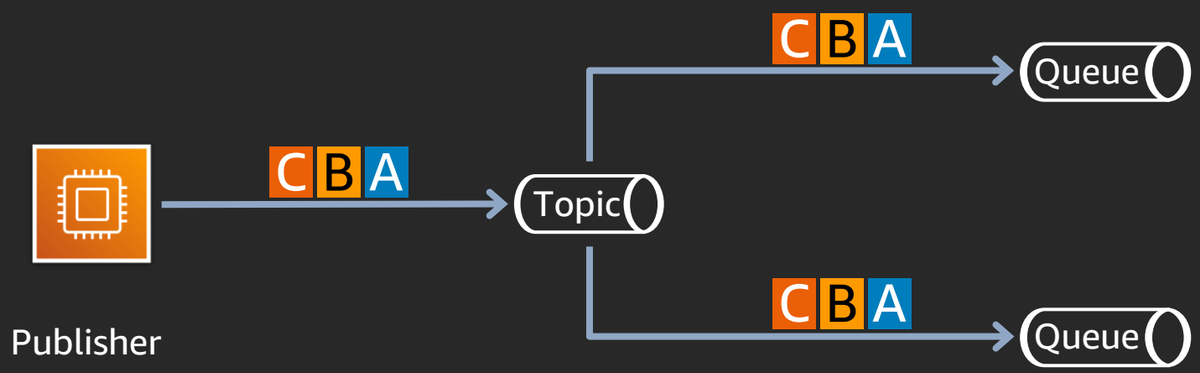
Message channels: Topic-queue-chaining

Topic-queue-chaining

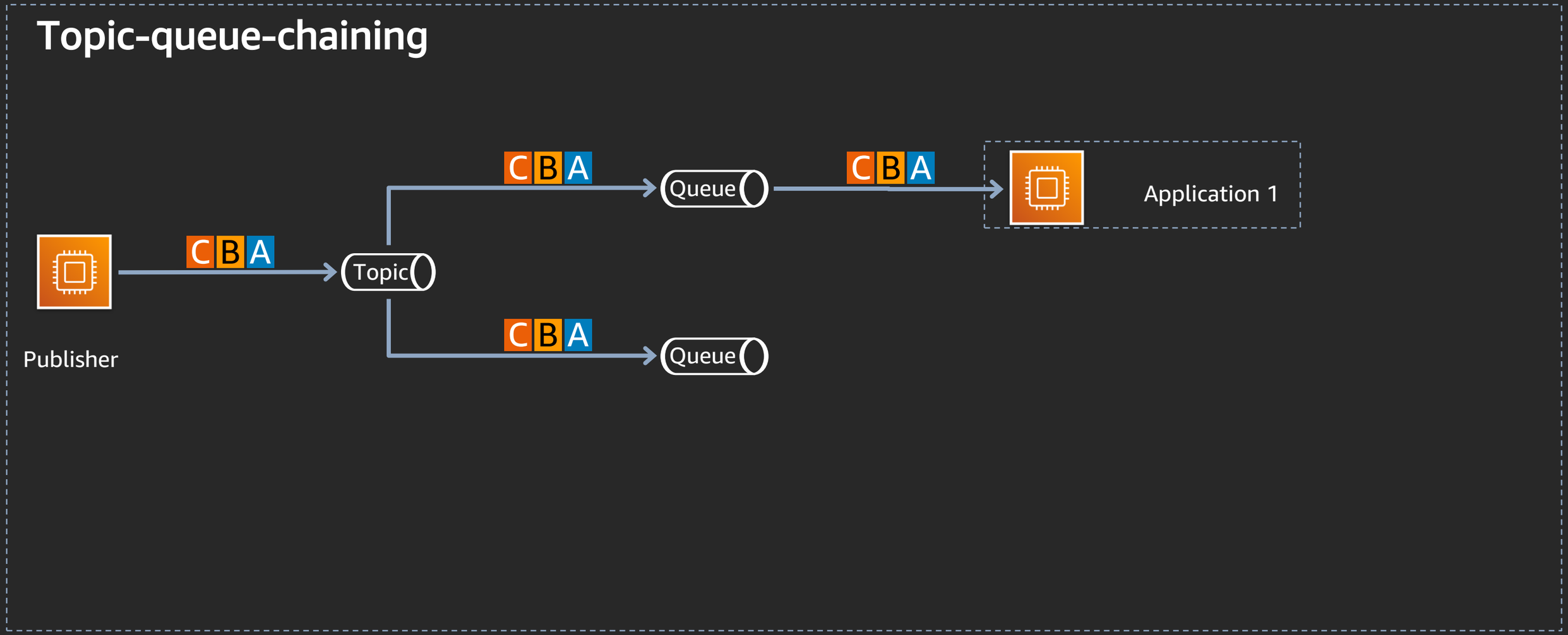


Message channels: Topic-queue-chaining

Topic-queue-chaining

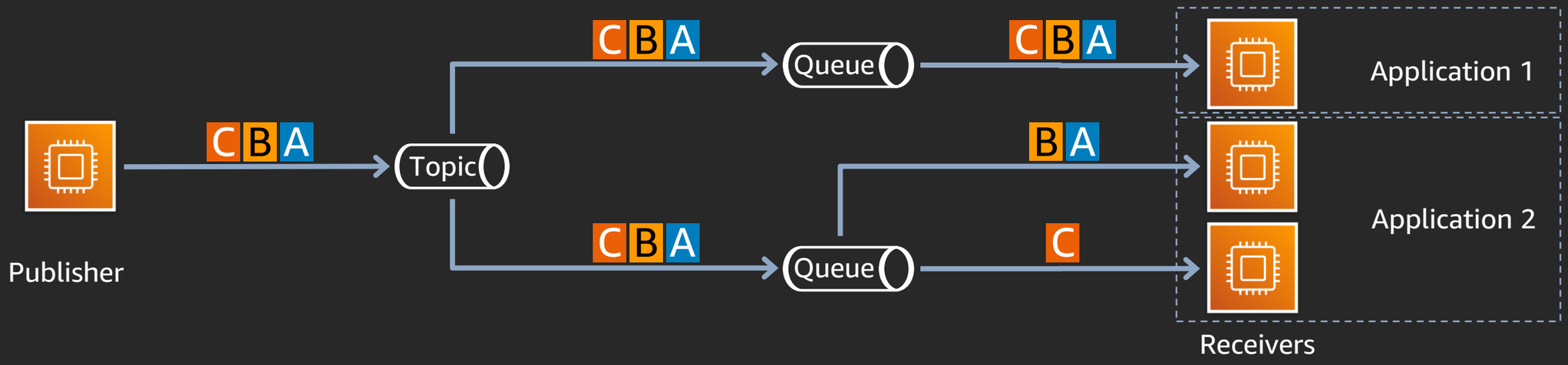


Message channels: Topic-queue-chaining



Message channels: Topic-queue-chaining

Topic-queue-chaining



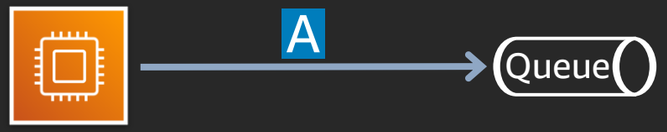
Allows fan-out and receiver scale-out at the same time

Message channels: Dead letter queue

Dead letter queue (DLQ)

Message channels: Dead letter queue

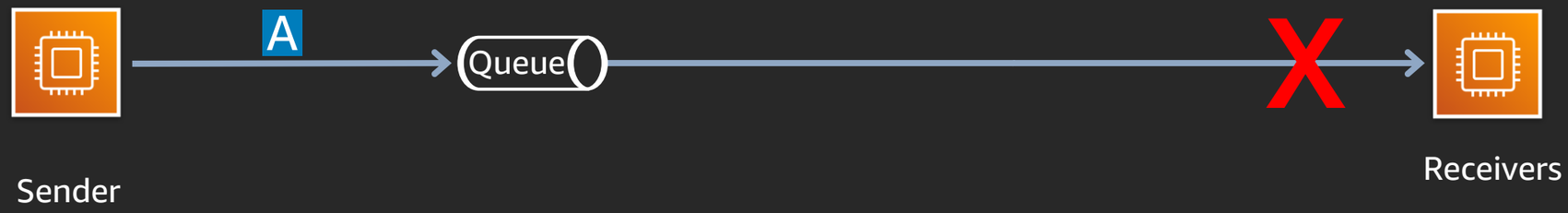
Dead letter queue (DLQ)



Sender

Message channels: Dead letter queue

Dead letter queue (DLQ)

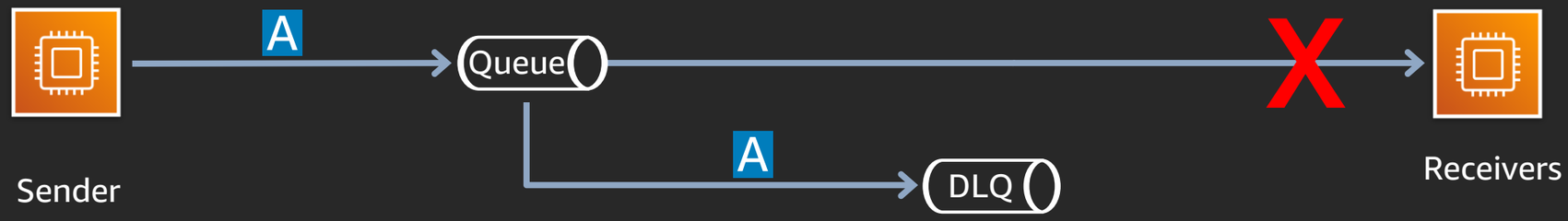


Transient failure mitigation

Poison pill handling

Message channels: Dead letter queue

Dead letter queue (DLQ)



Transient failure mitigation

Poison pill handling

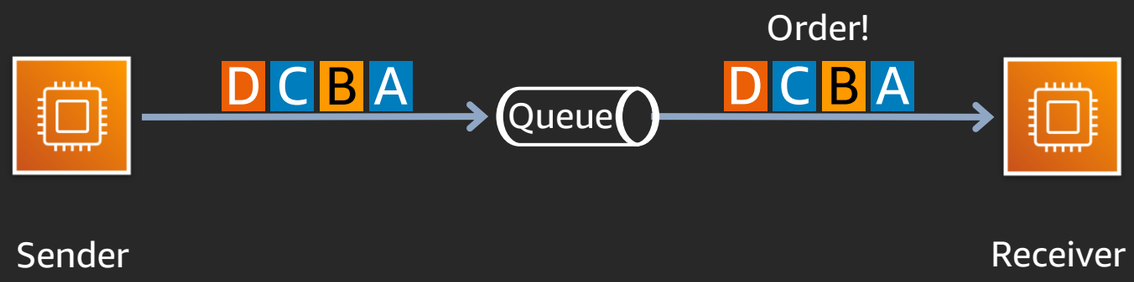
Message channels: FIFO queue

FIFO queue

Message groups

Message channels: FIFO queue

FIFO queue

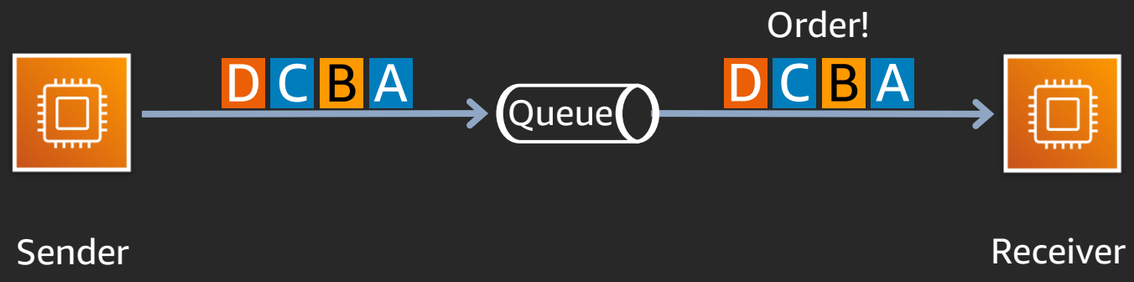


Guaranteed ordering

Message groups

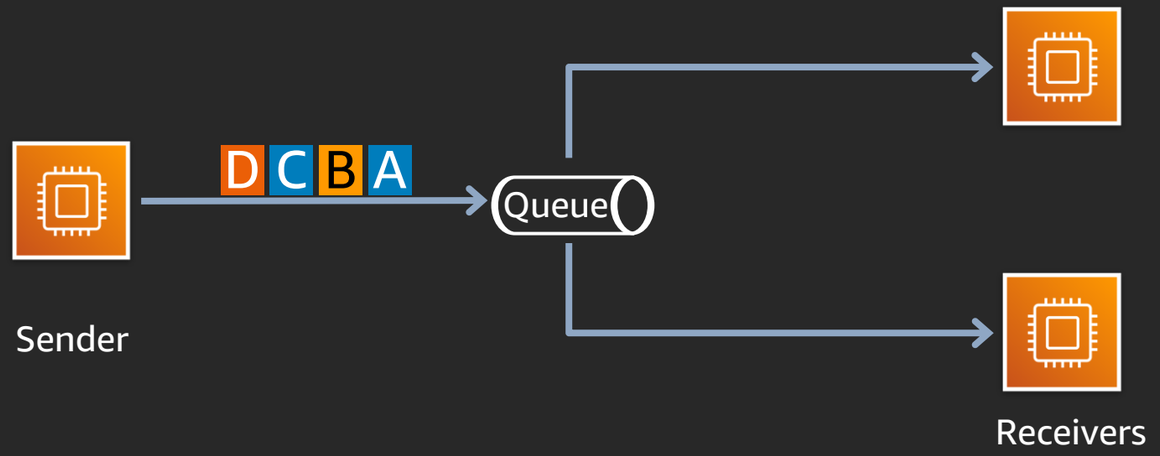
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

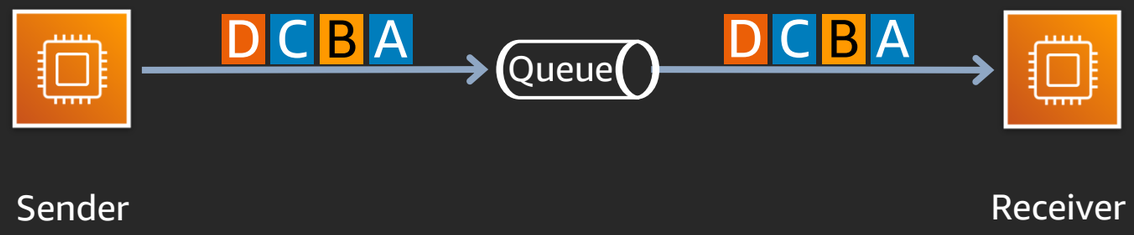


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

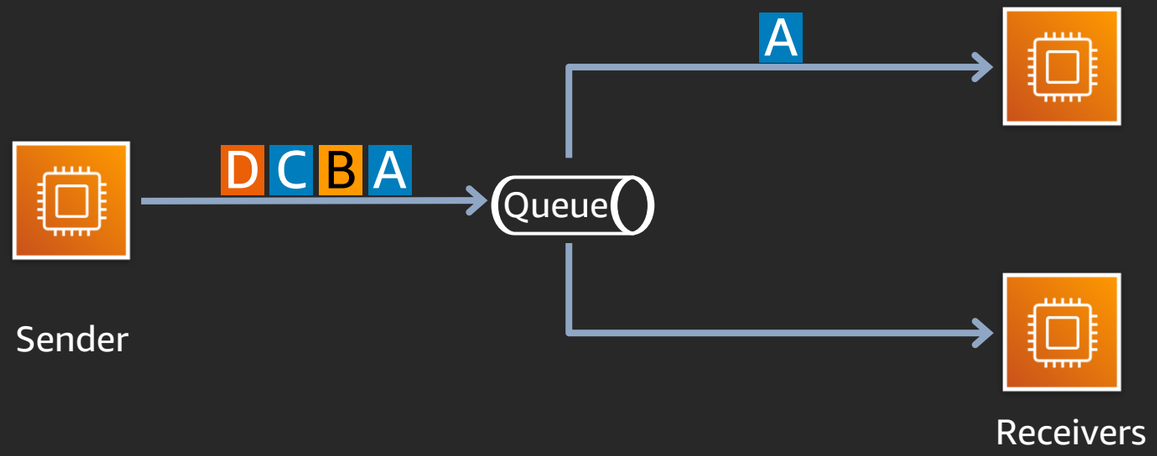
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

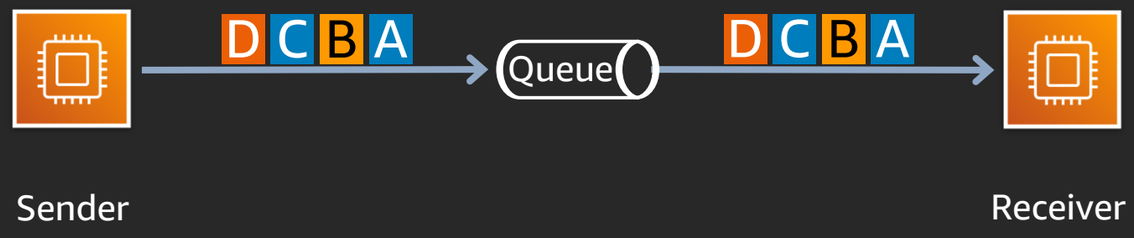


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

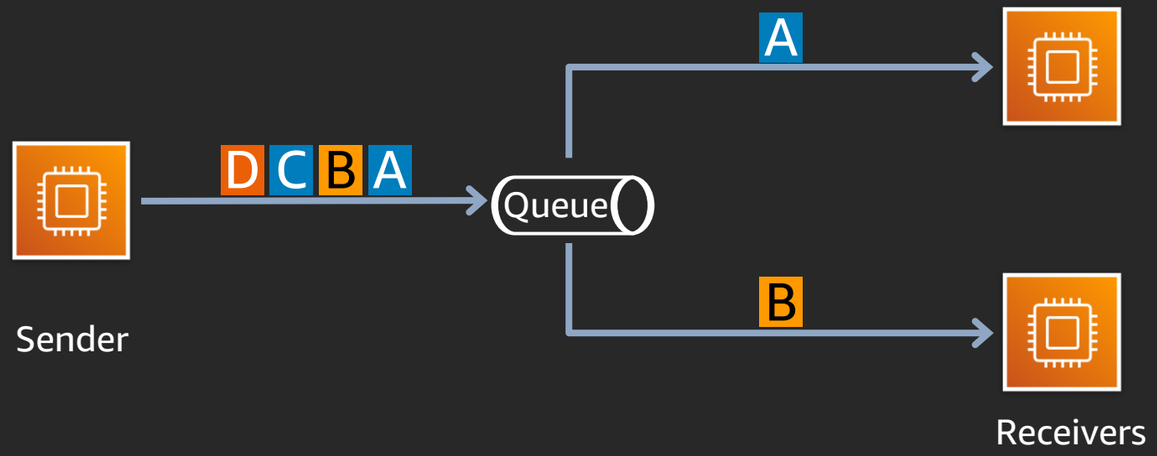
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

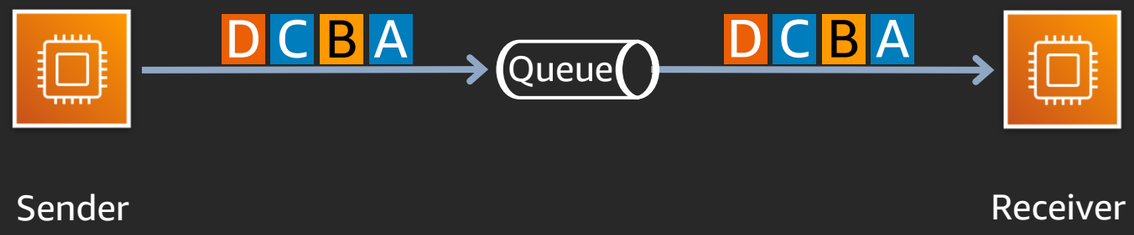


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

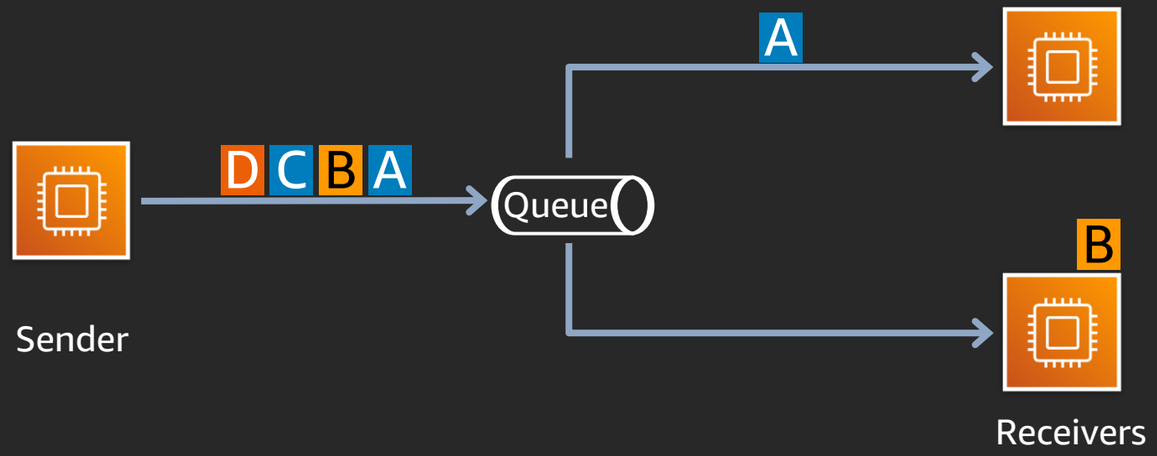
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

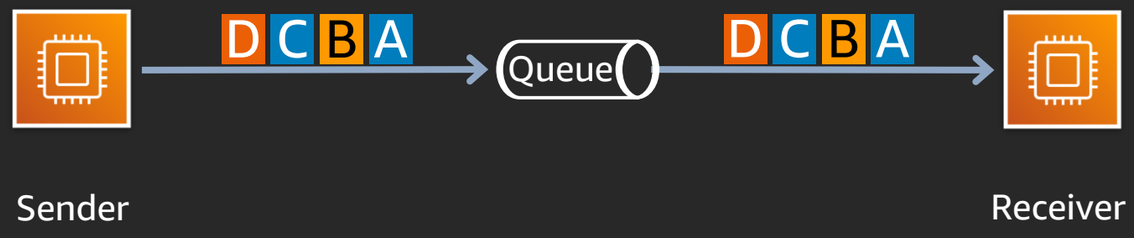


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

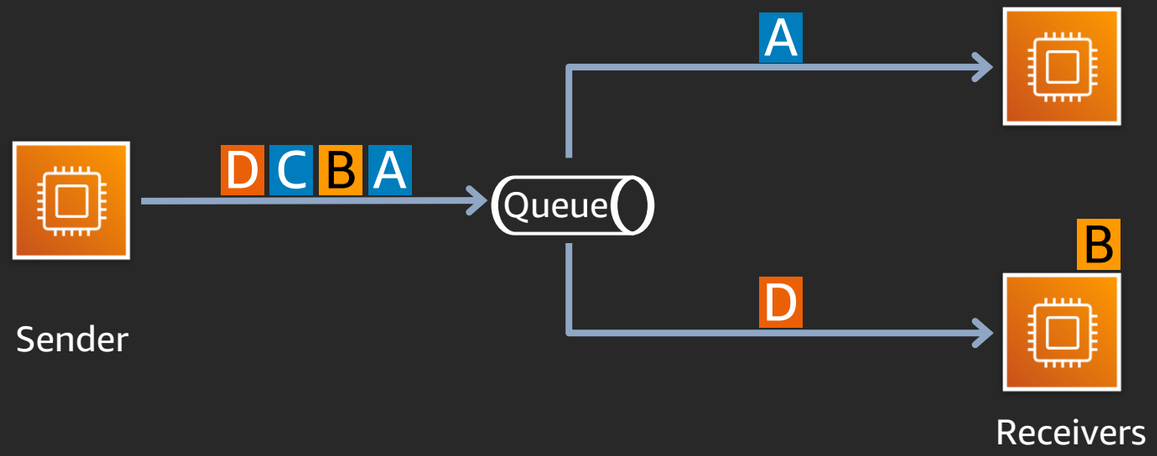
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

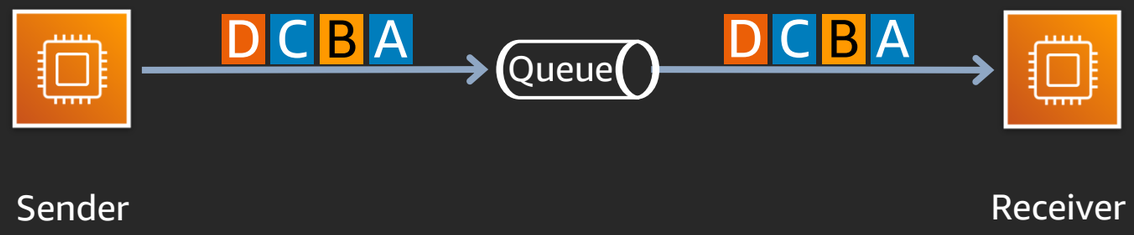


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

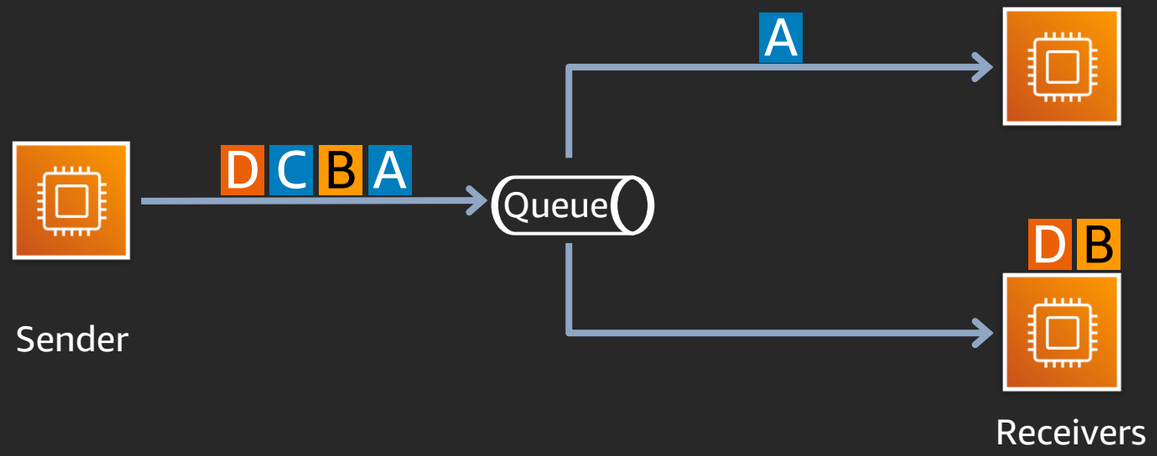
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

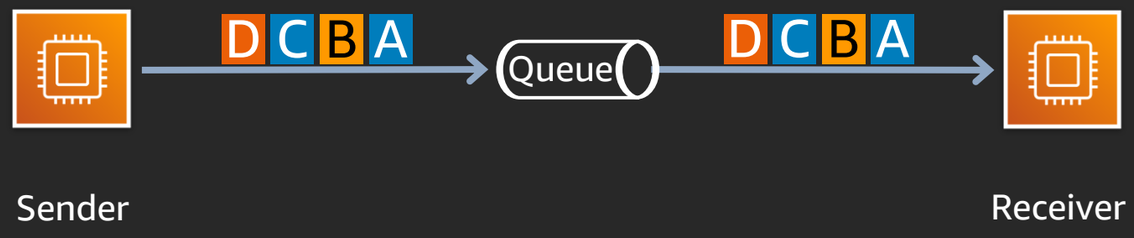


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

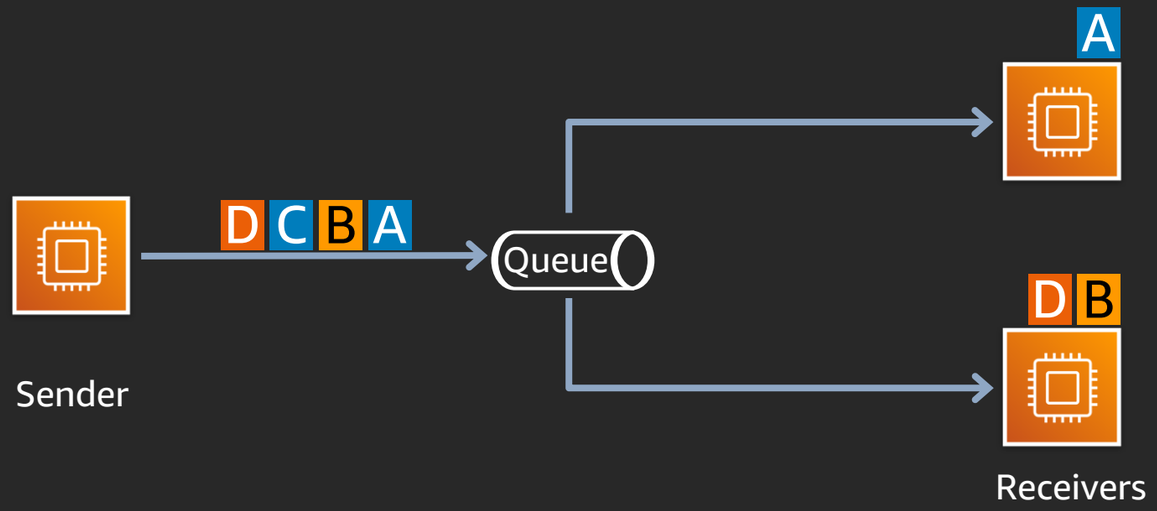
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

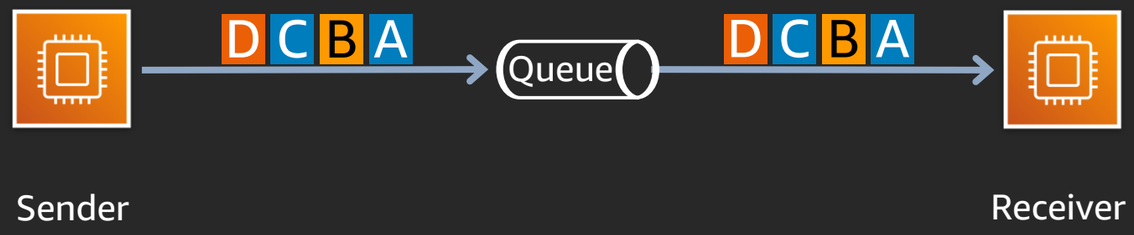


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

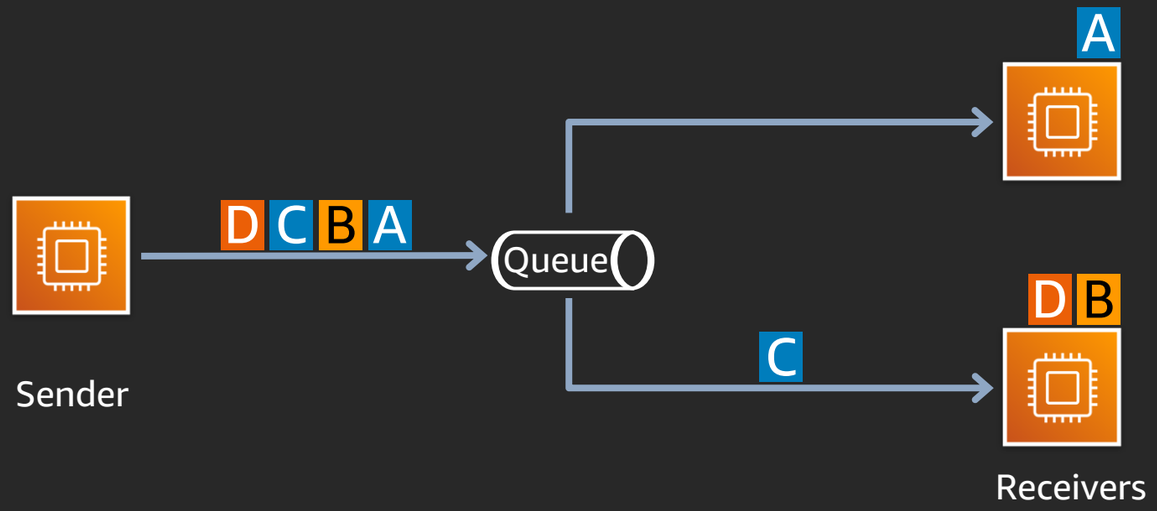
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

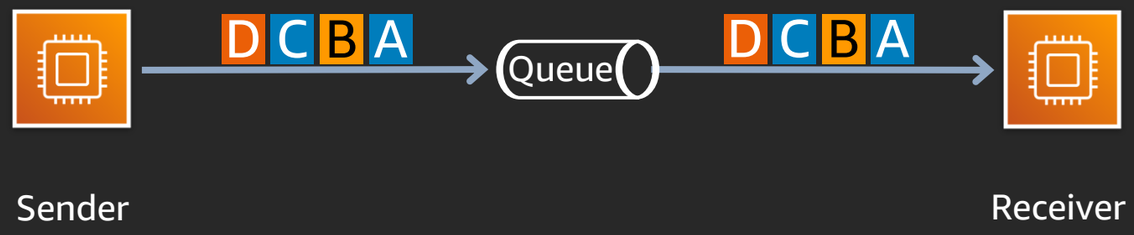


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

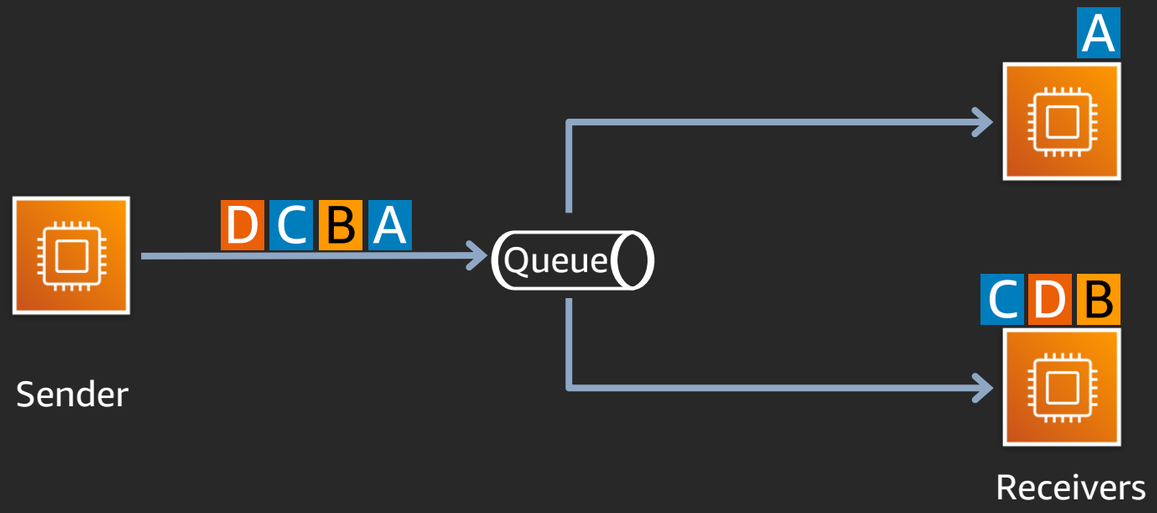
Message channels: FIFO queue

FIFO queue



Guaranteed ordering

Message groups

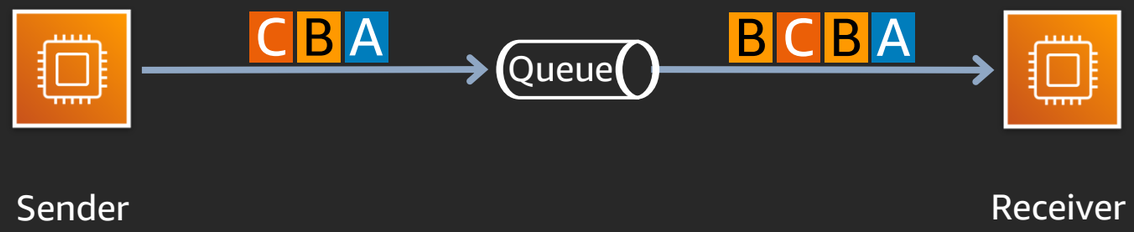


Messages grouped by discriminator attribute

Example: Amazon SQS with sender-provided message group IDs—no further consumption as long as messages with particular message group ID are invisible

Message channels: Message delivery QoS

Message delivery QoS



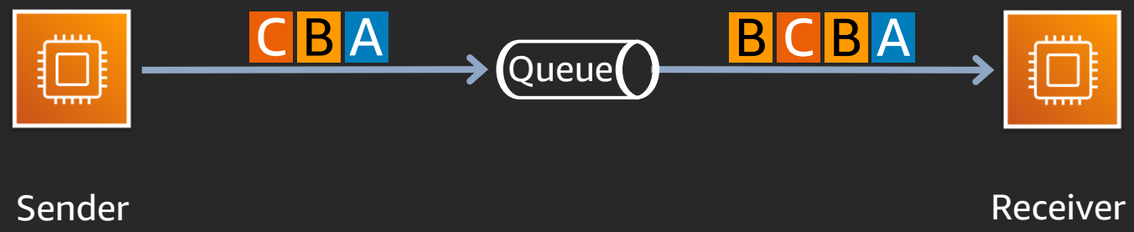
At least once

At most once

Exactly once

Message channels: Message delivery QoS

Message delivery QoS



At least once

At most once

Exactly once

Exactly once? Well!

How do you deal with a situation where the message was consumed but never acknowledged?

→ Your systems still have to be able to handle duplicate messages

→ Messages should be processed in an idempotent manner

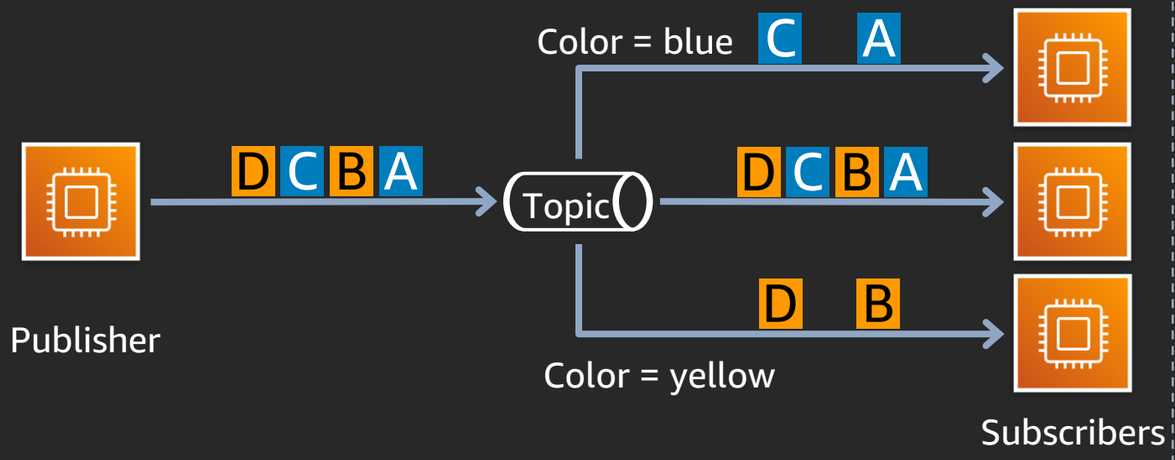
Message routing: Message filter

Message filter

Recipient list

Message routing: Message filter

Message filter



Receive only a relevant subset of messages

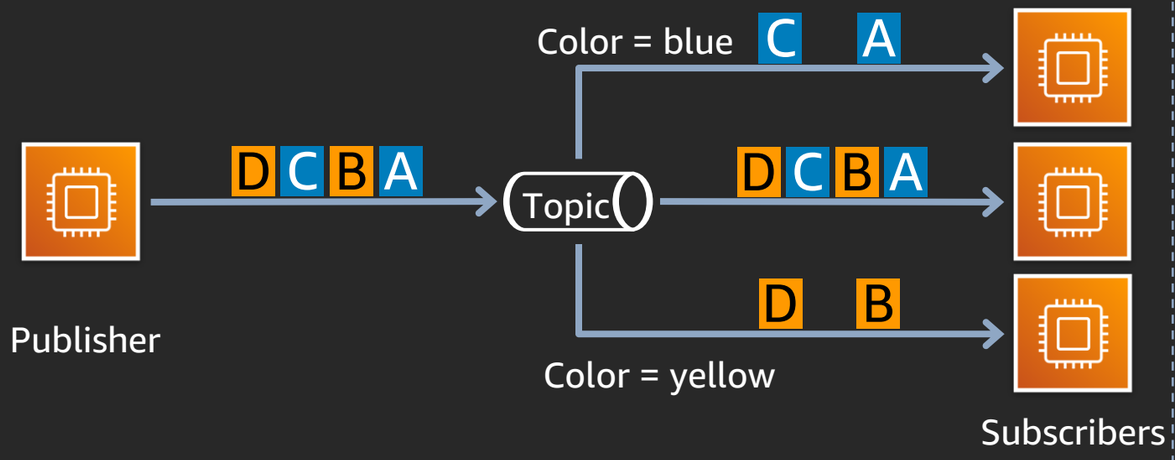
Controlled by subscriber

Publisher remains completely unaware

Recipient list

Message routing: Message filter

Message filter

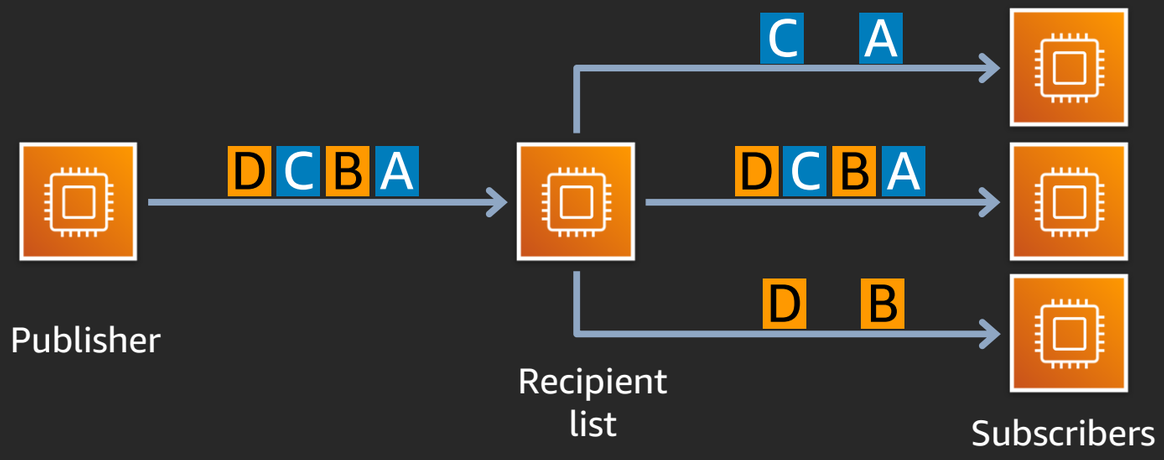


Receive only a relevant subset of messages

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: Saga orchestration

Saga orchestration

Message routing: Saga orchestration

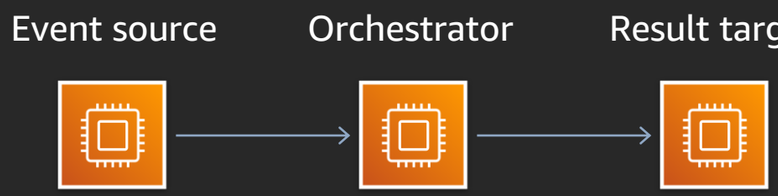
Saga orchestration

Event source

Result target

Message routing: Saga orchestration

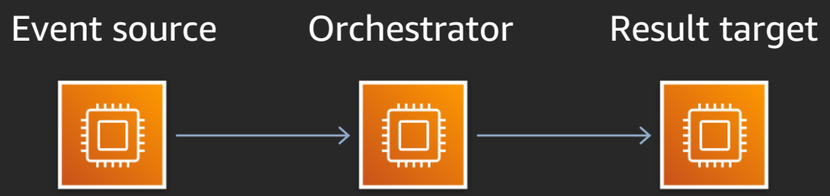
Saga orchestration



Event triggers orchestrated workflow

Message routing: Saga orchestration

Saga orchestration

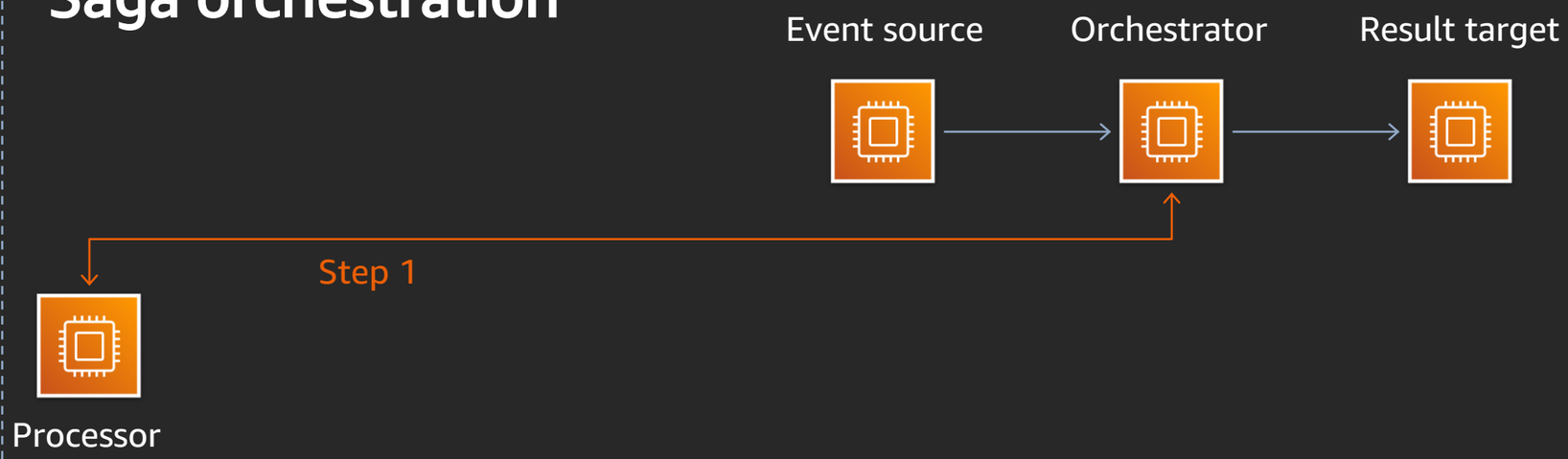


Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component and for potential rollback

Message routing: Saga orchestration

Saga orchestration



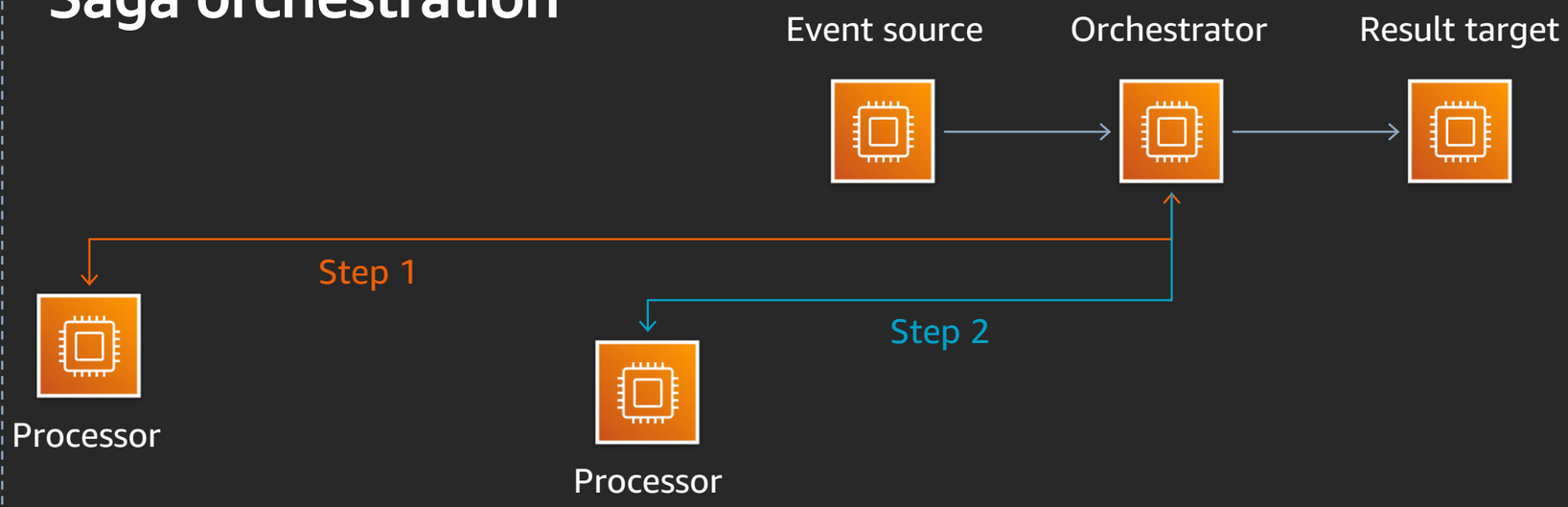
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component and for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing: Saga orchestration

Saga orchestration



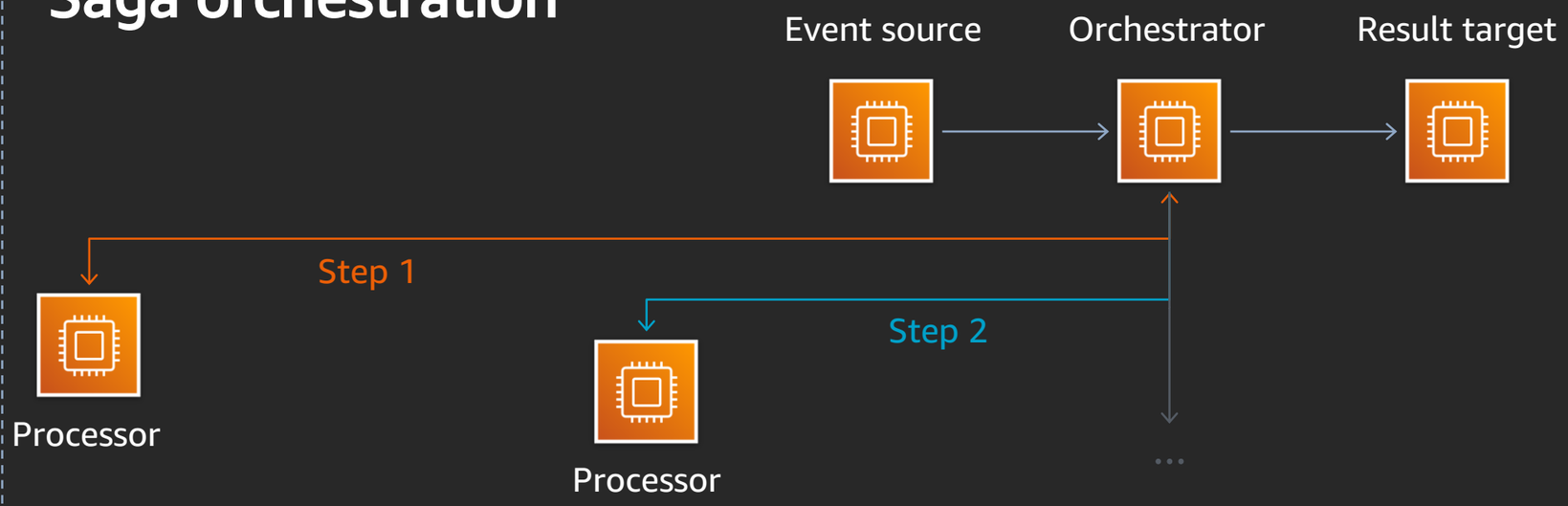
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component and for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing: Saga orchestration

Saga orchestration



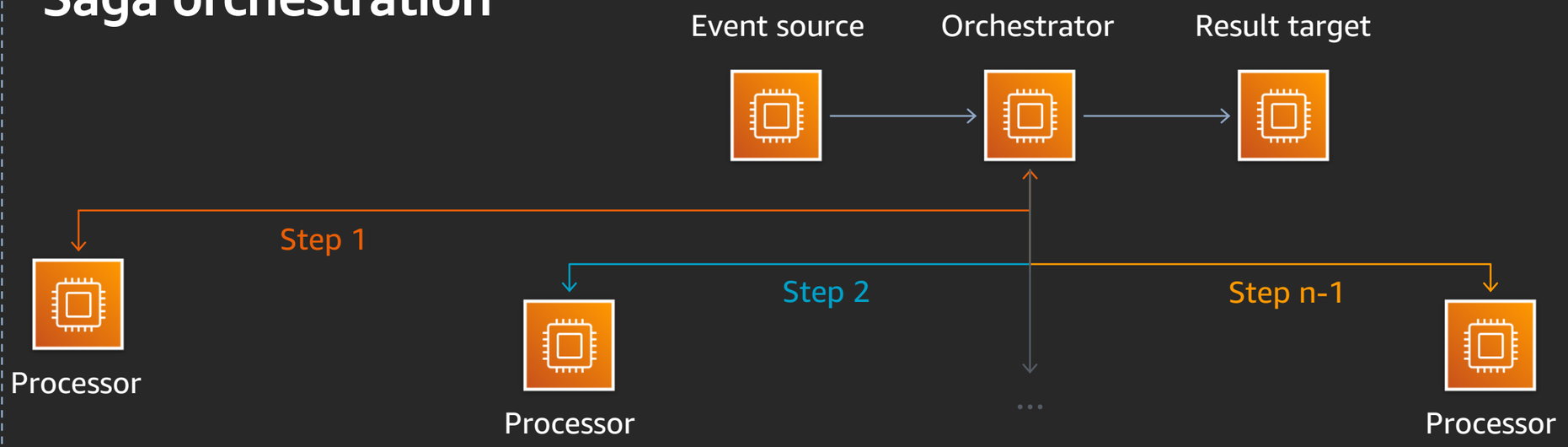
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component and for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing: Saga orchestration

Saga orchestration



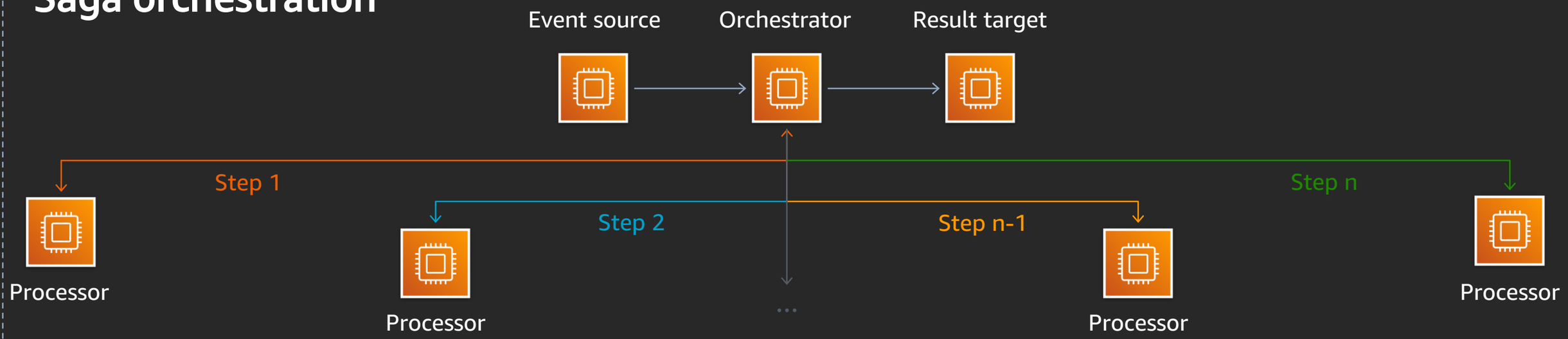
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component and for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing: Saga orchestration

Saga orchestration



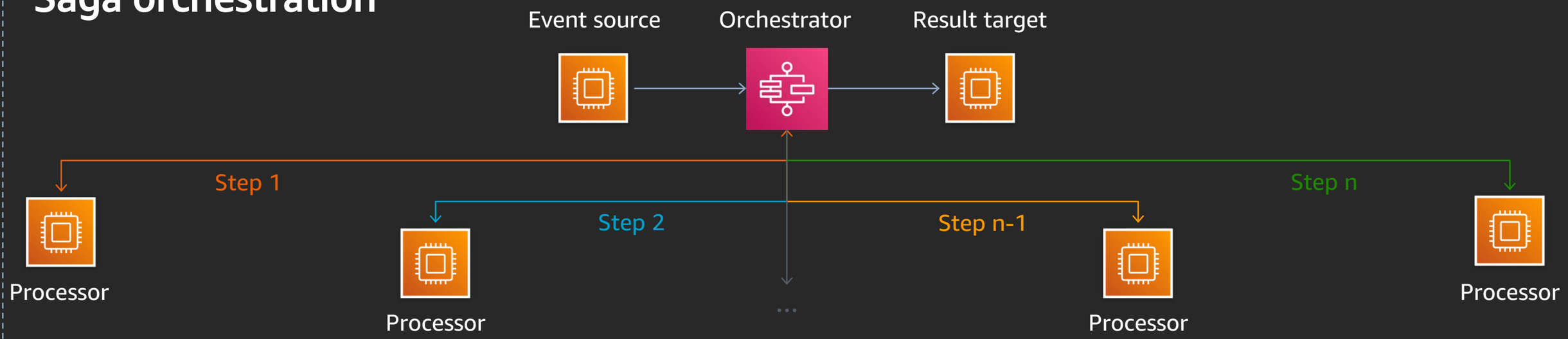
Event triggers orchestrated workflow

Knowledge of workflow is externalized into orchestrator component and for potential rollback

Workflow participants remain as loosely coupled as possible

Message routing: AWS Step Functions

Saga orchestration



AWS service for saga orchestration (serverless)

AWS Step Functions

Relevant use cases

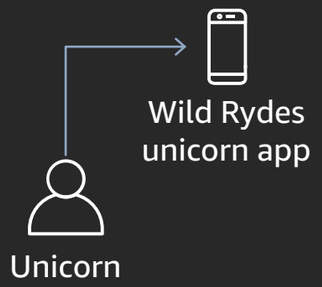
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Context: Wild Rydes, Inc.

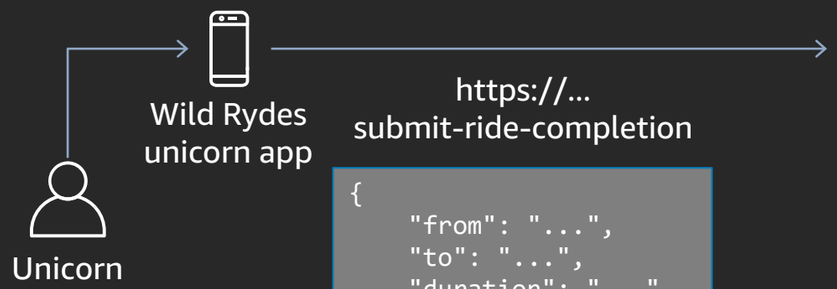
Use case: Submit a ride completion

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

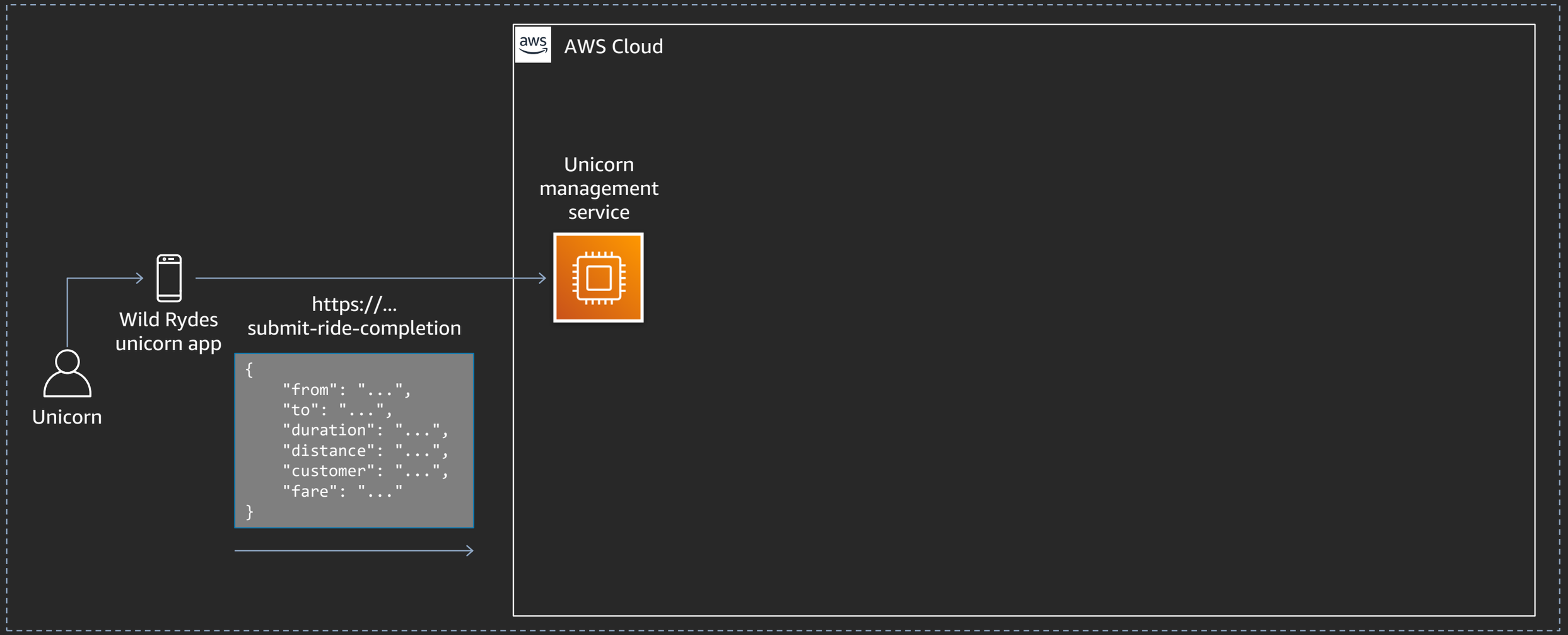
Use case: Submit a ride completion



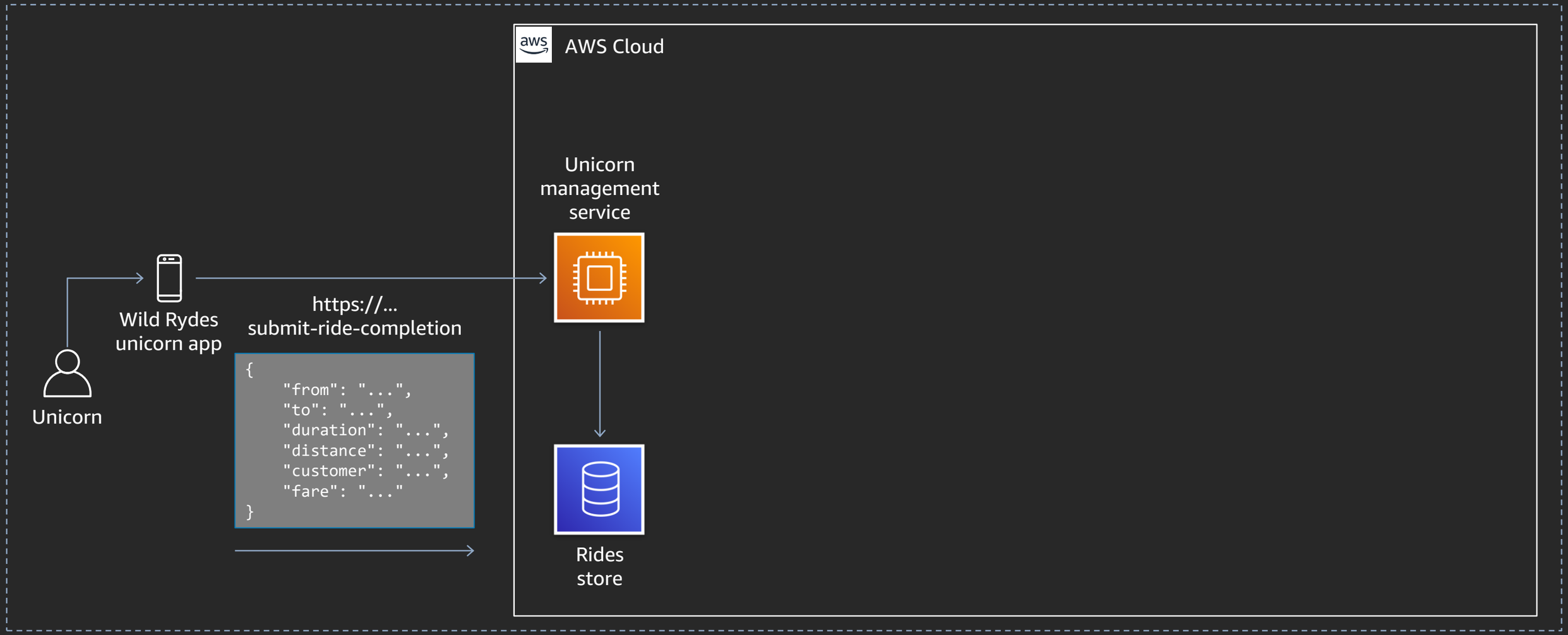
Use case: Submit a ride completion



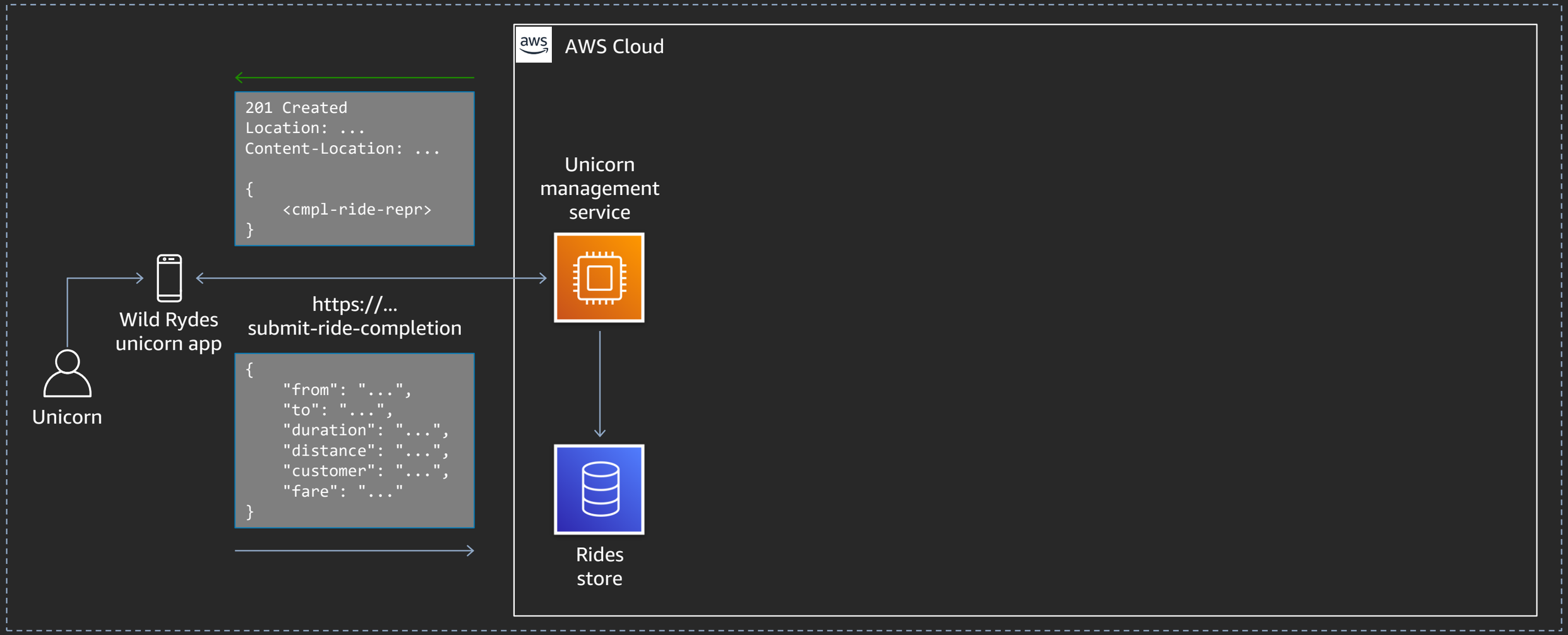
Use case: Submit a ride completion



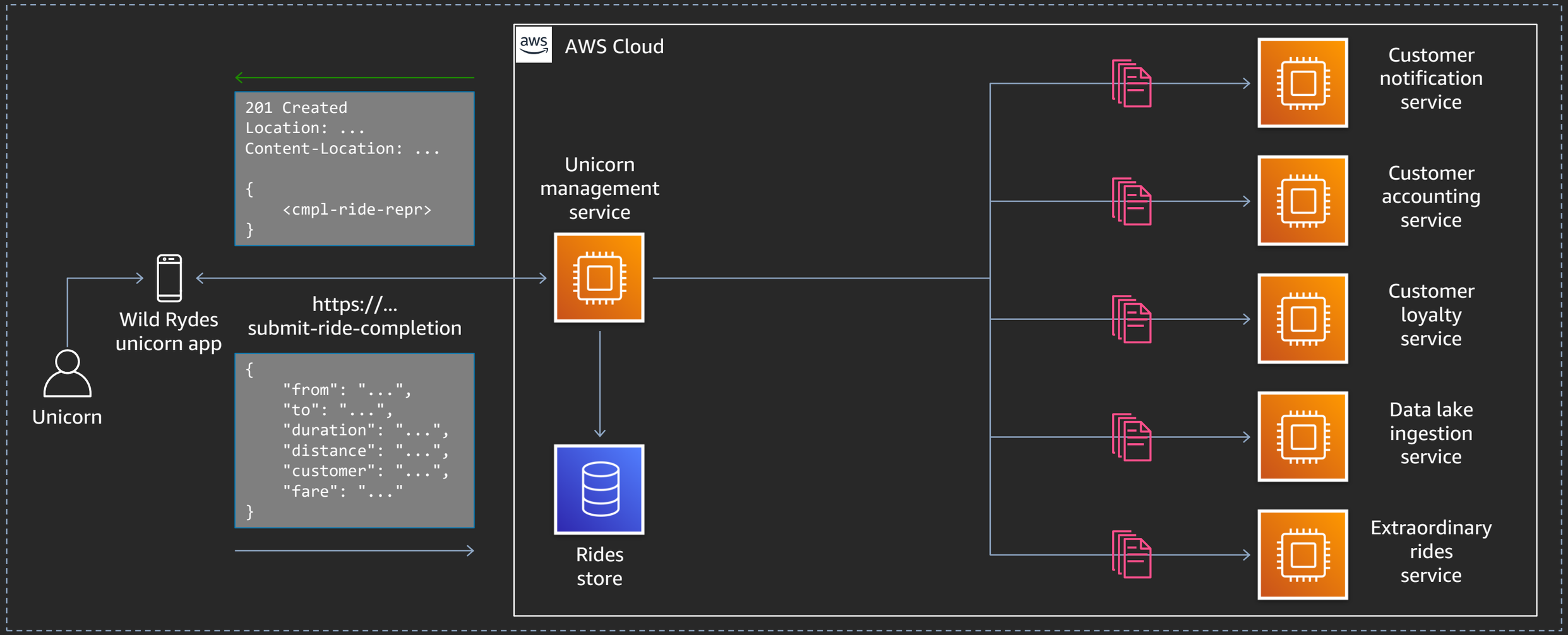
Use case: Submit a ride completion



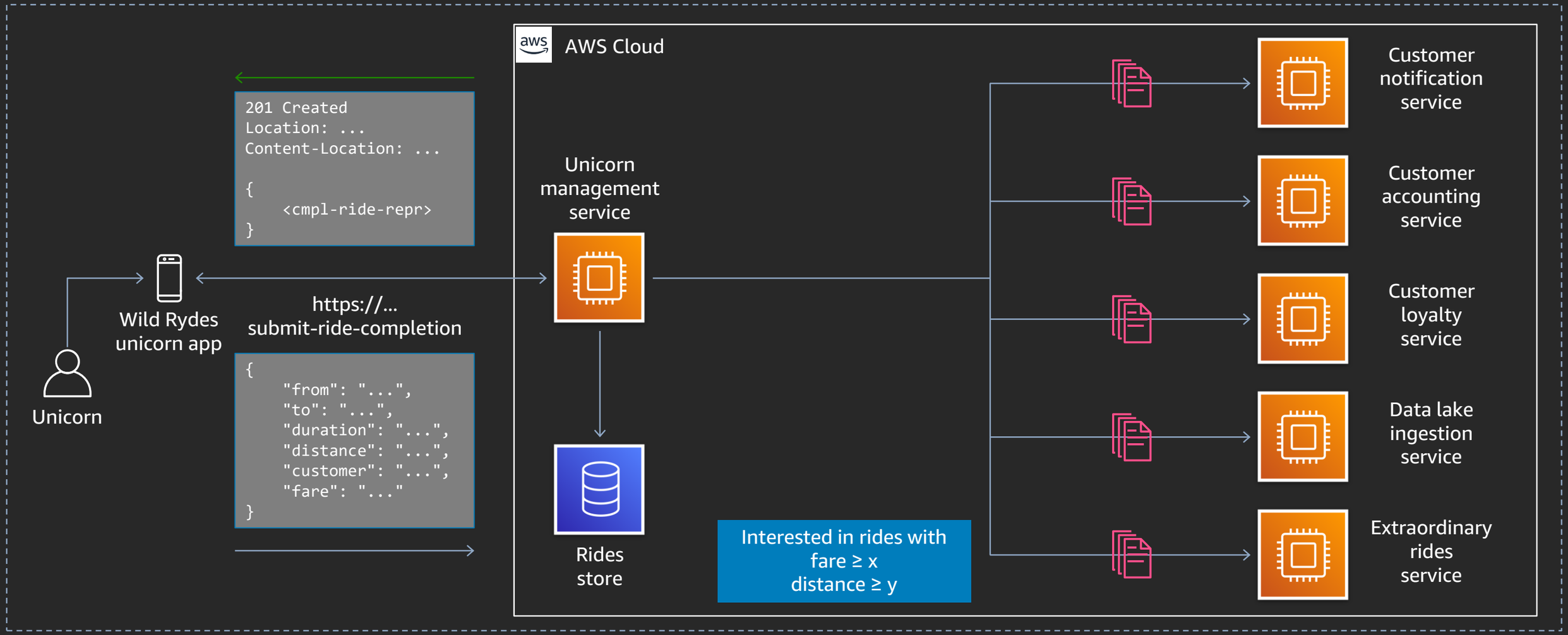
Use case: Submit a ride completion



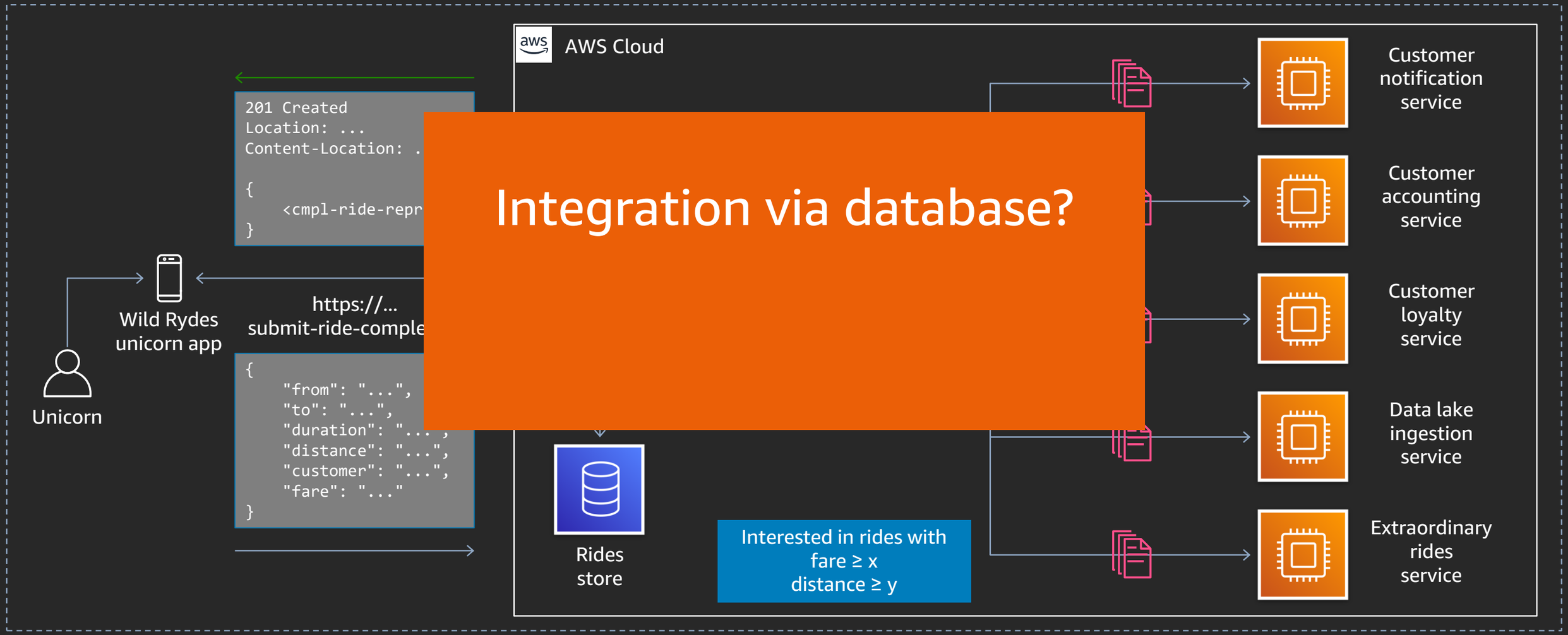
Use case: Submit a ride completion



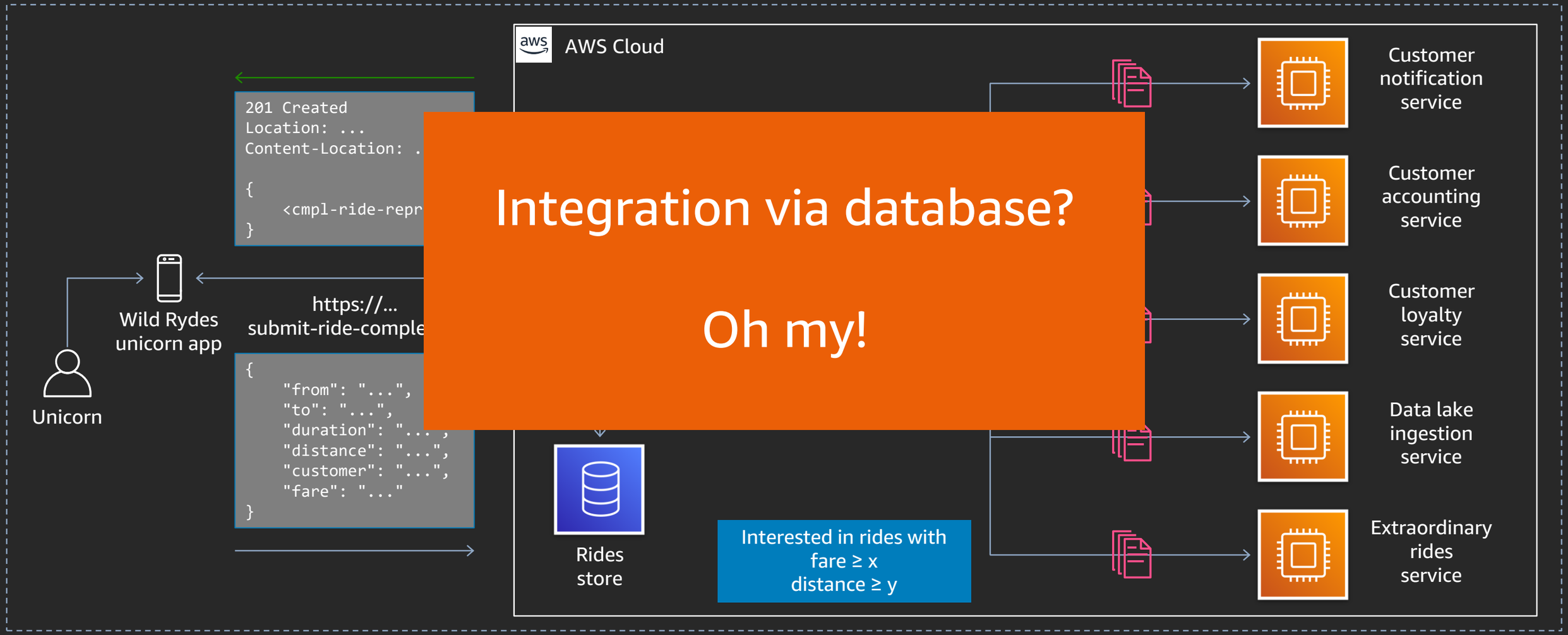
Use case: Submit a ride completion



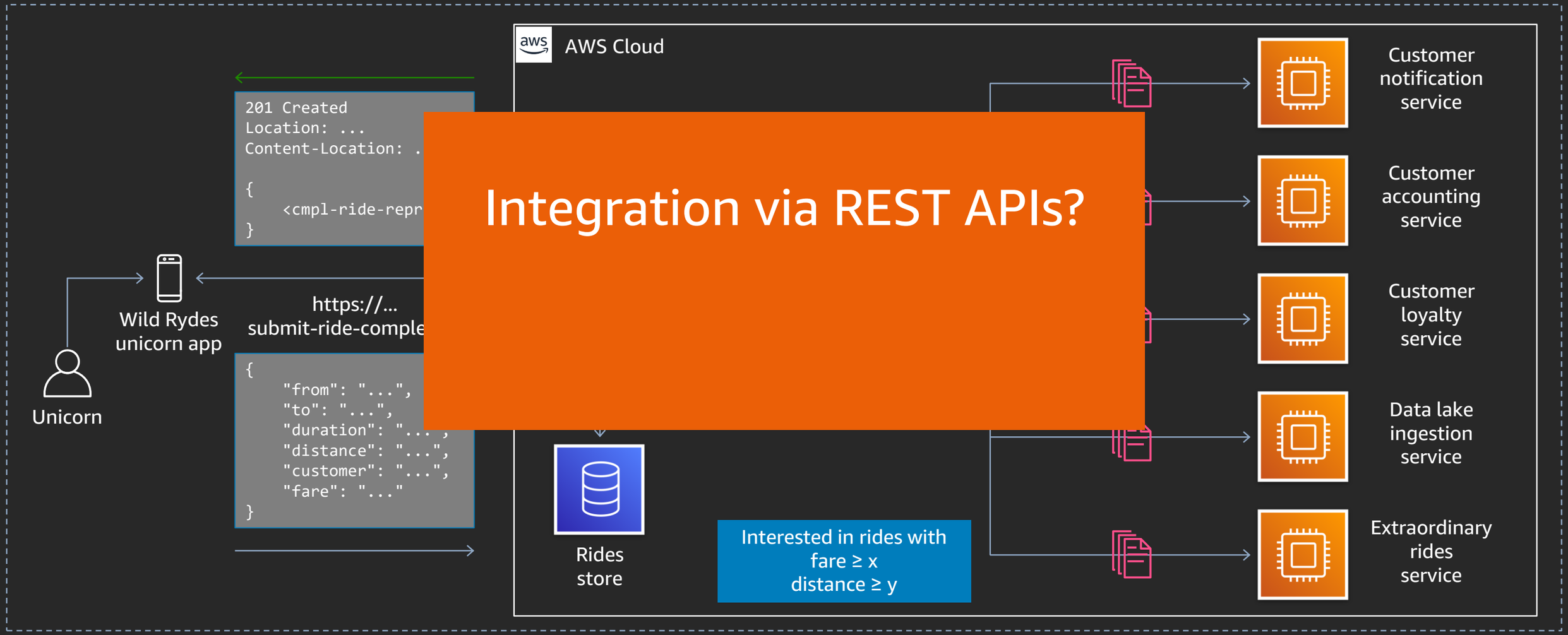
Use case: Submit a ride completion



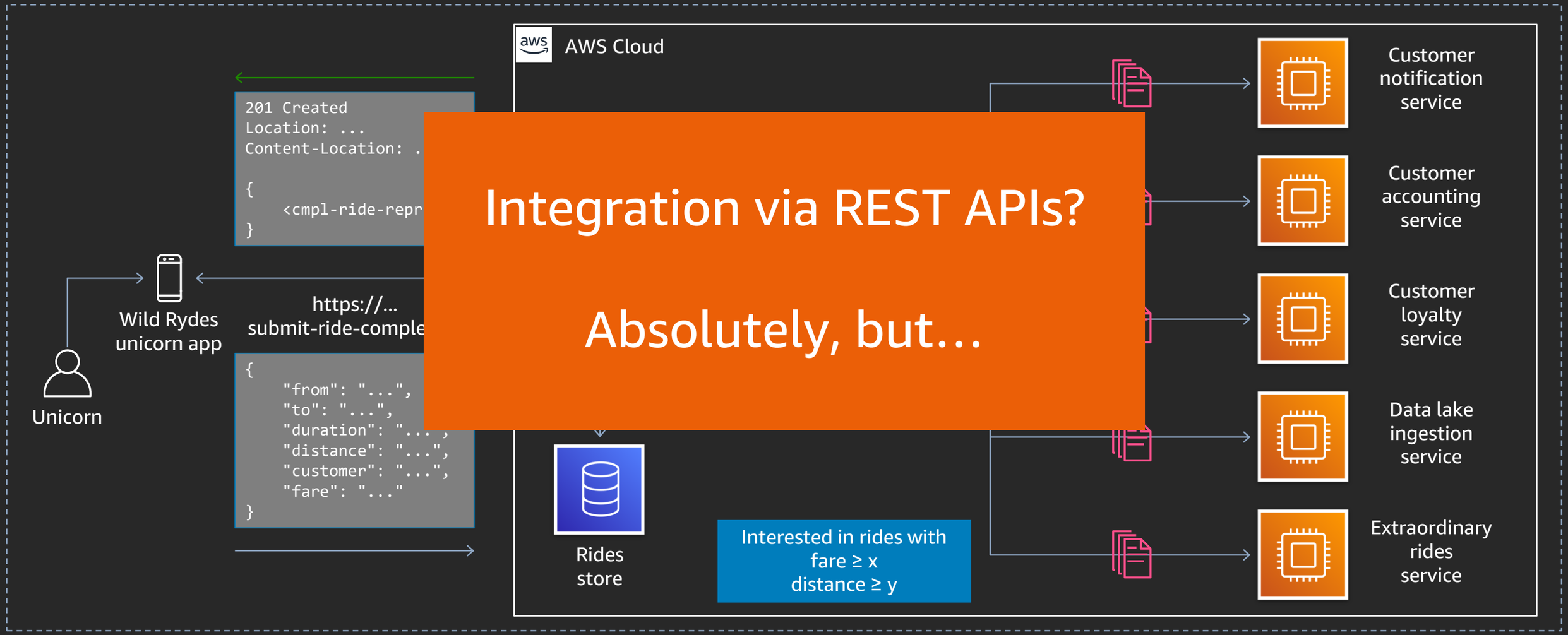
Use case: Submit a ride completion



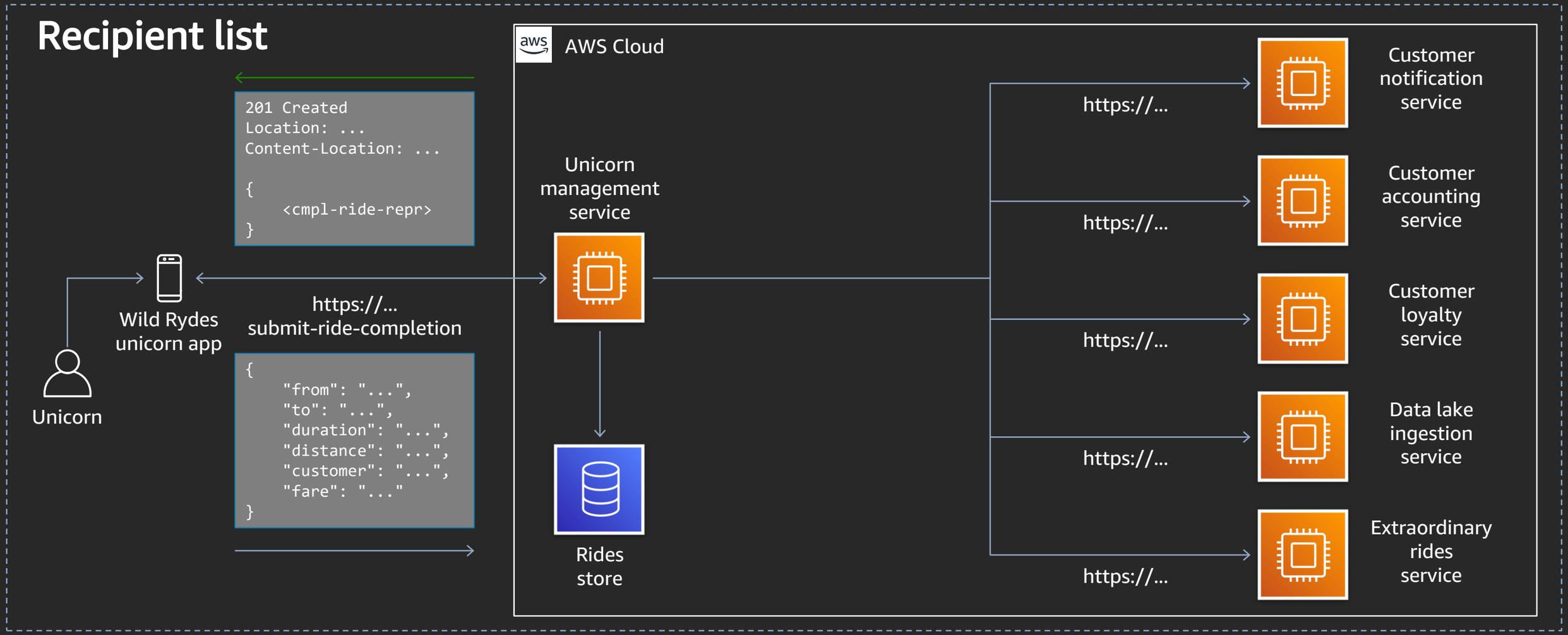
Use case: Submit a ride completion



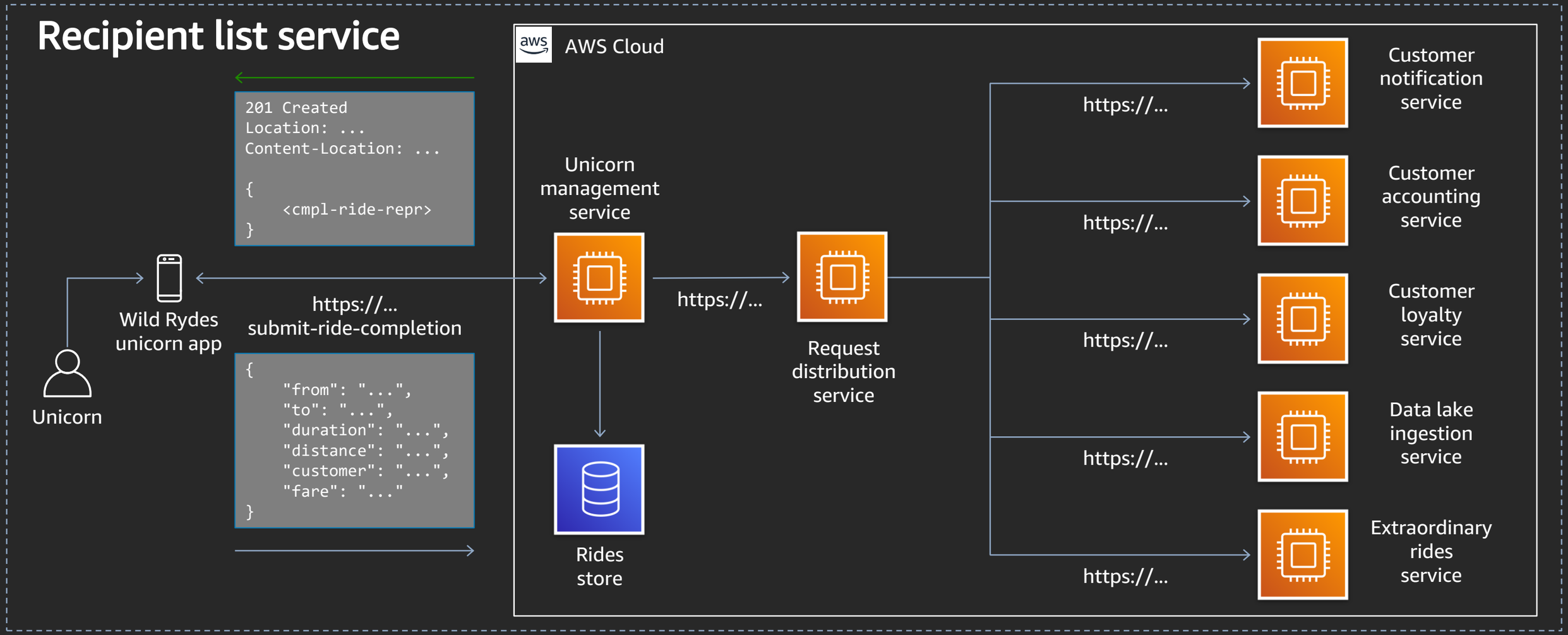
Use case: Submit a ride completion



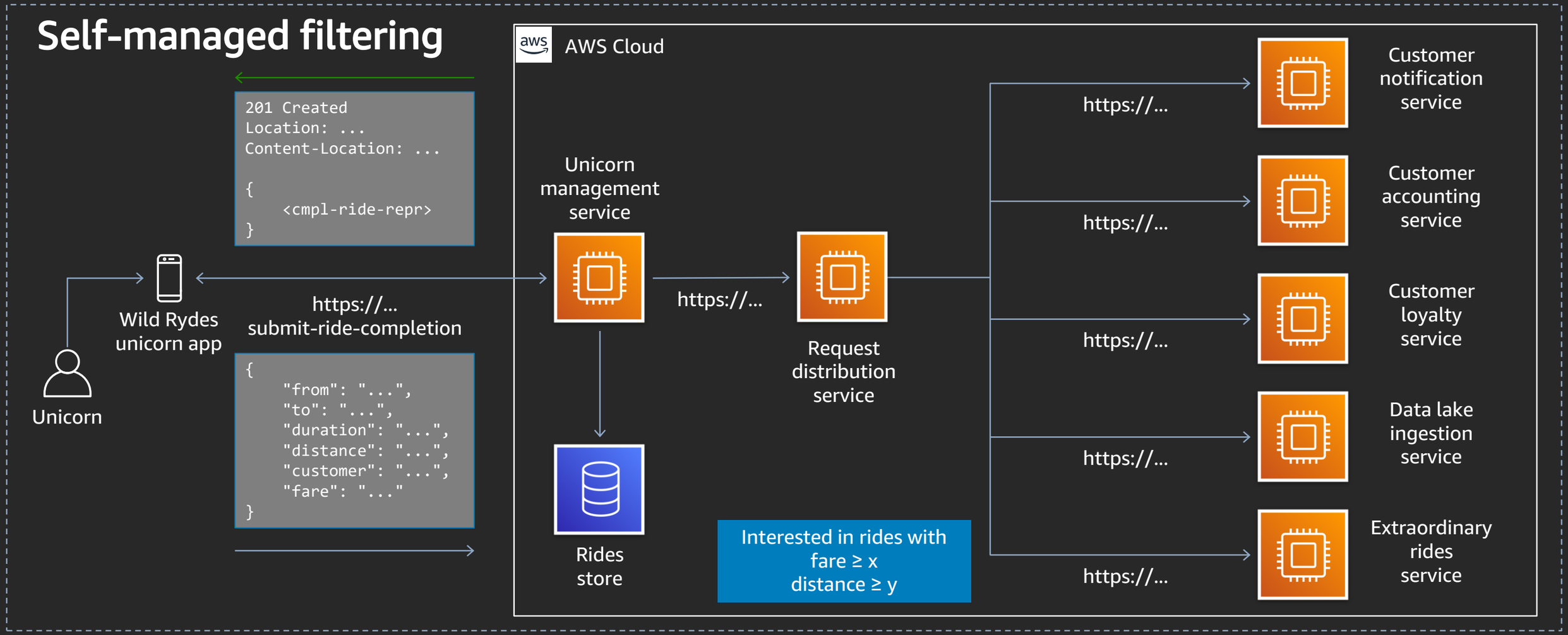
Use case: Submit a ride completion



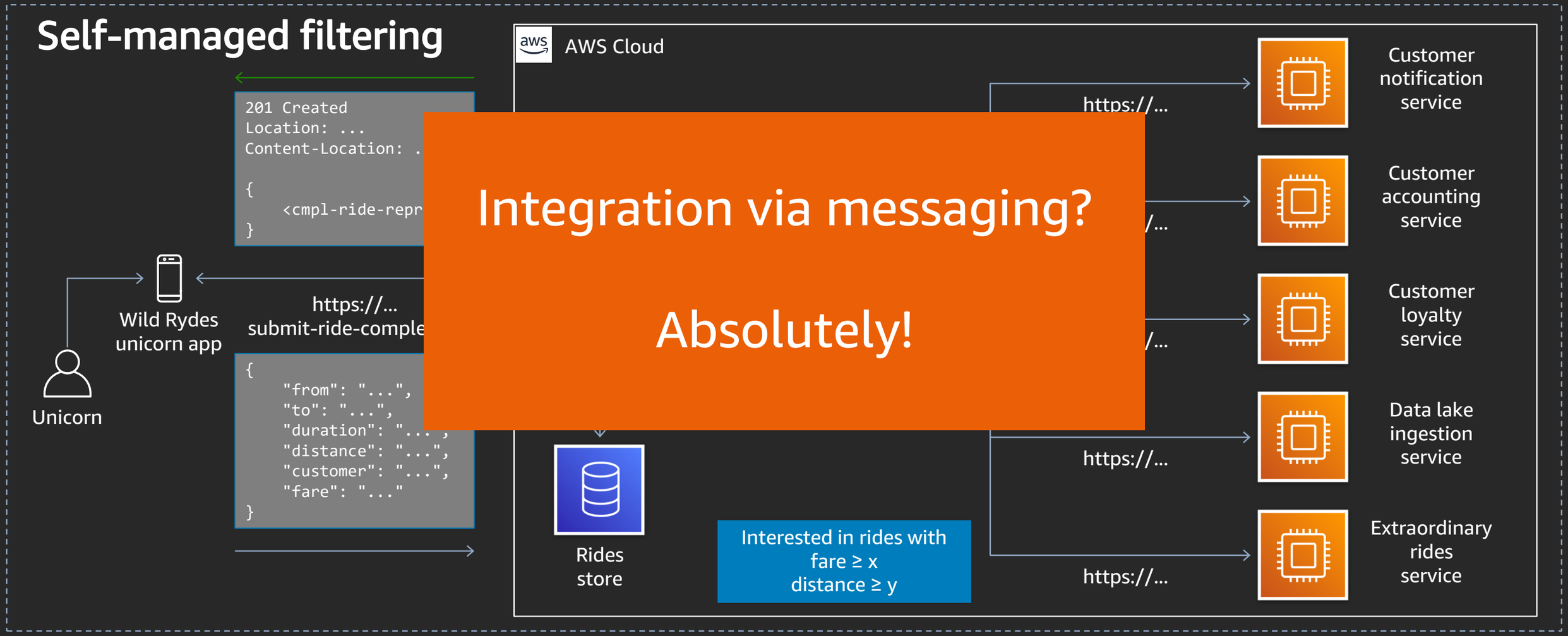
Use case: Submit a ride completion



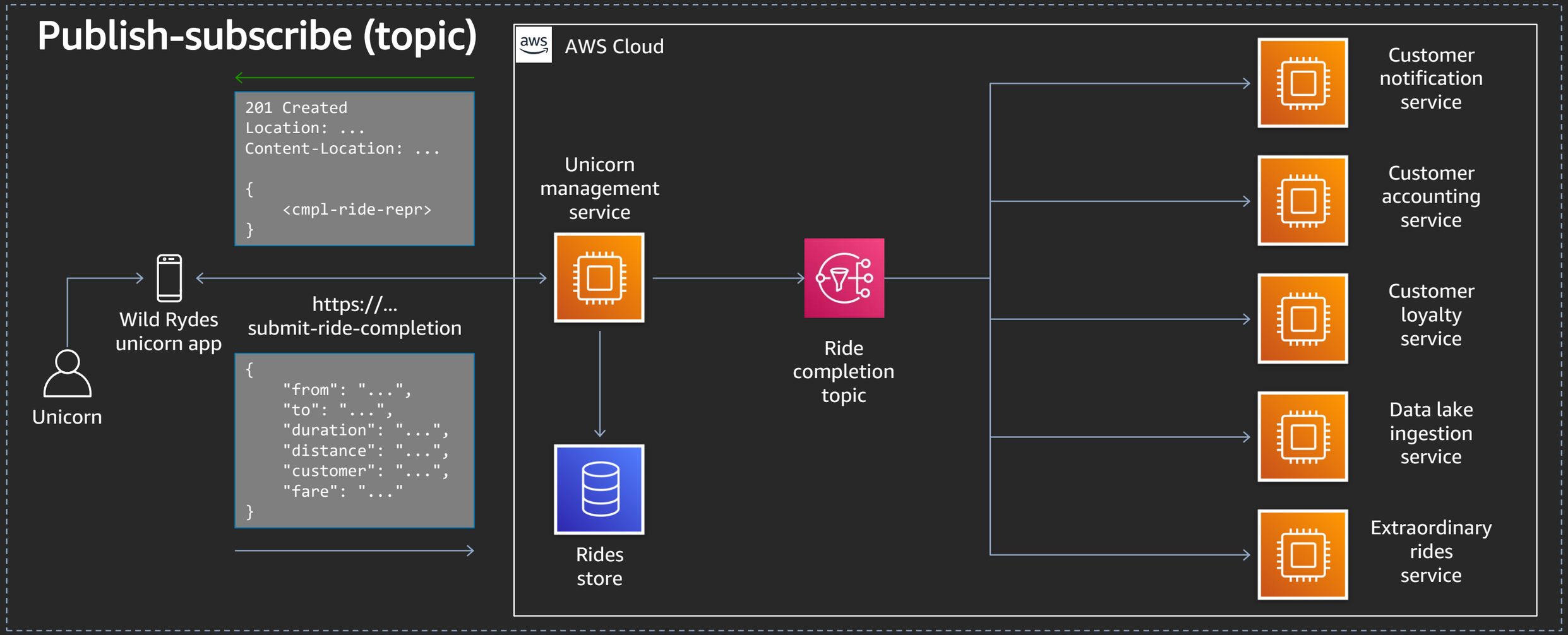
Use case: Submit a ride completion



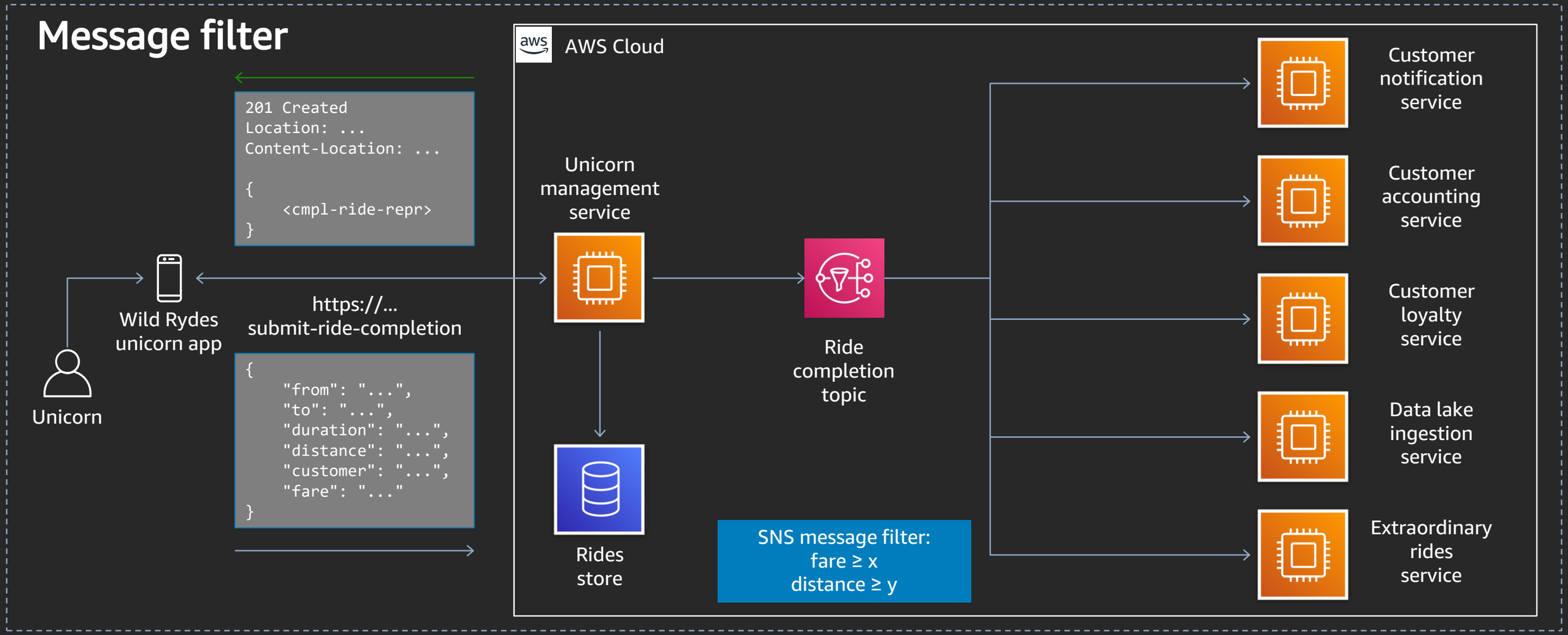
Use case: Submit a ride completion



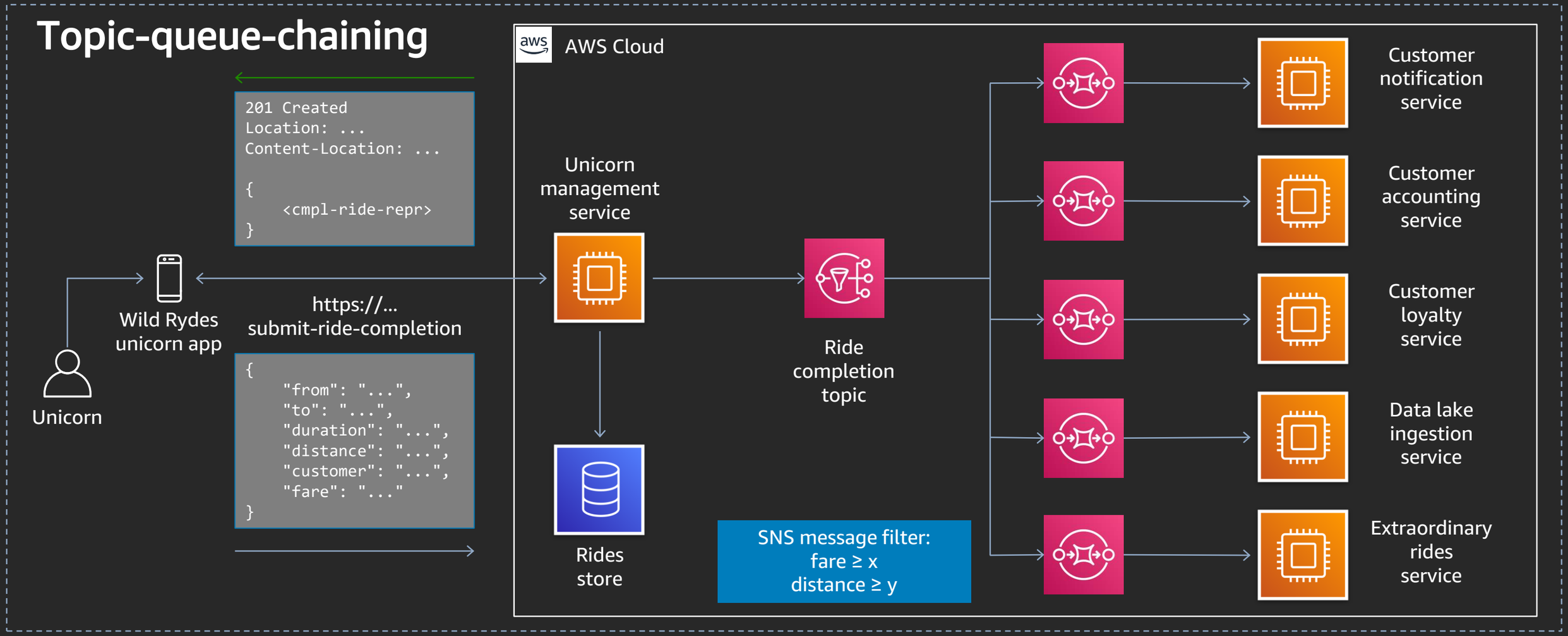
Use case: Submit a ride completion



Use case: Submit a ride completion



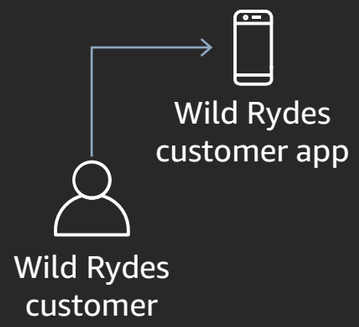
Use case: Submit a ride completion



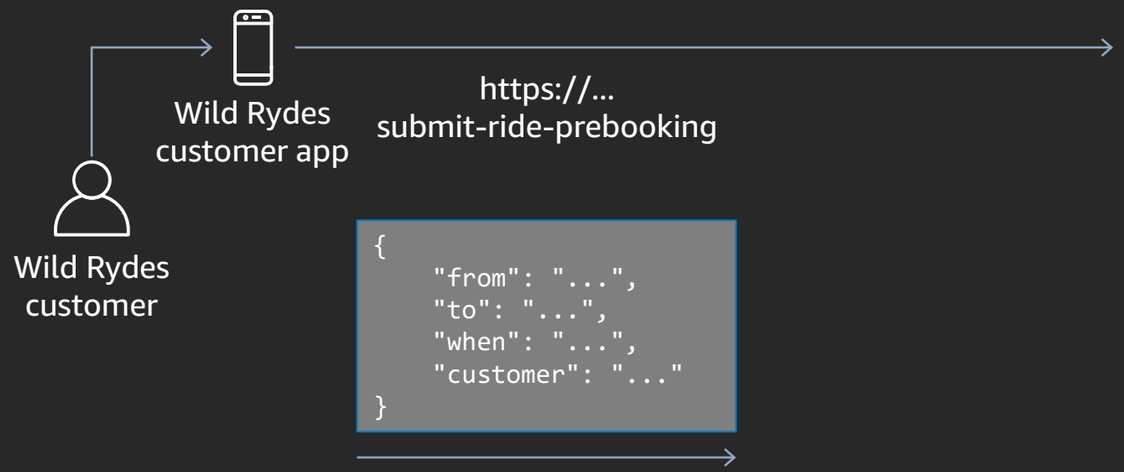
Use case: Prebooking campaigns

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

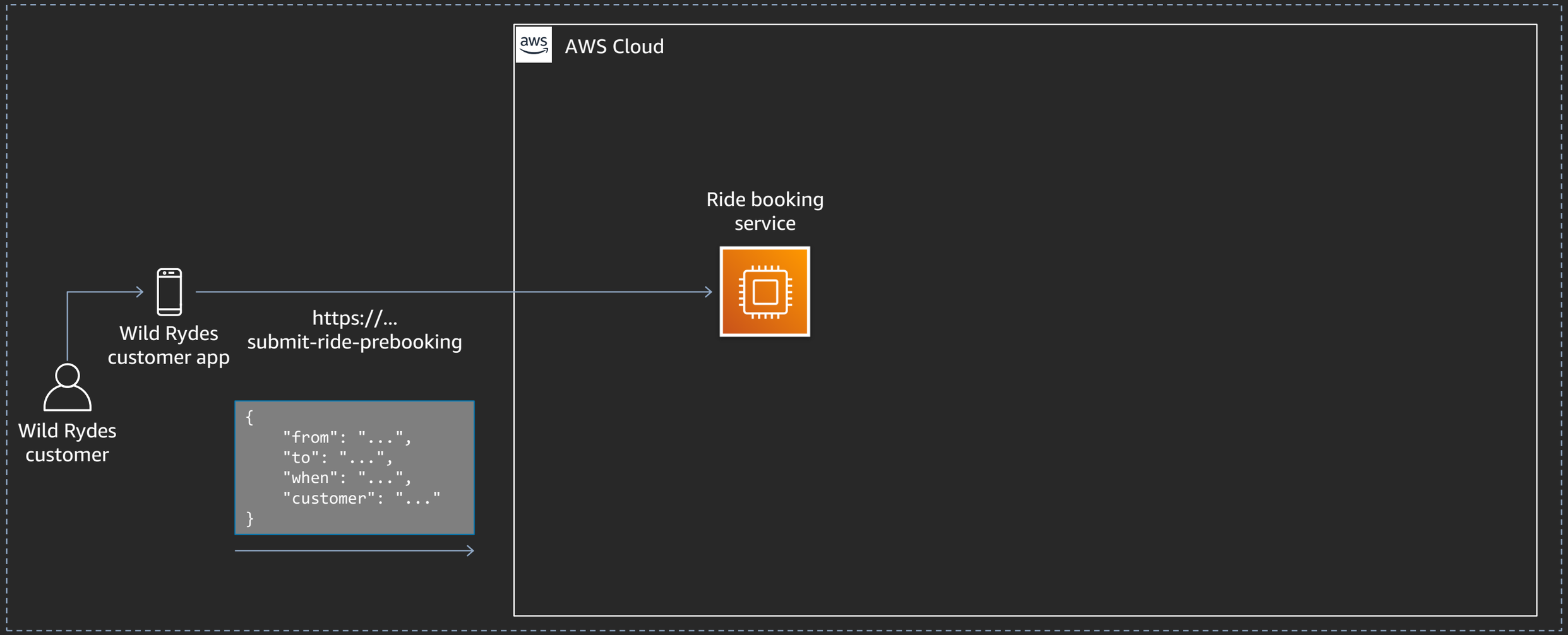
Use case: Prebooking campaigns



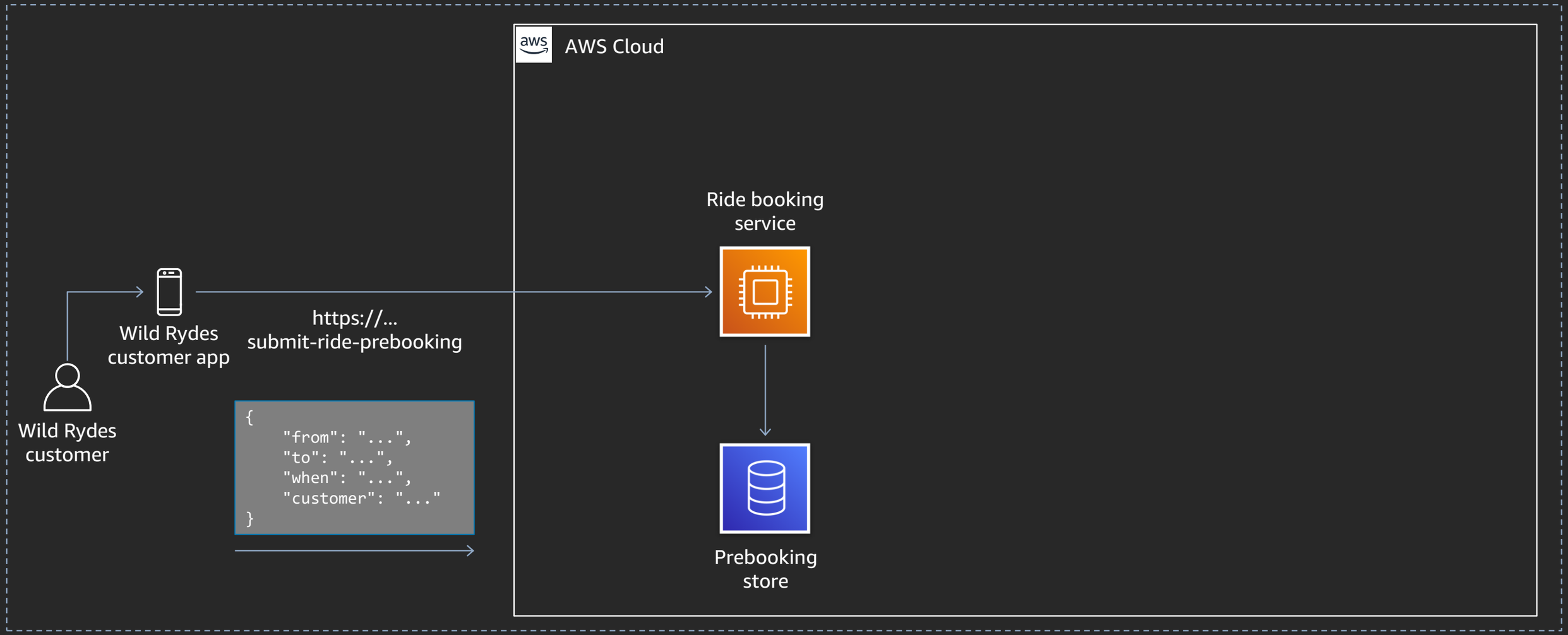
Use case: Prebooking campaigns



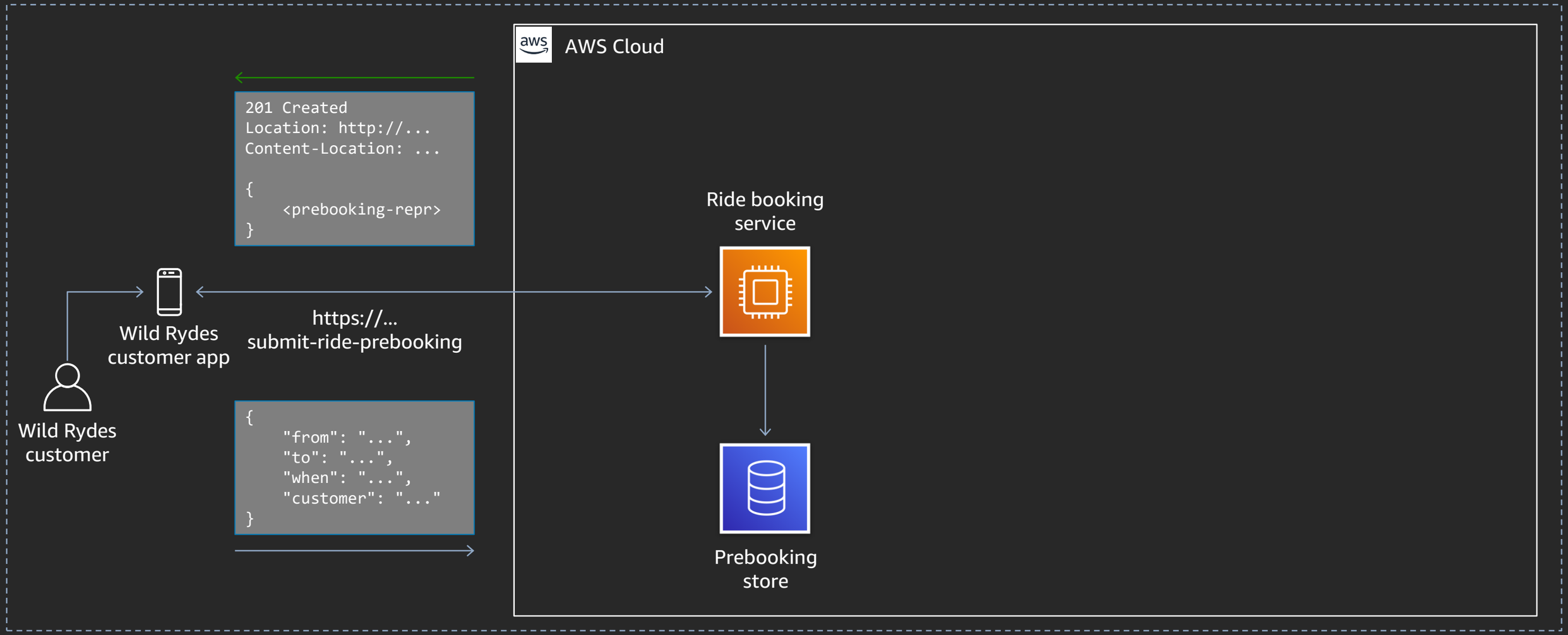
Use case: Prebooking campaigns



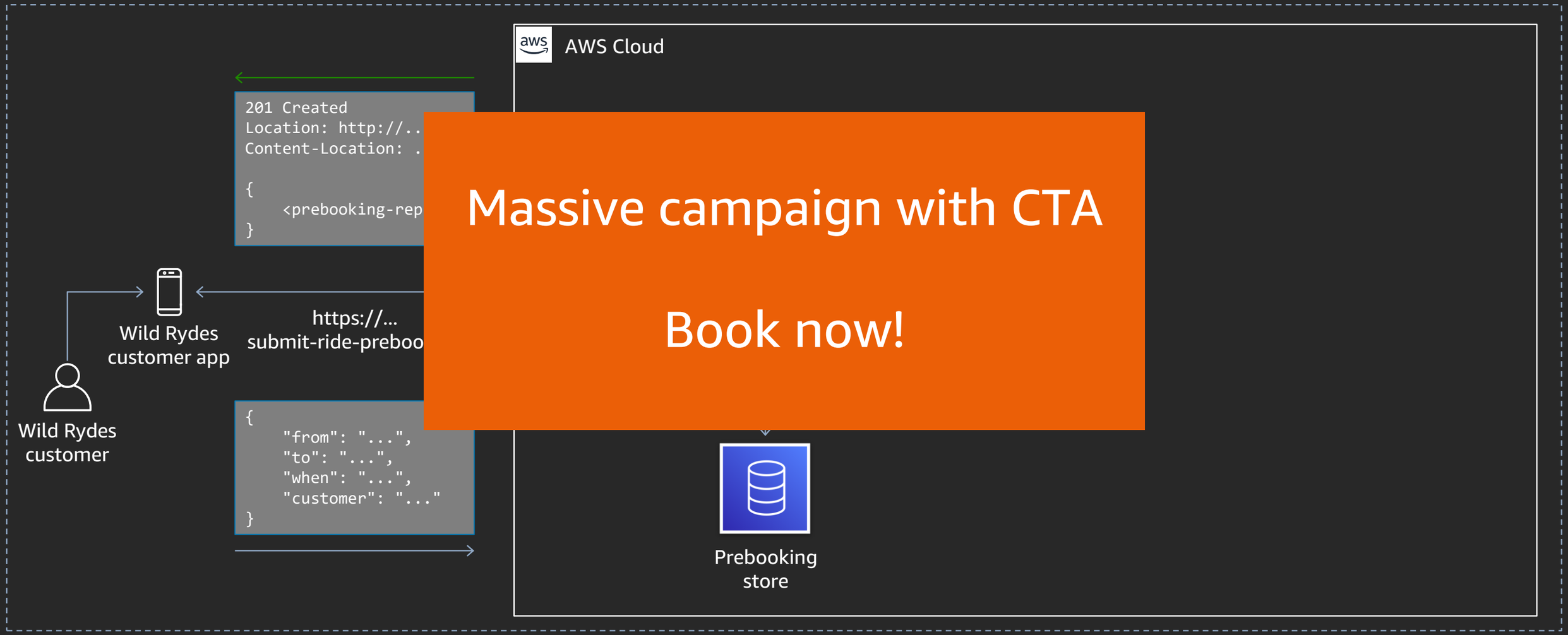
Use case: Prebooking campaigns



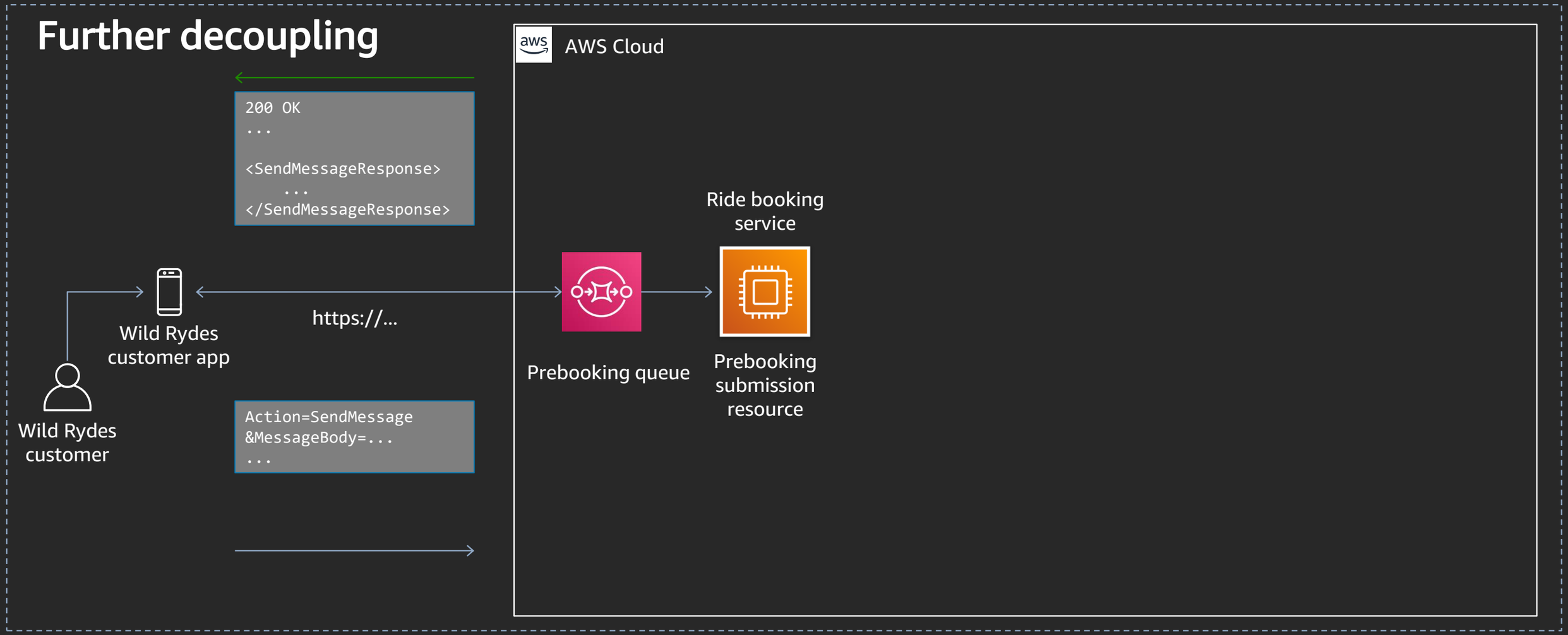
Use case: Prebooking campaigns



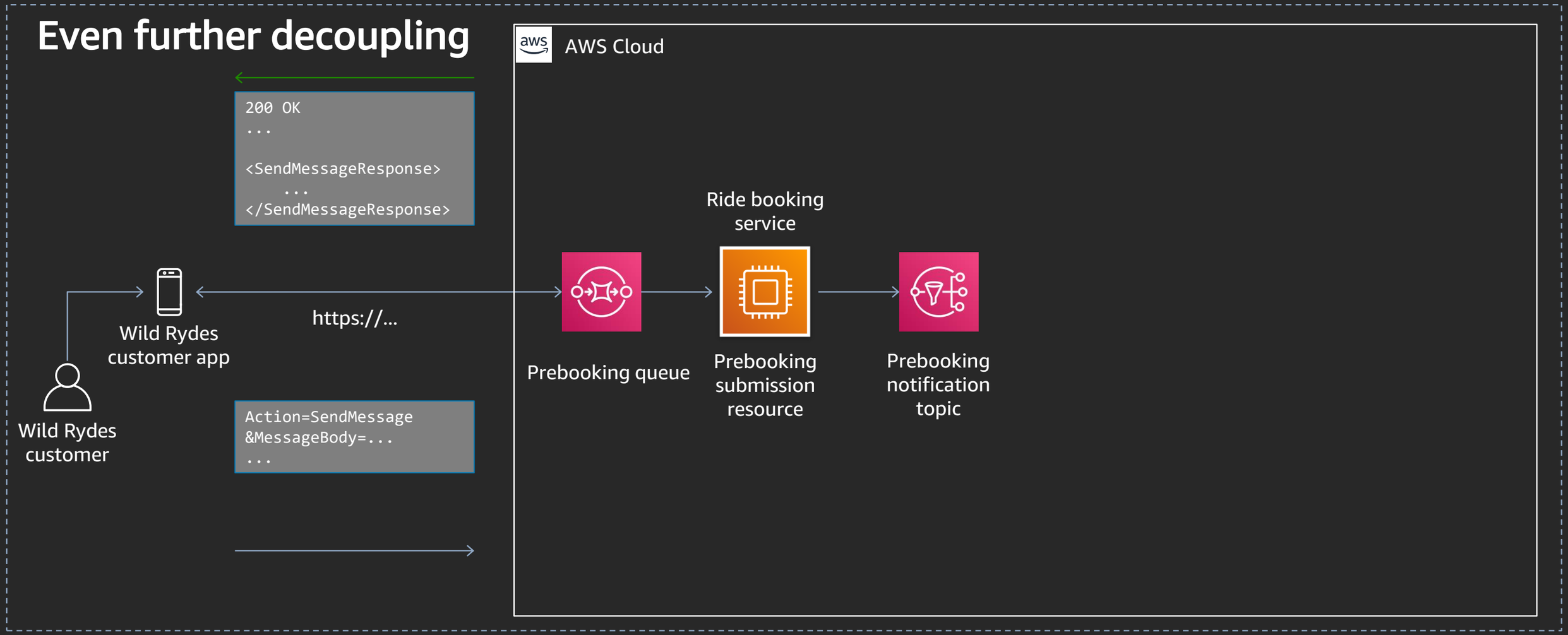
Use case: Prebooking campaigns



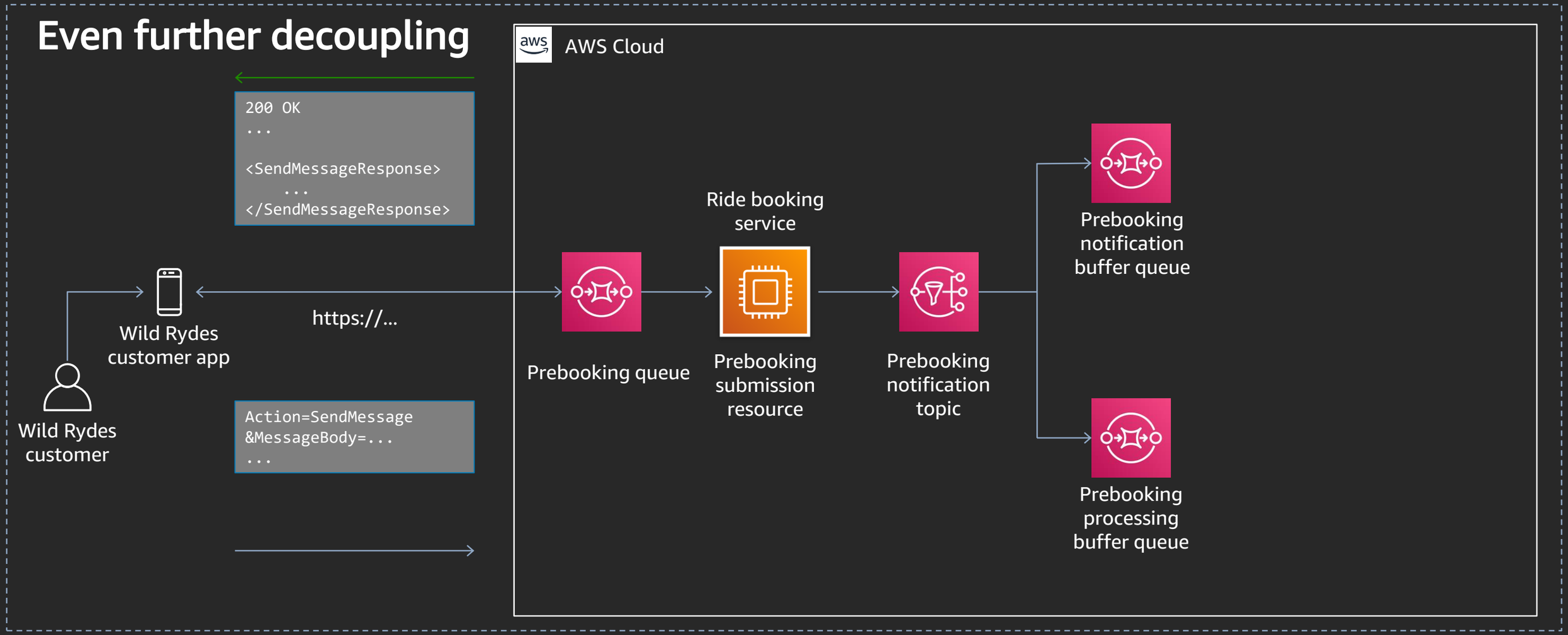
Use case: Prebooking campaigns



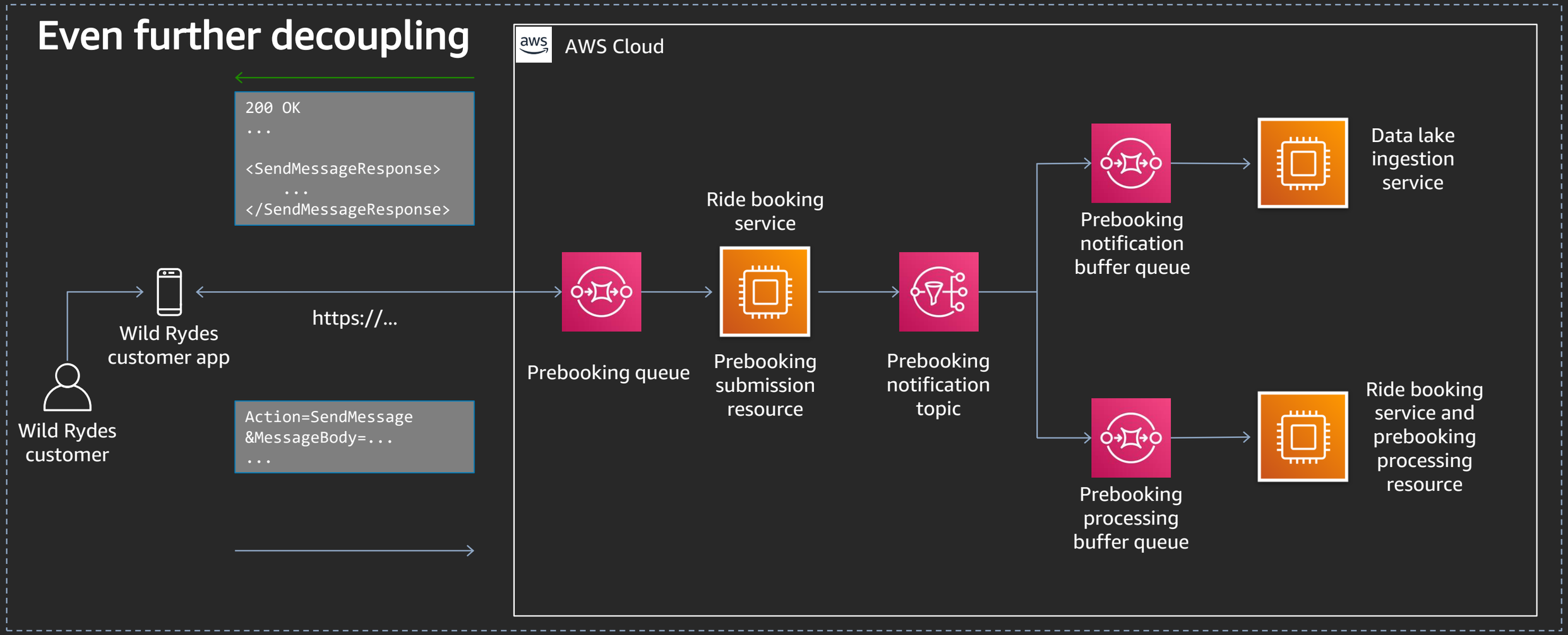
Use case: Prebooking campaigns



Use case: Prebooking campaigns



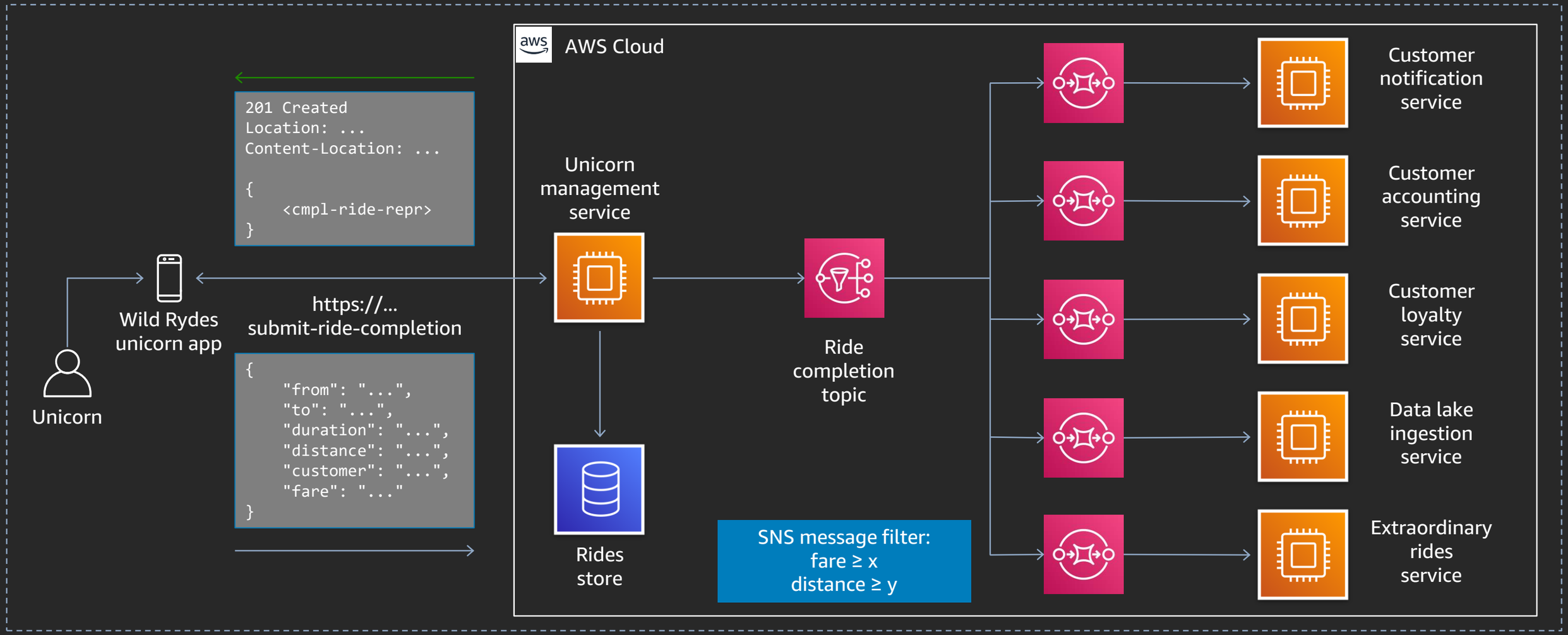
Use case: Prebooking campaigns



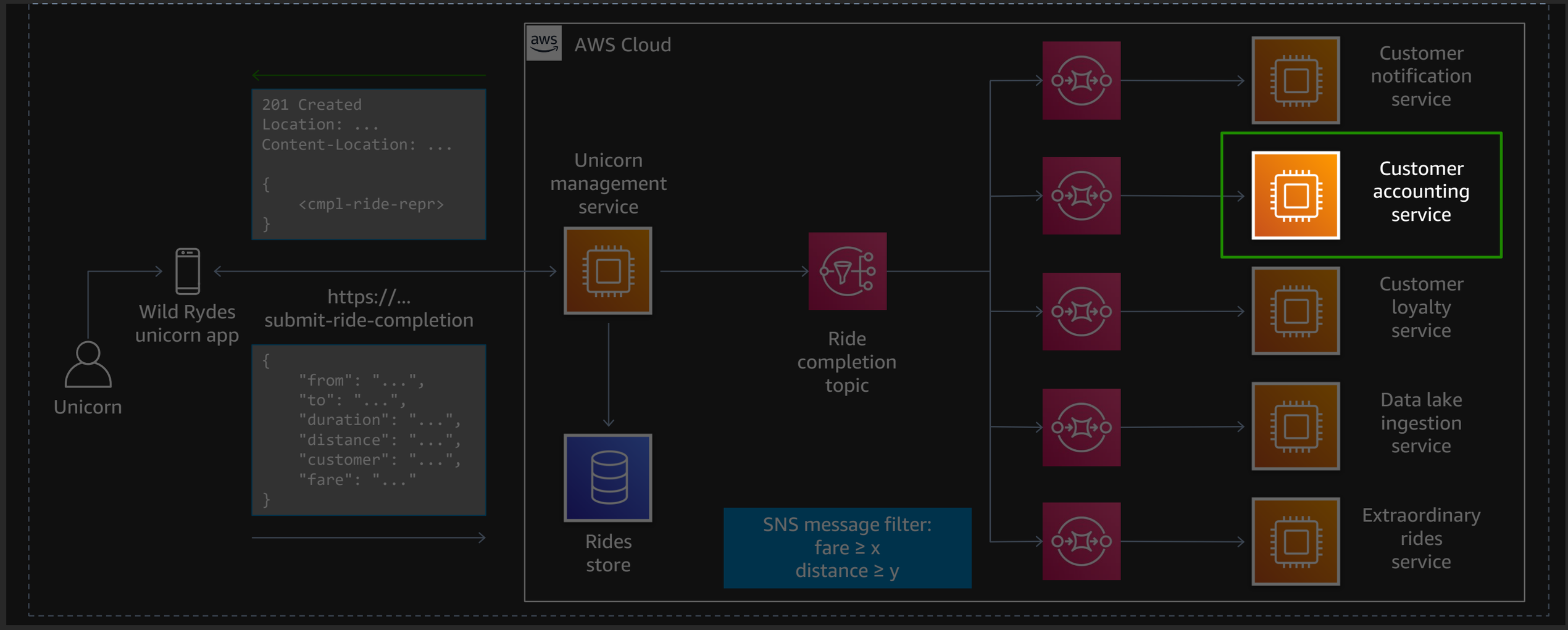
Use case: Fare collection

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

Use case: Submit a ride completion



Use case: Submit a ride completion

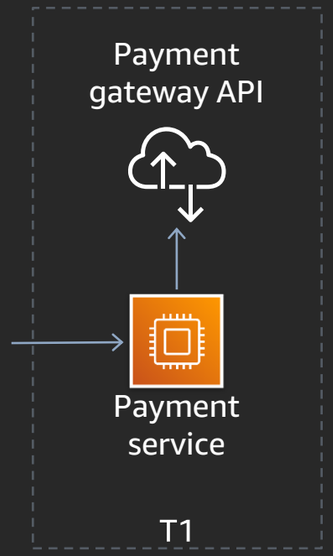


Use case: Fare collection

Saga orchestration

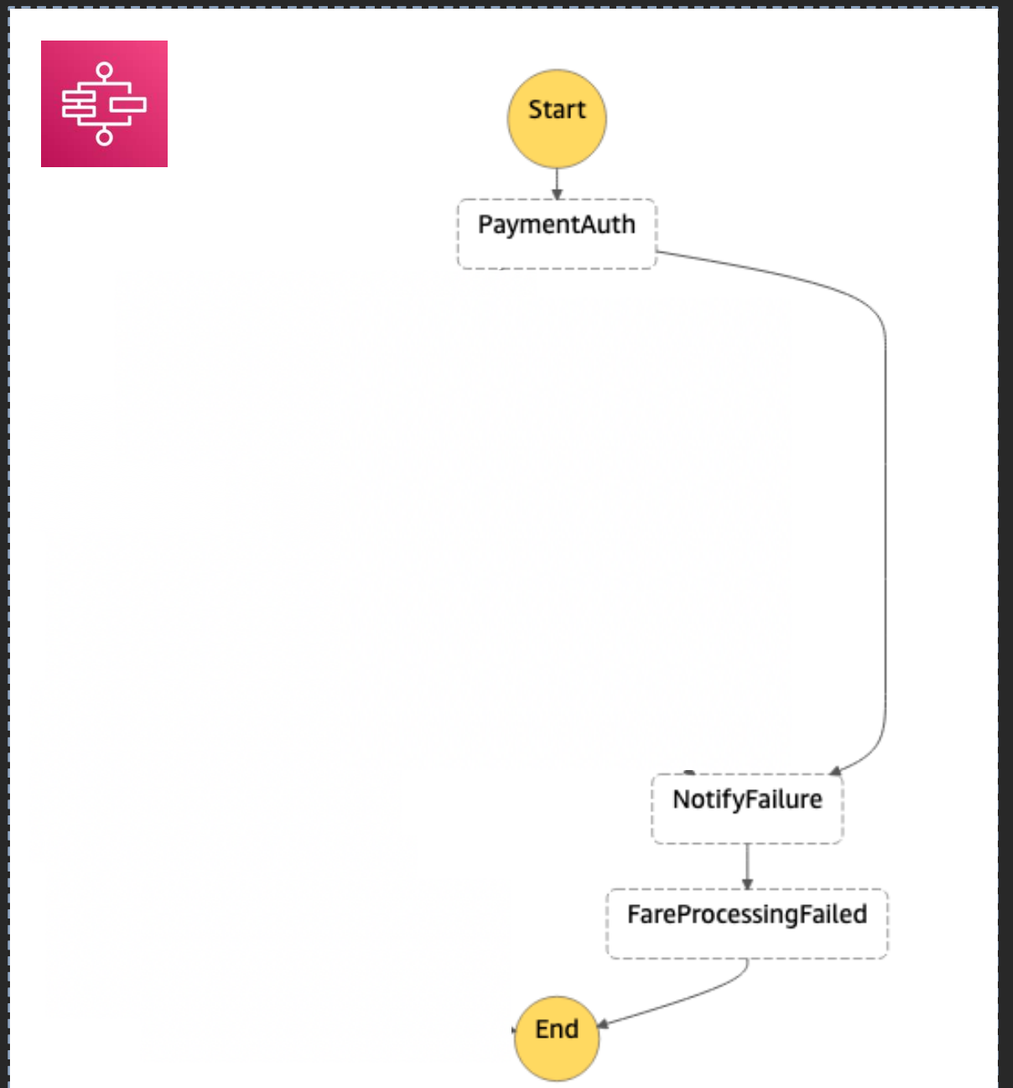
Use case: Fare collection

Saga orchestration



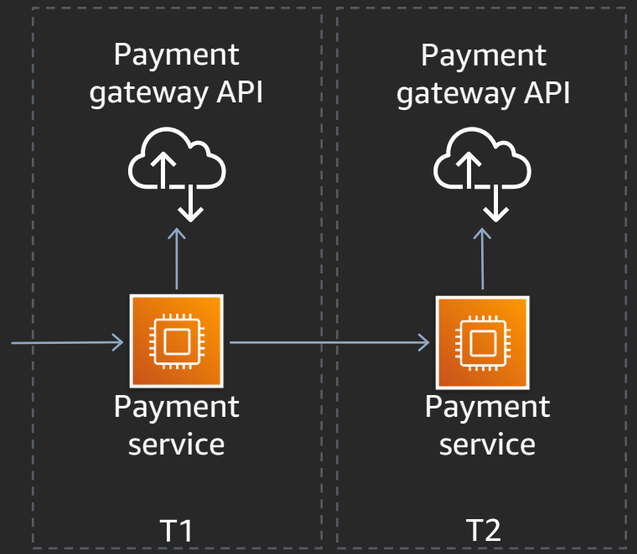
Discrete transactions

1. Credit card preauthorization



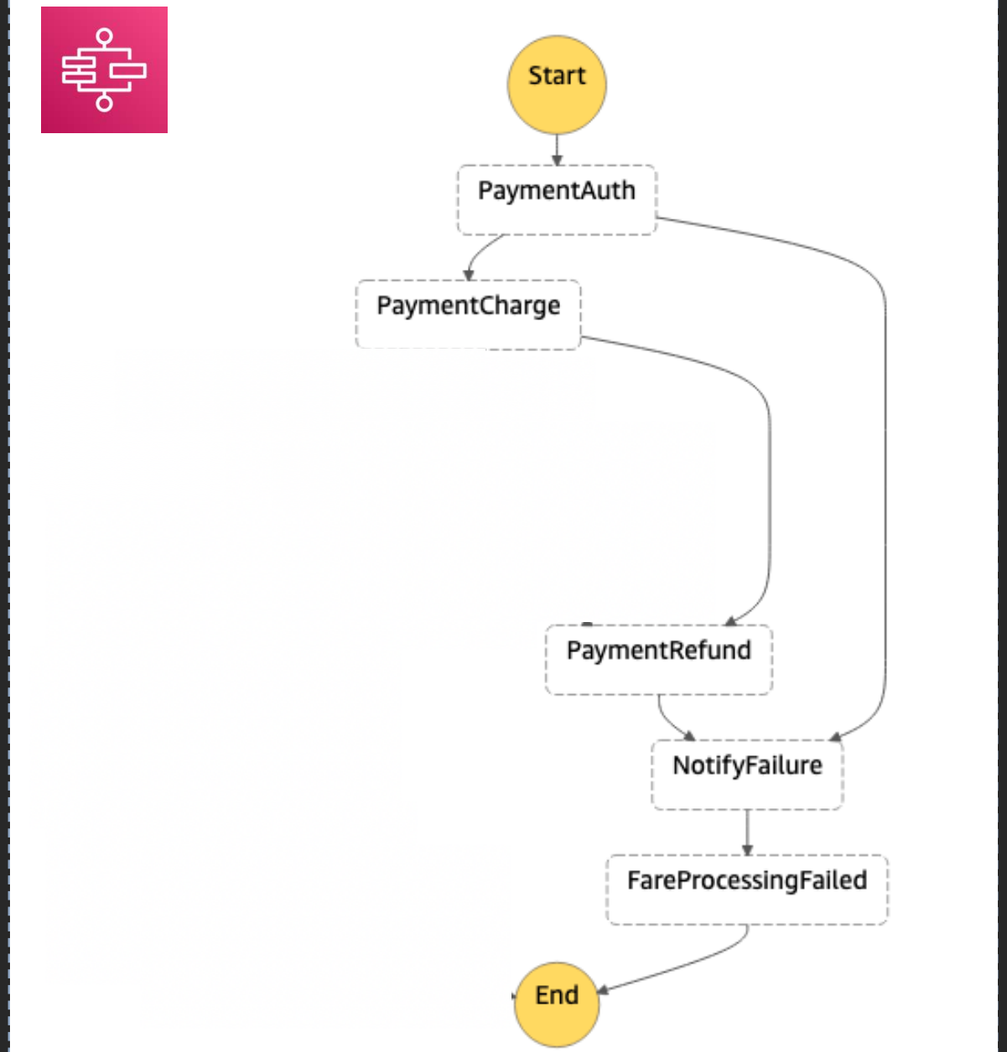
Use case: Fare collection

Saga orchestration



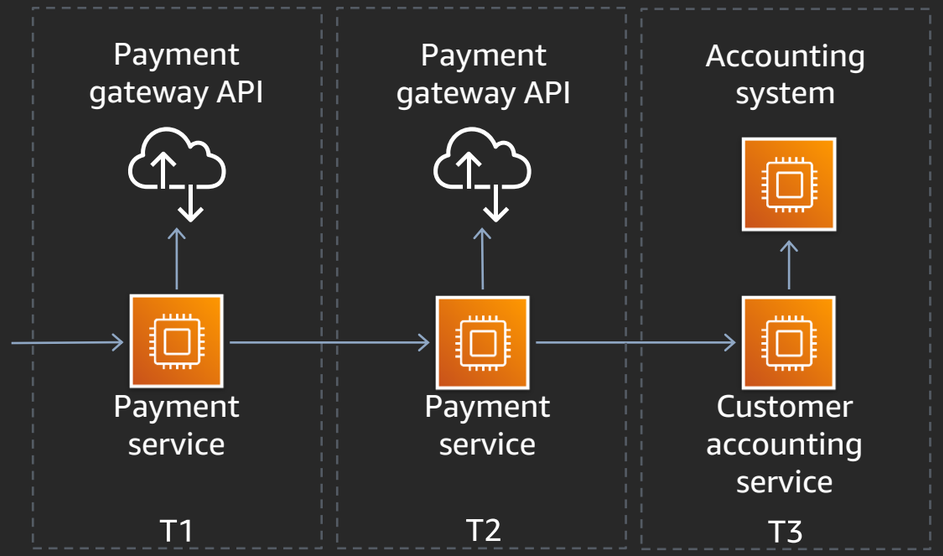
Discrete transactions

1. Credit card preauthorization
2. Charge card using preauthorization code



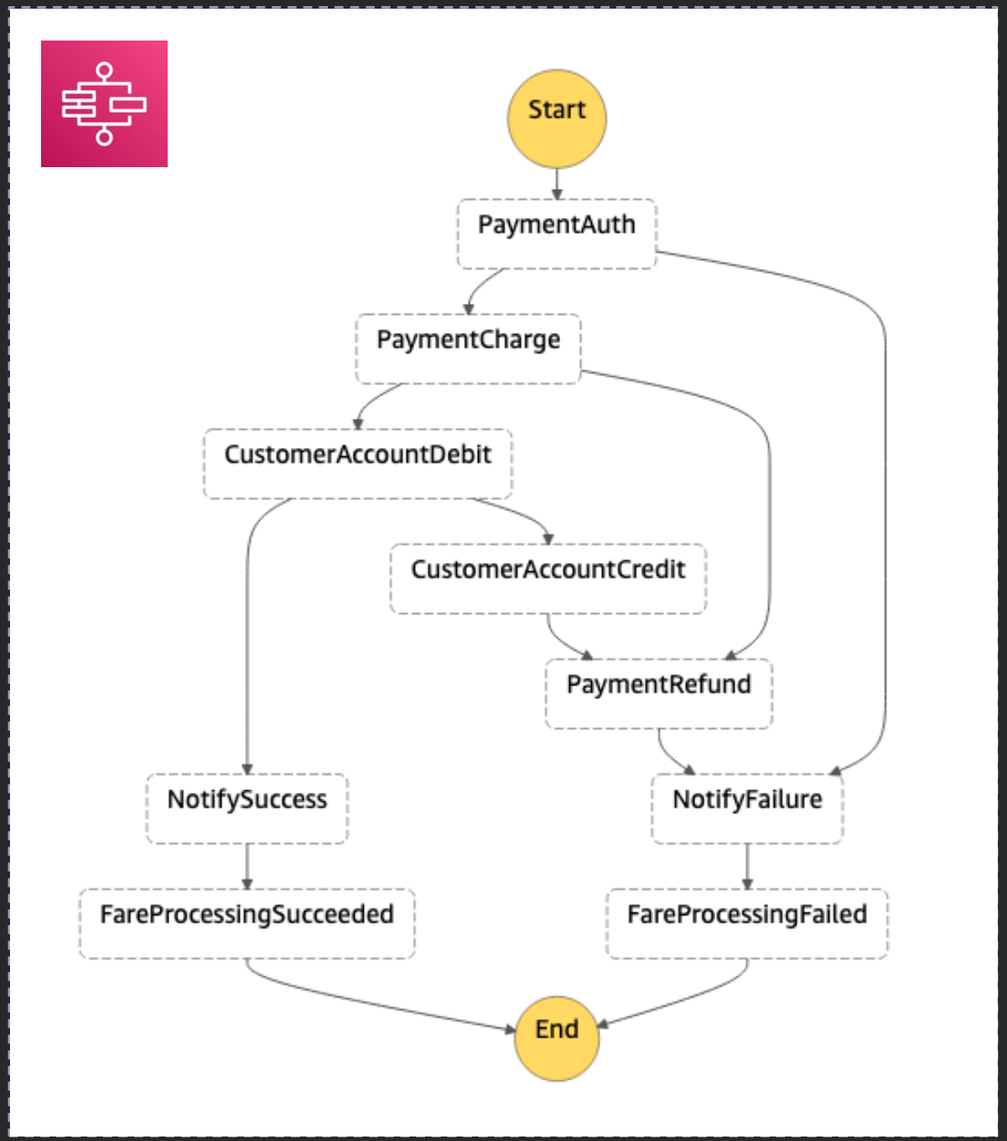
Use case: Fare collection

Saga orchestration



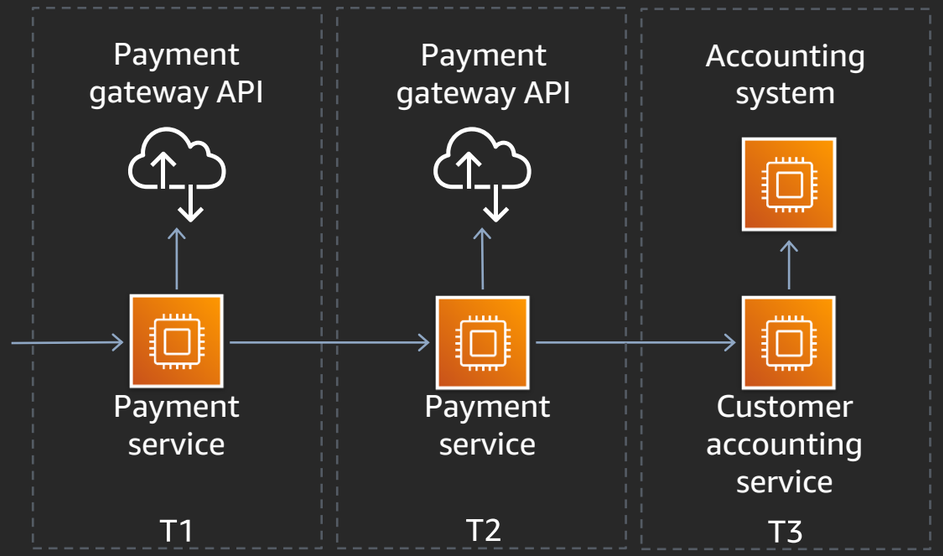
Discrete transactions

1. Credit card preauthorization
2. Charge card using preauthorization code
3. Update customer account



Use case: Fare collection

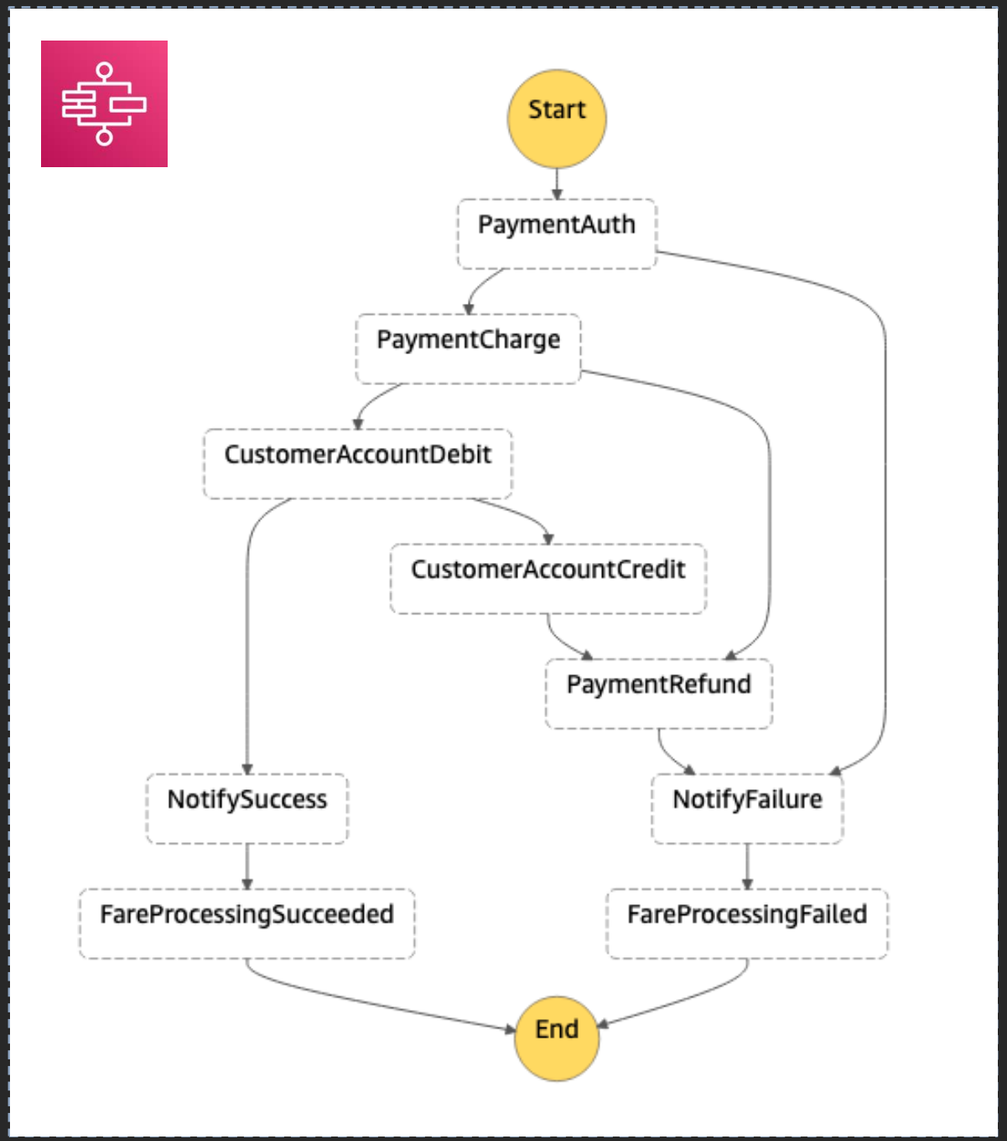
Saga orchestration



Discrete transactions

1. Credit card preauthorization
2. Charge card using preauthorization code
3. Update customer account

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



Resources and call to action

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

Resources and call to action

AWS blogs and other content about application integration

https://bit.ly/aws-msgn

Resources and call to action

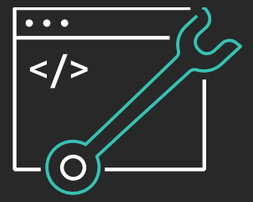
AWS blogs and other content about application integration

https://bit.ly/aws-msgn

Keep in mind – loose coupling is better than lousy coupling

Learn to build modern applications on AWS

Resources created by the experts at AWS to help you build and validate developer skills



Enable rapid innovation by developing your skills in designing, building, and managing modern applications

Learn to modernize your applications with free digital training and classroom offerings, including Architecting on AWS, Developing on AWS, and DevOps Engineering on AWS

Validate expertise with the AWS Certified DevOps – Professional or AWS Certified Developer – Associate exams

Visit the developer learning path at aws.amazon.com/training/path-developing

Thank you!

Anshul Sharma

anshulx@amazon.com
@anshuldsharma

aws SUMMIT ONLINE

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