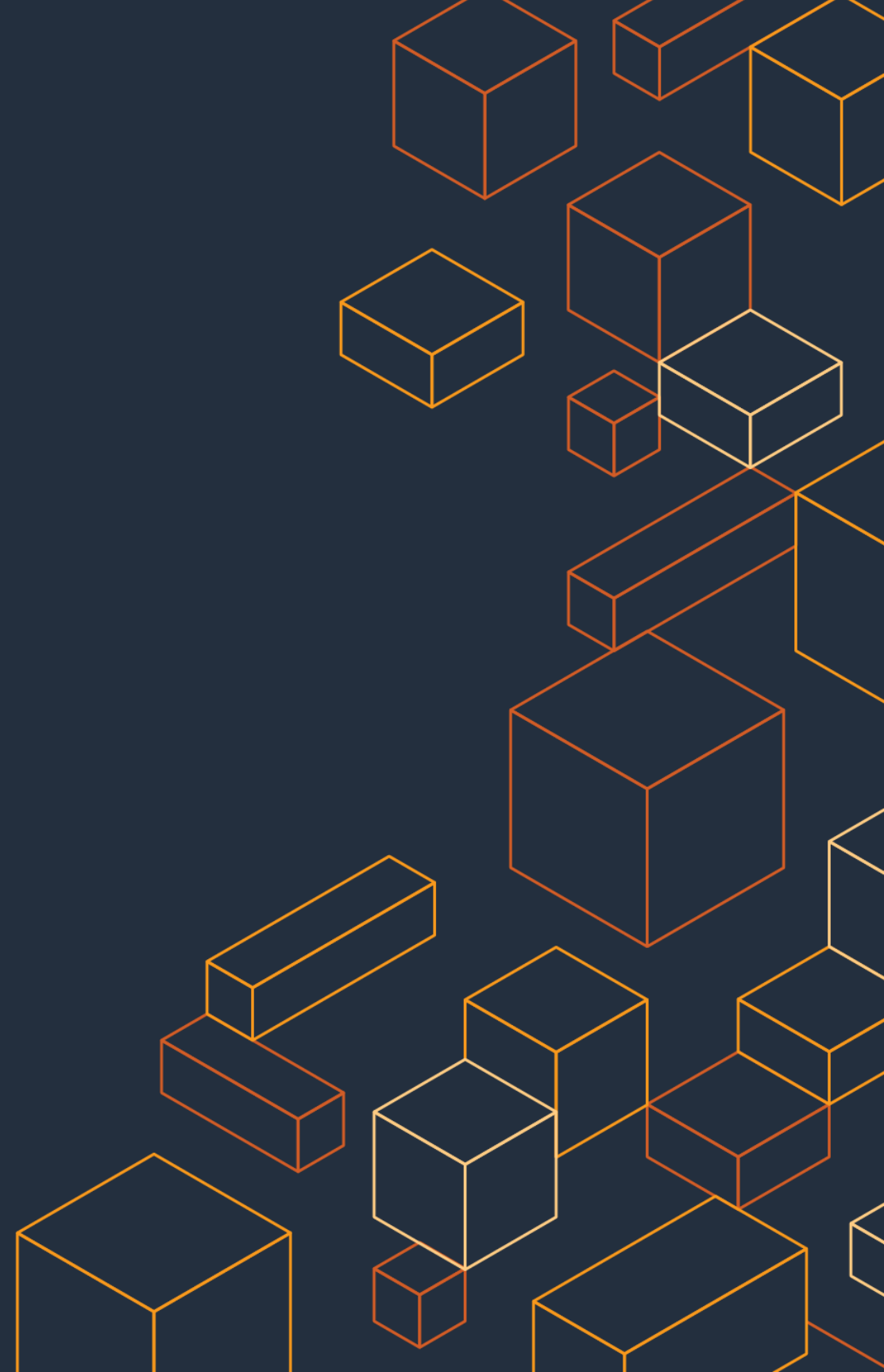




Serverless on AWS

Immersion Day

Vamsi Vikash Ankam
2021



Event-Driven Architectures



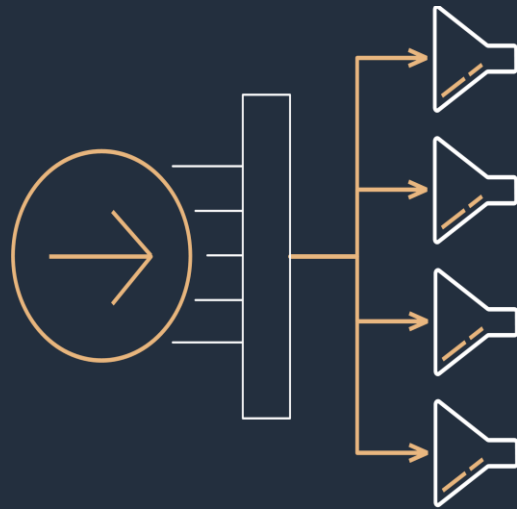
event

[i-'vent] noun

A thing that happens,
particularly one of
importance.

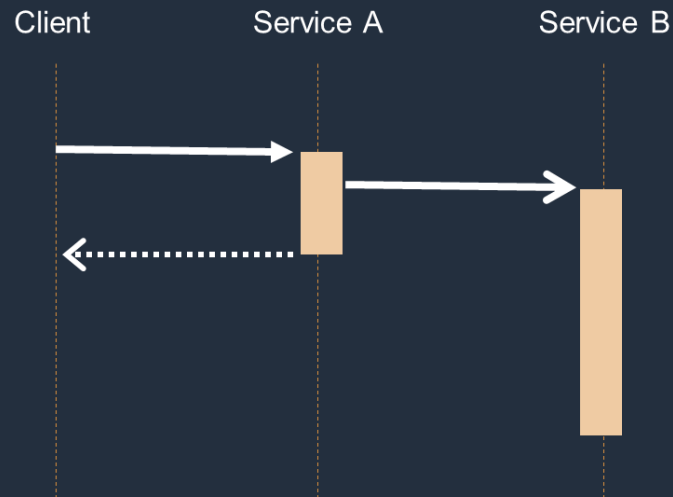
Event-driven architectures drive reliability and scalability

Event Routers



Abstract producers and consumers from each other

Asynchronous Events



Improve responsiveness and reduce dependencies

Event Stores

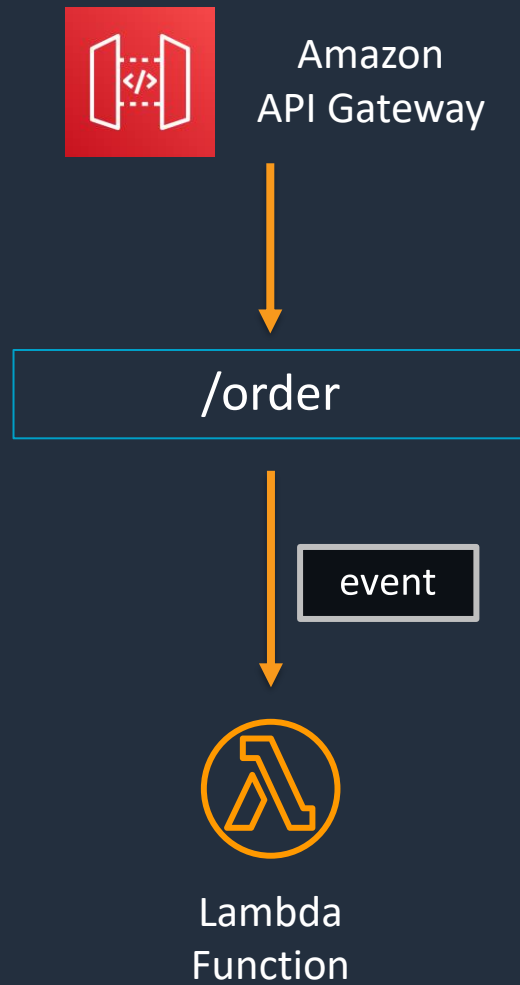


Buffer messages until services are available to process

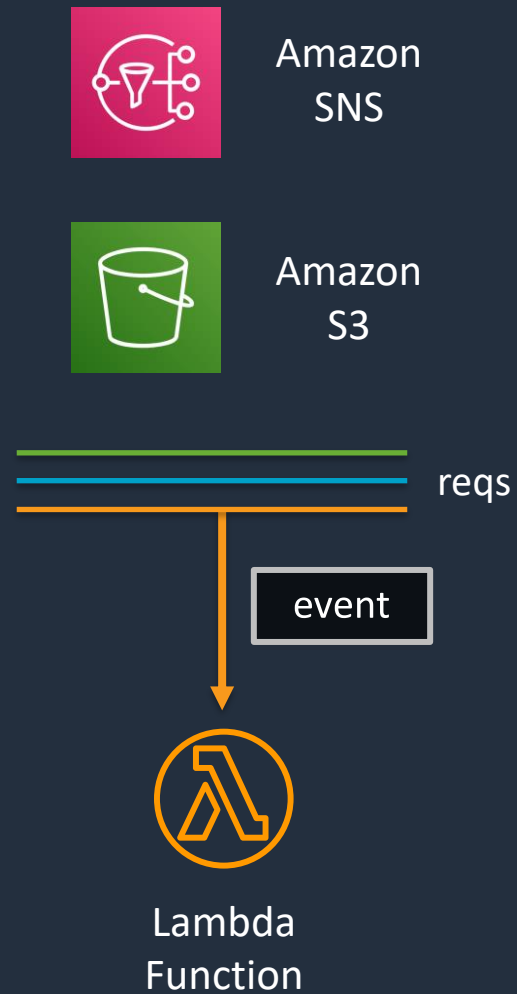
Lambda can be invoked via three different methods

All methods deliver an event payload

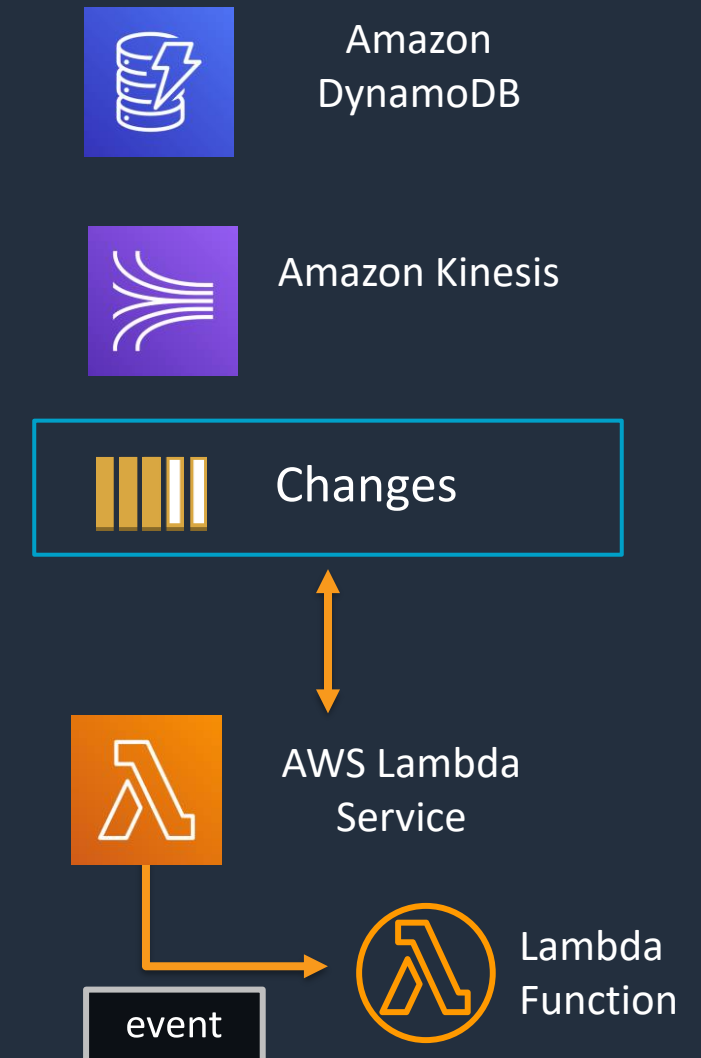
Synchronous



Asynchronous

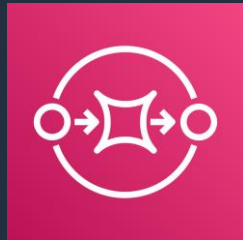


Poll-Based



Events enable interaction between services

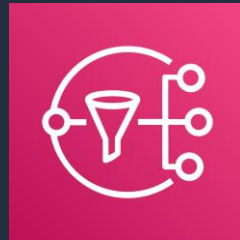
Managed services provide routing, storage, and distribution of events



Amazon SQS

Messaging

Durable and scalable
Fully managed
Comprehensive security



Amazon SNS

Eventing

Performance at scale
Fully managed
Enterprise-ready



EventBridge

Choreography

Event filtering
Managed & scalable
SaaS integration



Step Functions

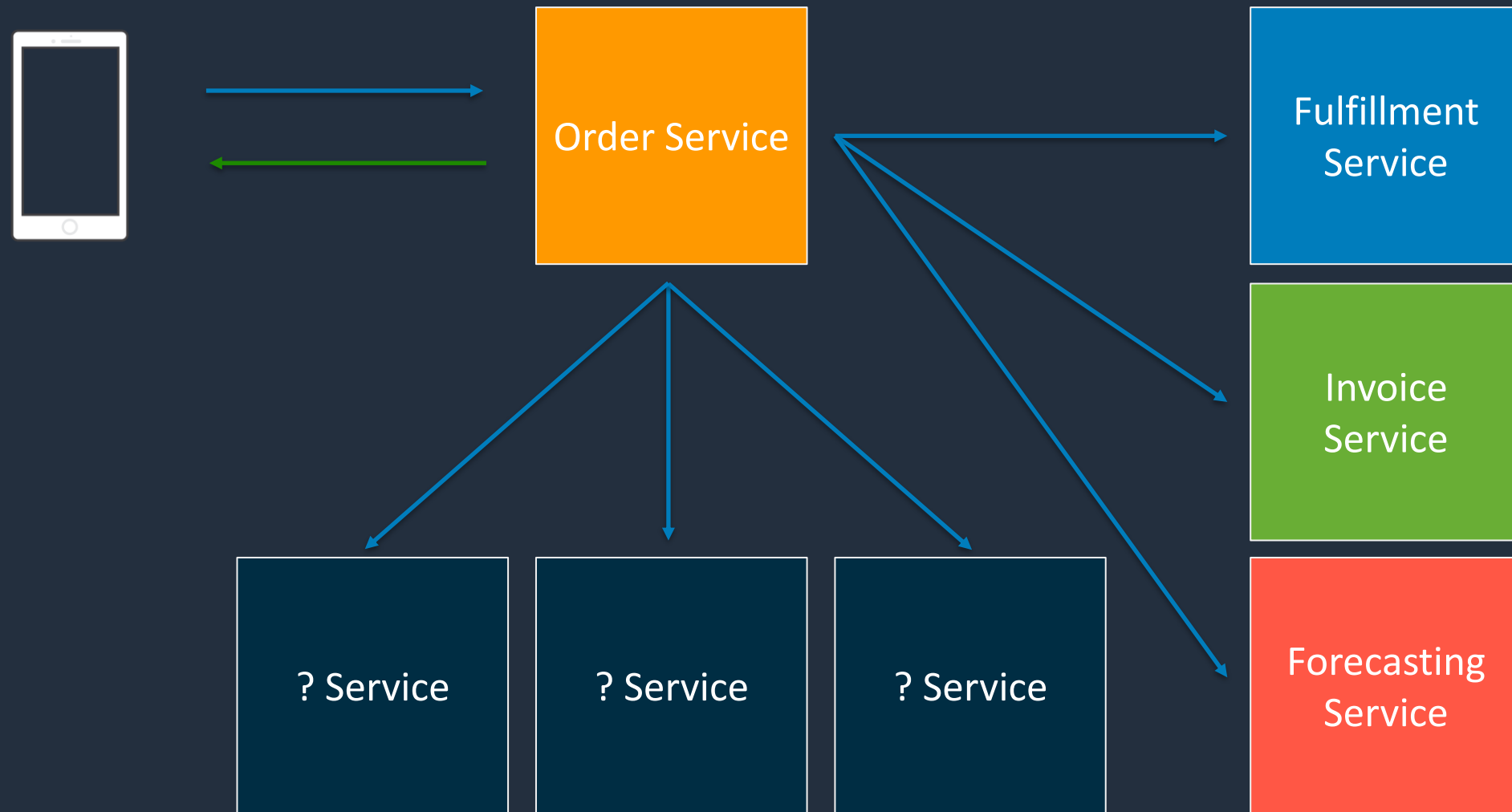
Orchestration

Sequencing
Parallel execution
State management

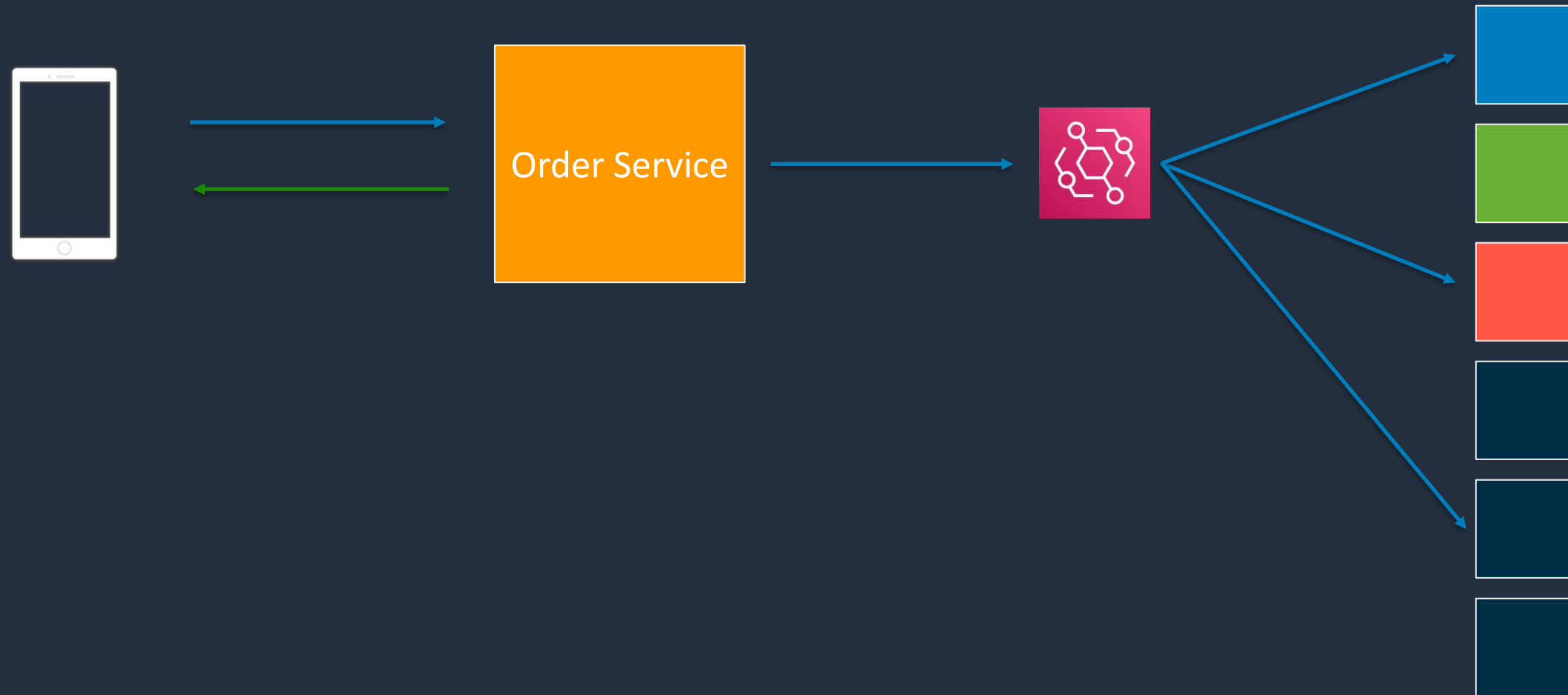
Let's launch an eCommerce site...



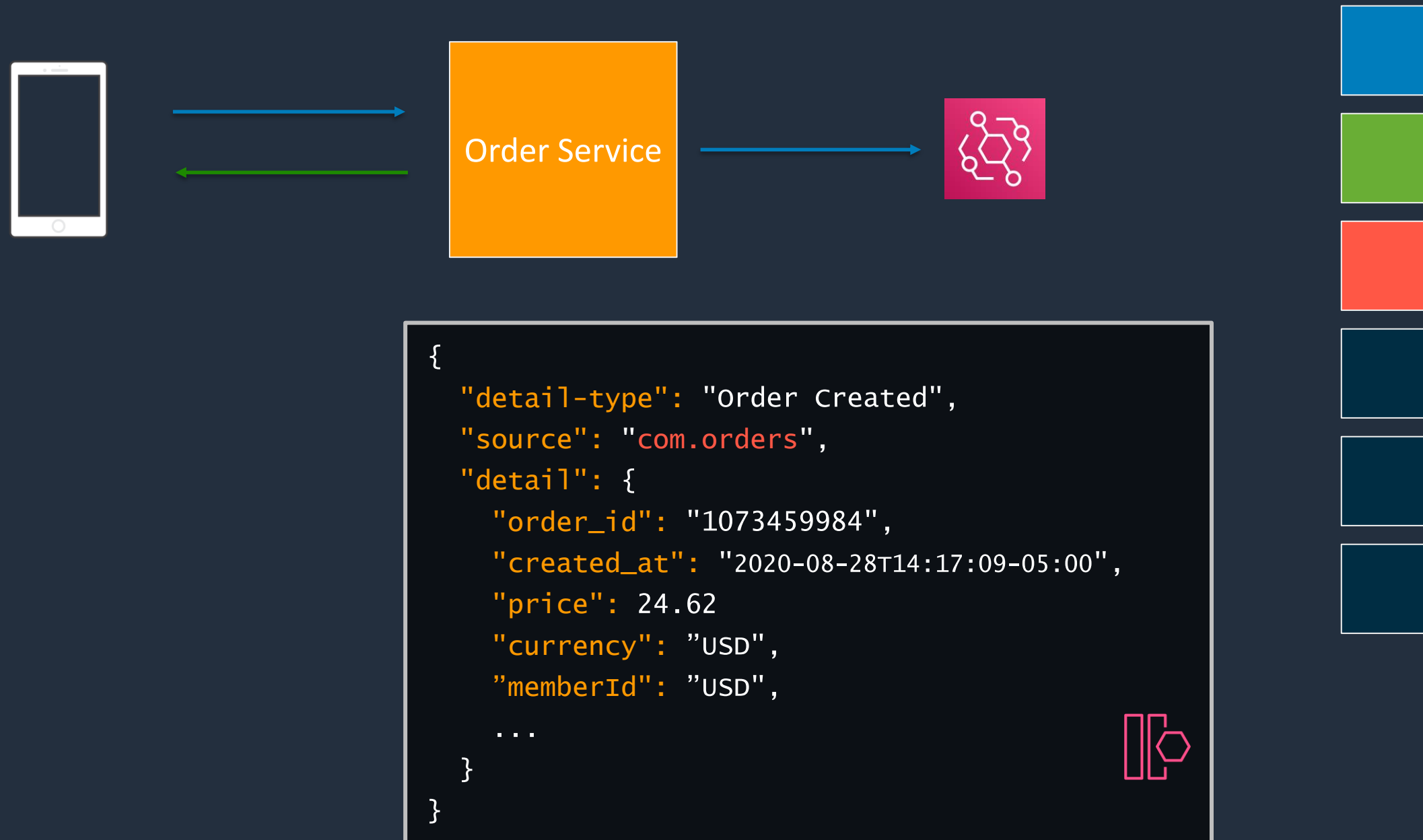
Growing our business...



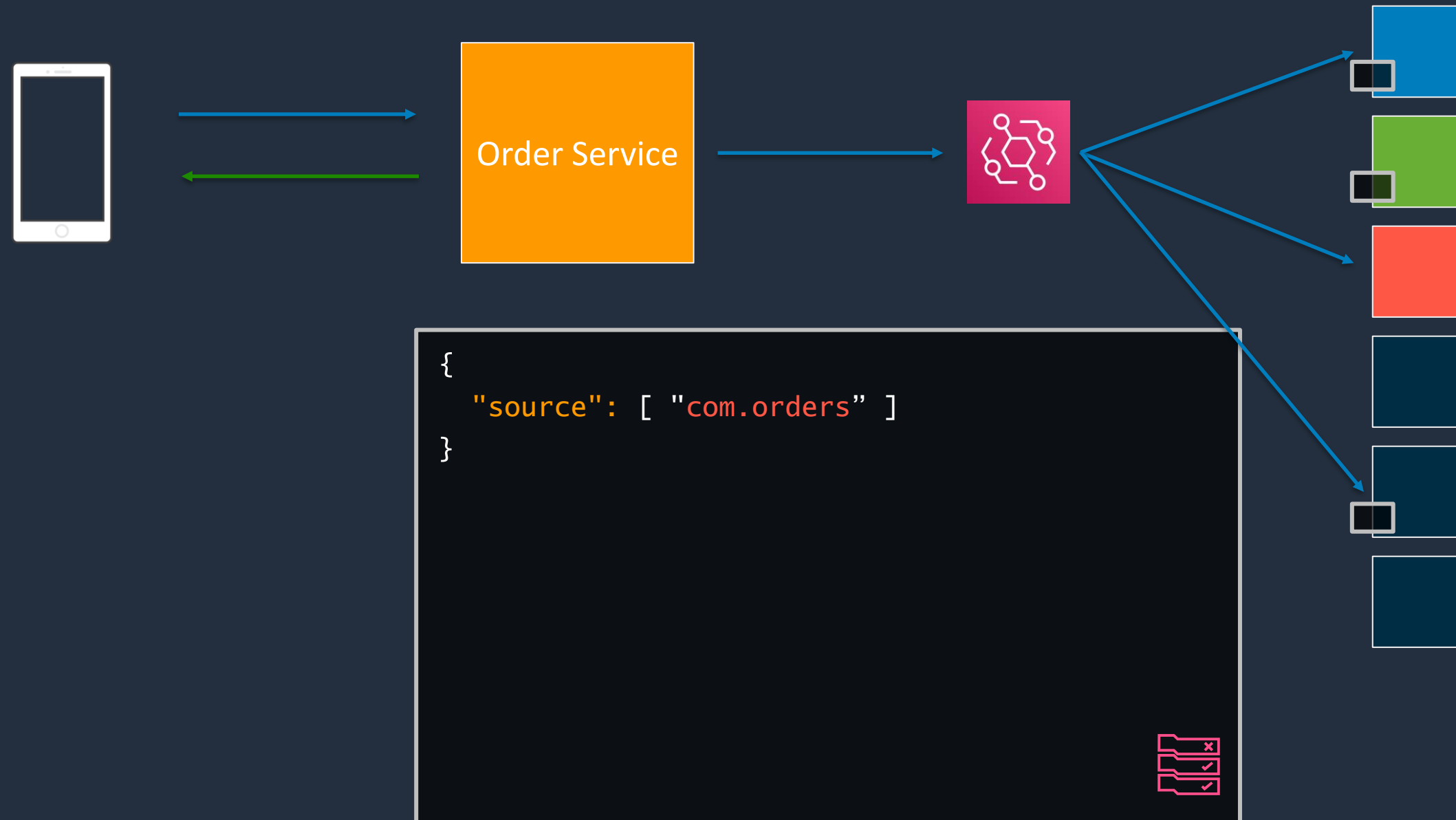
Building an event-driven architecture



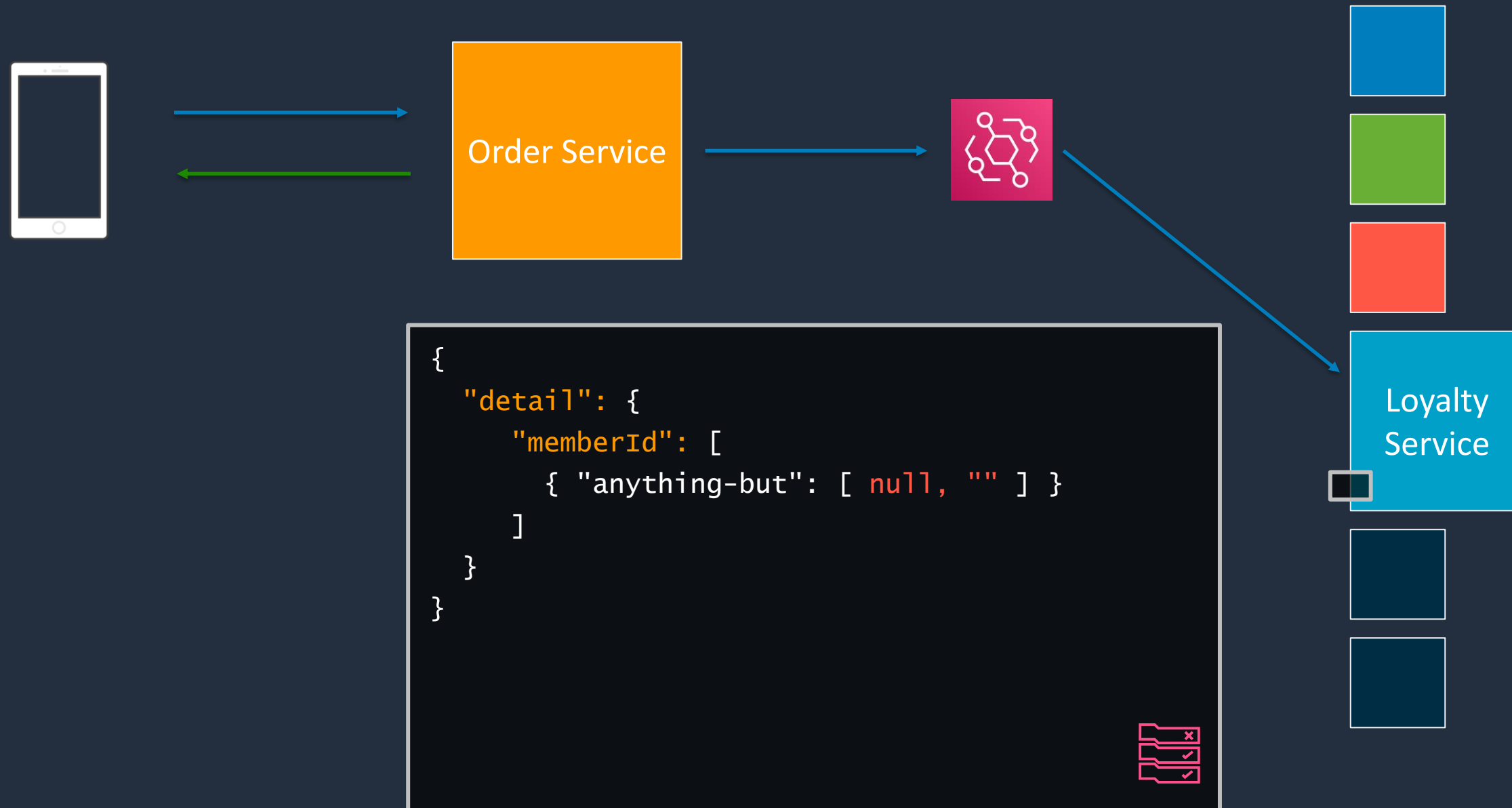
On an “Order Created” event



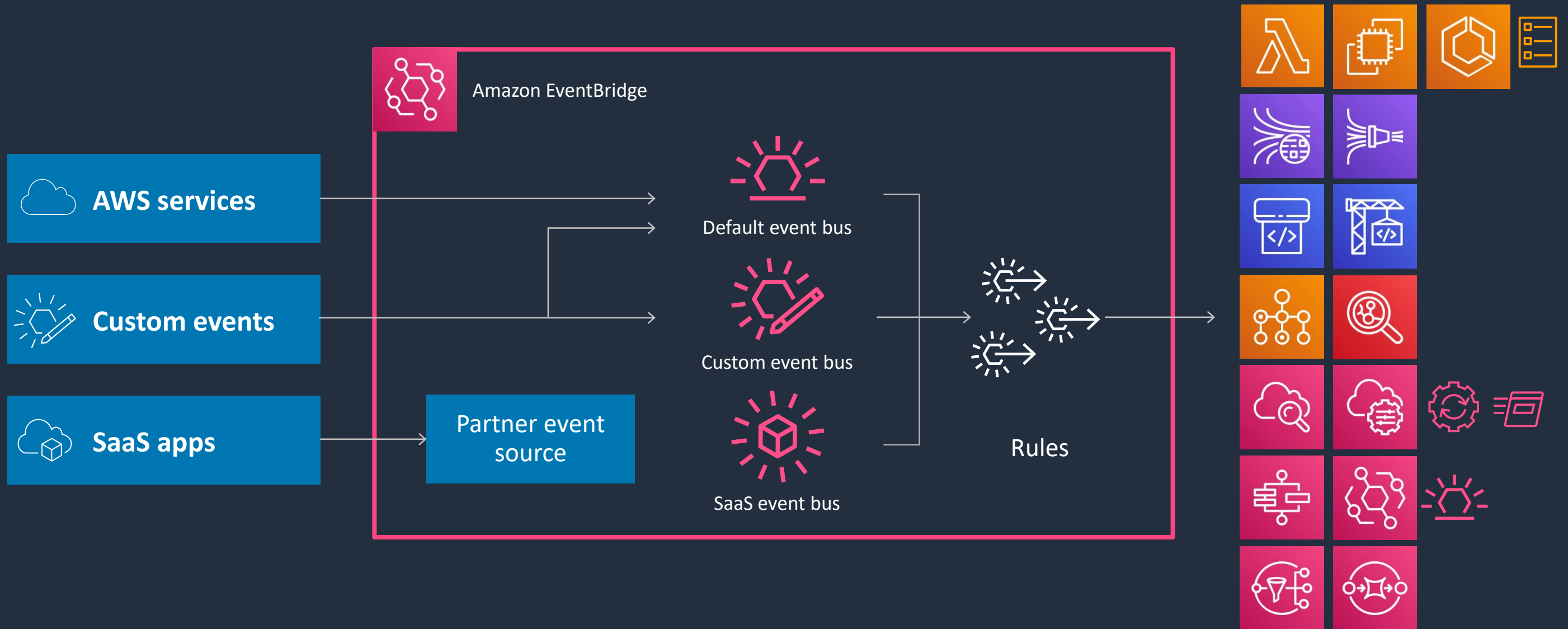
Route orders to appropriate services



Event-driven, serverless applications are flexible

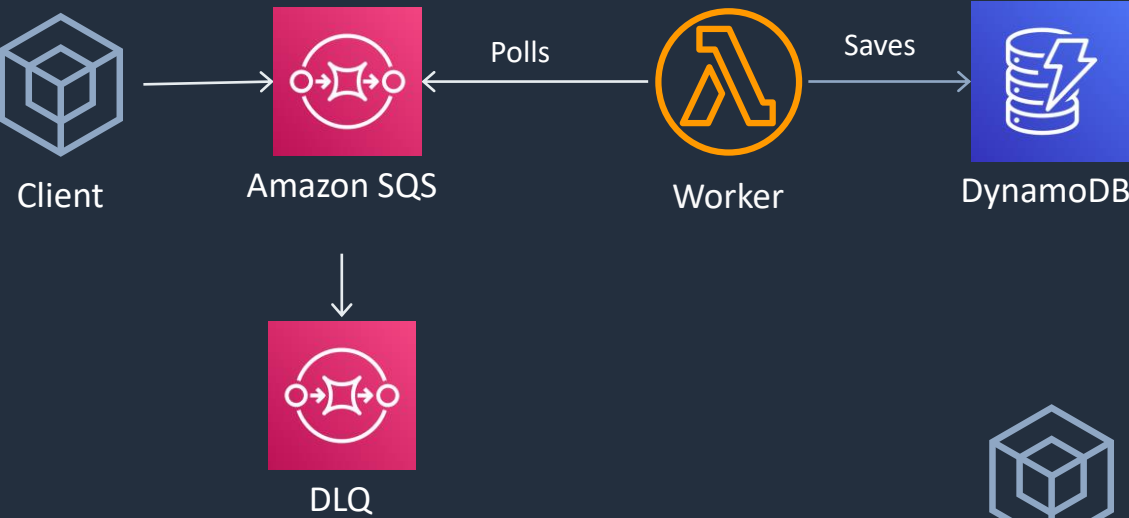


Amazon EventBridge architecture

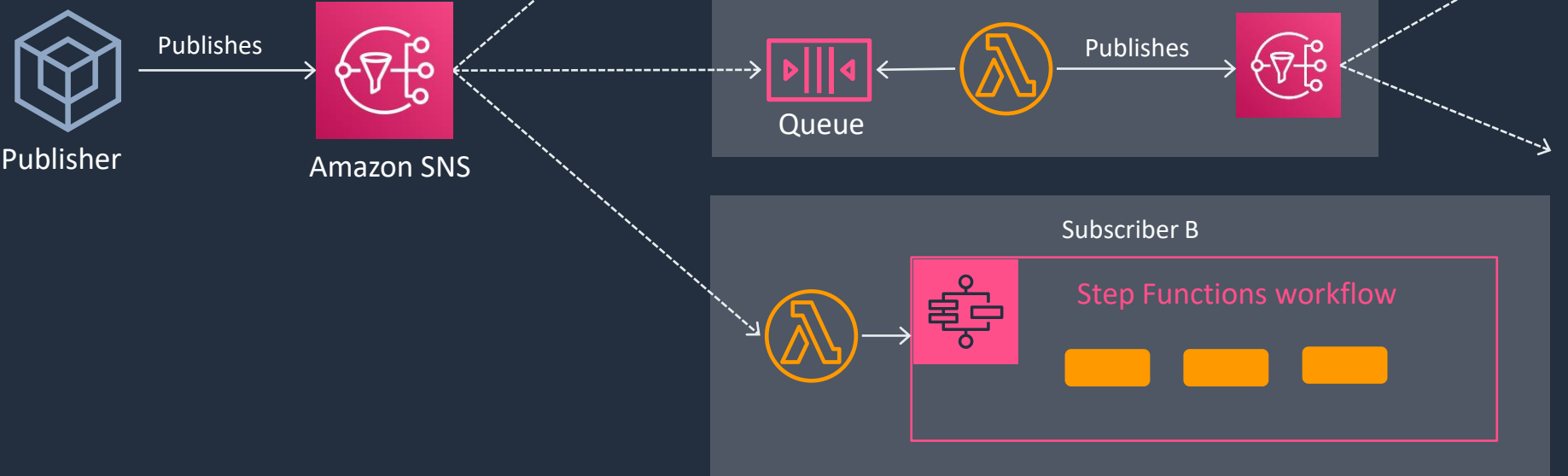


Event-driven architectural patterns

Decoupled messaging



Publish/subscribe

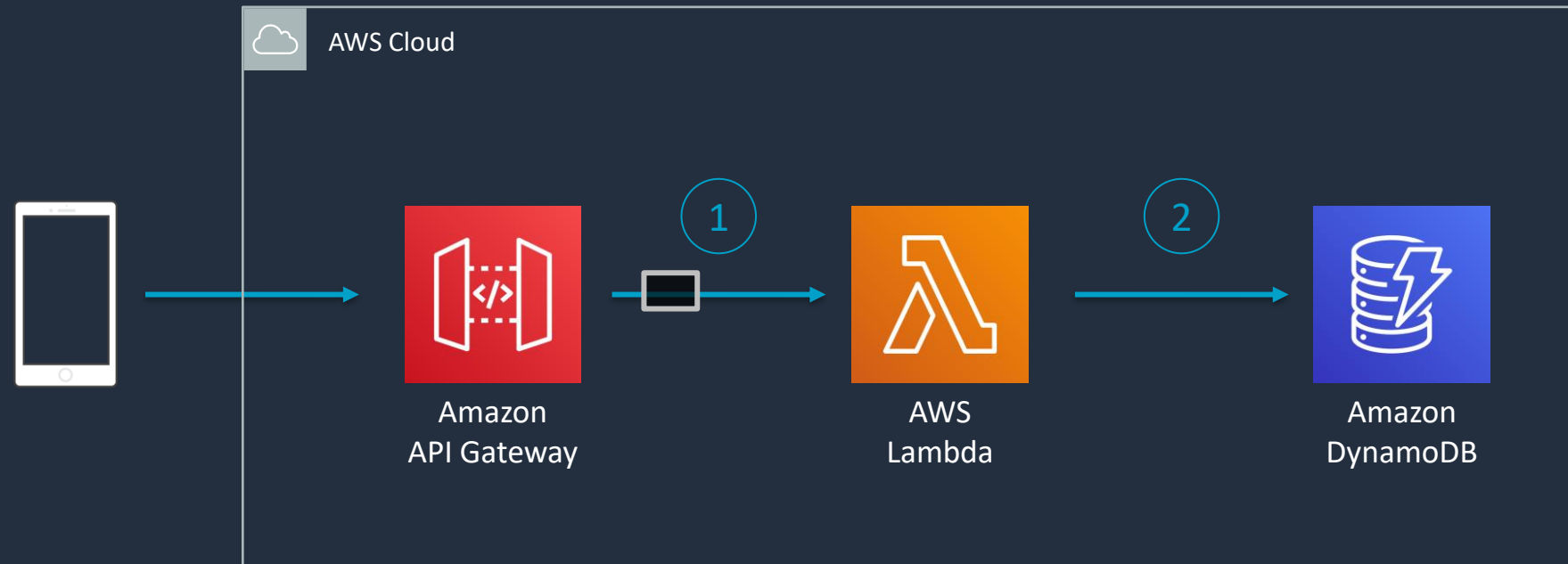


API-Driven Use Cases

Also **event** driven, synchronously processed

RESTful Microservices

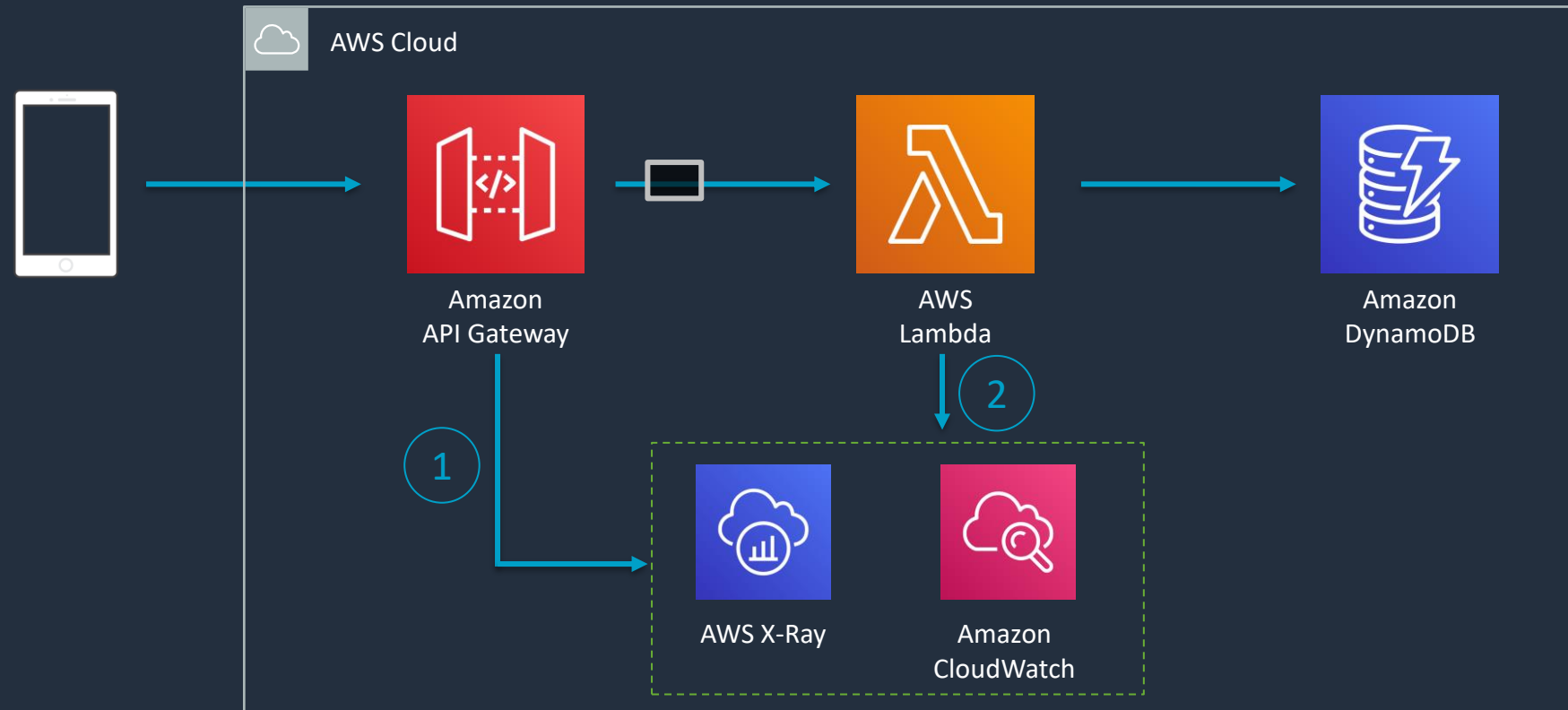
Highly-scalable microservices



1. API Gateway “translates” incoming HTTP request to event payload
2. Lambda reads / writes data from data store

RESTful Microservices with enhanced observability

Enable access logs, structured logging, and instrument code



1. Enable access logs and tracing

2. Instrument code and create metrics
asynchronously with CloudWatch
Embedded Metric Format

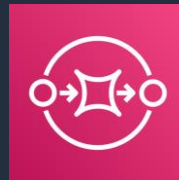
Choreography

Streams, Topics, and Queues

AWS Messaging and Event Services

Event Store

Queues

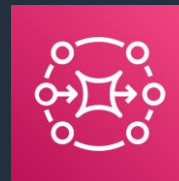


Amazon SQS

Streams



Amazon Kinesis



Amazon MQ



Amazon MSK

Event Router

Topics



Amazon SNS

Bus



Amazon EventBridge



Amazon MQ

Cloud Native

Managed Open
Source

Comparing messaging and event services



Scale / Concurrency
Controls



Durability



Persistence



Consumption
Models

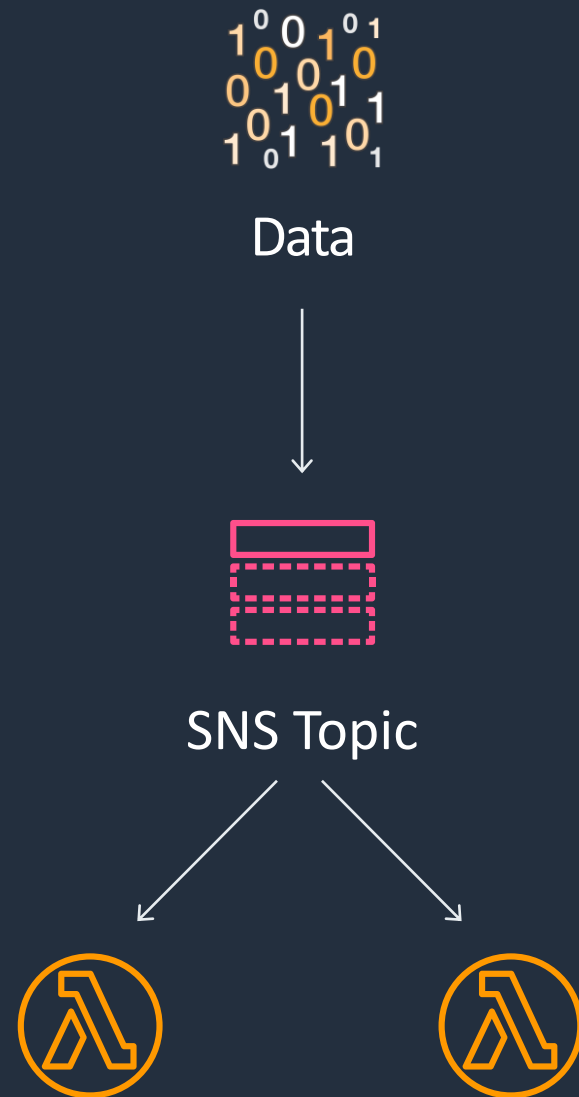


Retries



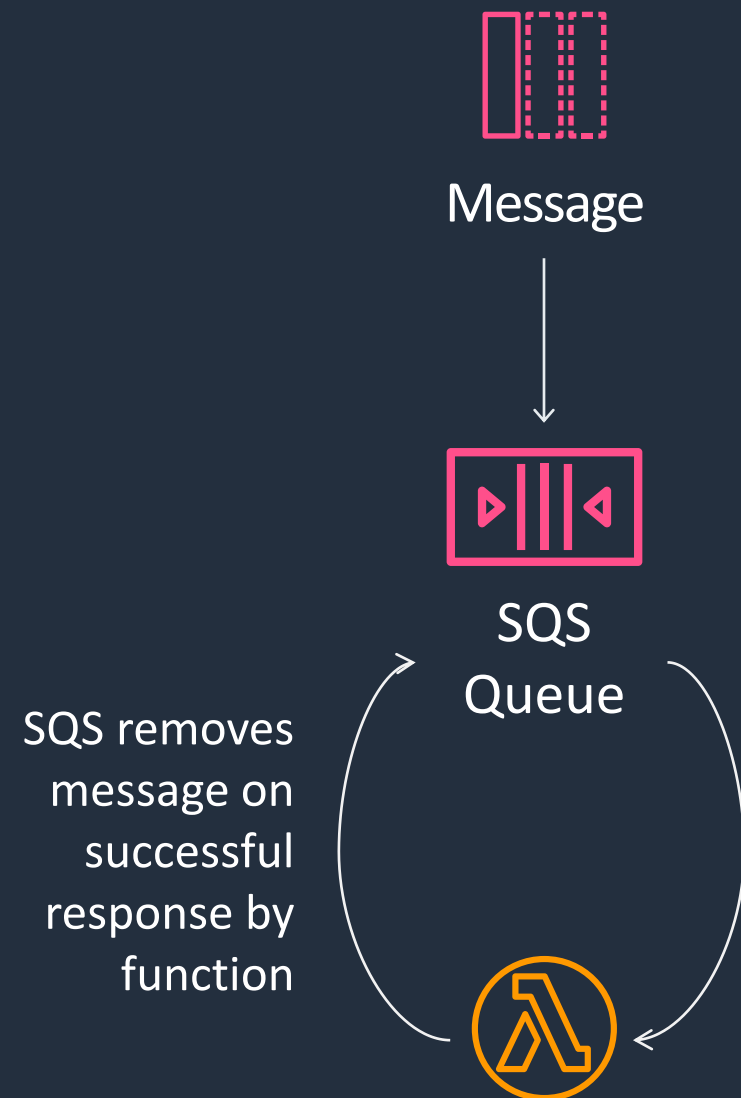
Pricing

Amazon Simple Notification Service (SNS)



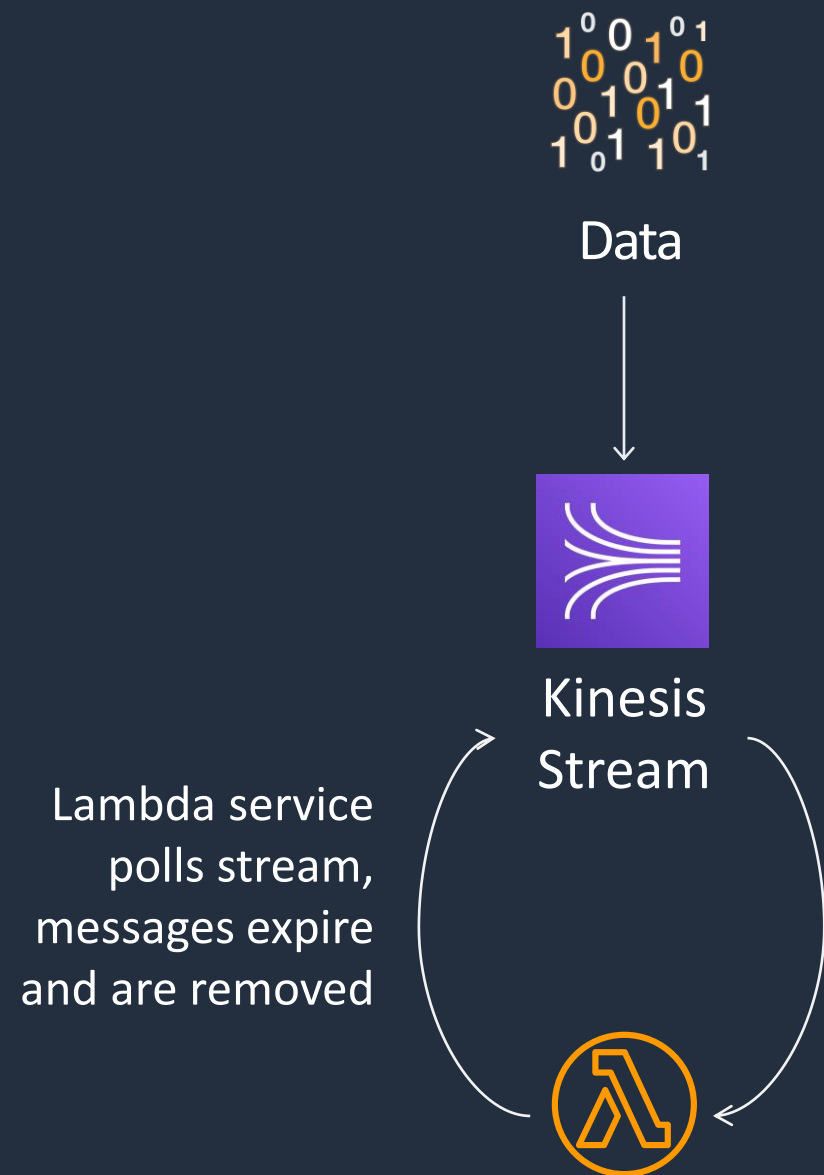
- Publish / subscribe messaging
- High throughput, highly reliable message delivery
- Messages are published to a Topic with multiple subscribers – “fan out”
- Messages can be filtered and only sent to certain subscribers
- **Asynchronous** Lambda invocation

Amazon Simple Queue Service (SQS)



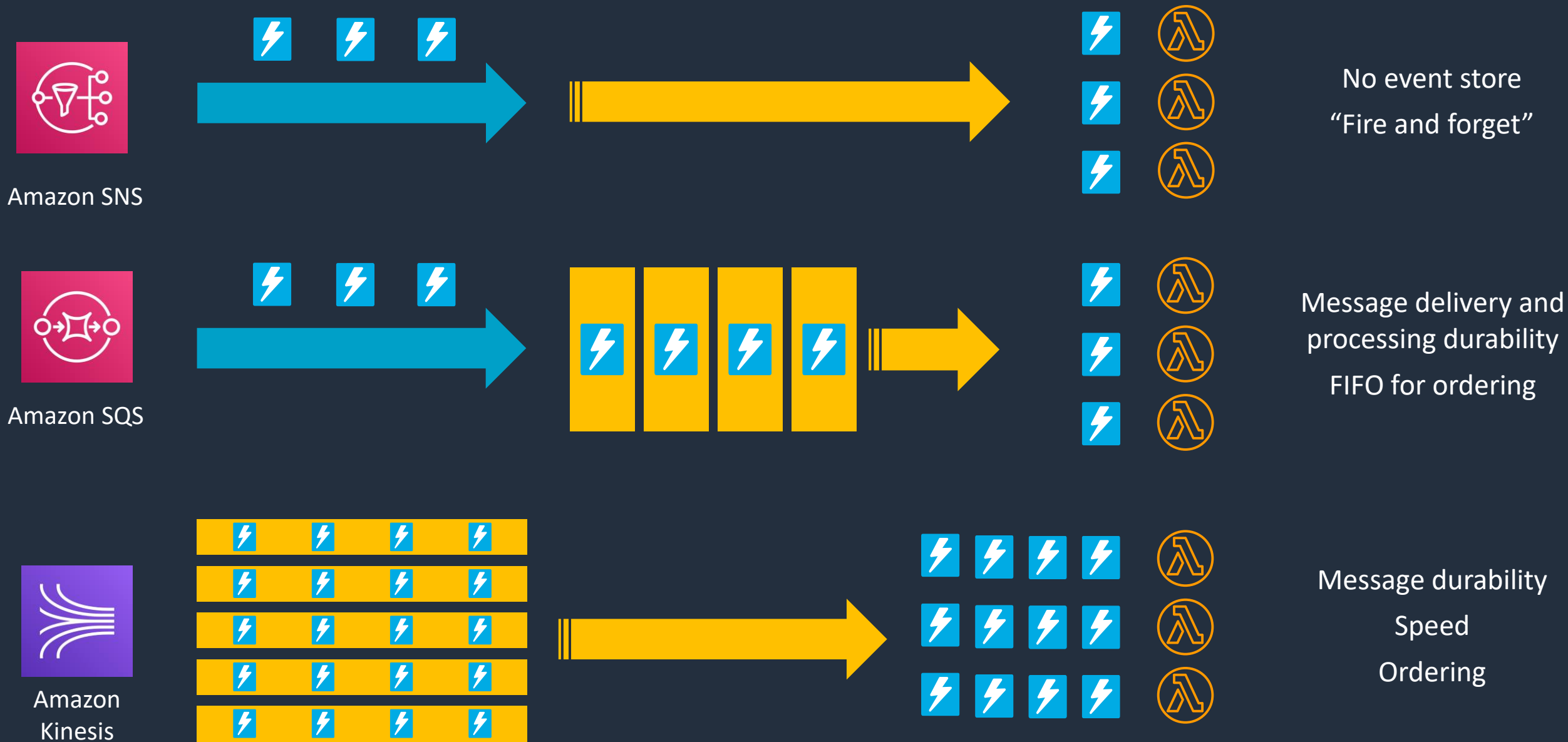
- Any volume of messages
- Messages processed in batches
- At least once delivery
- Visibility timeout allows handling failures
- **Synchronous** Lambda invocation
- Service **long-polls** queues

Amazon Kinesis Data Streams

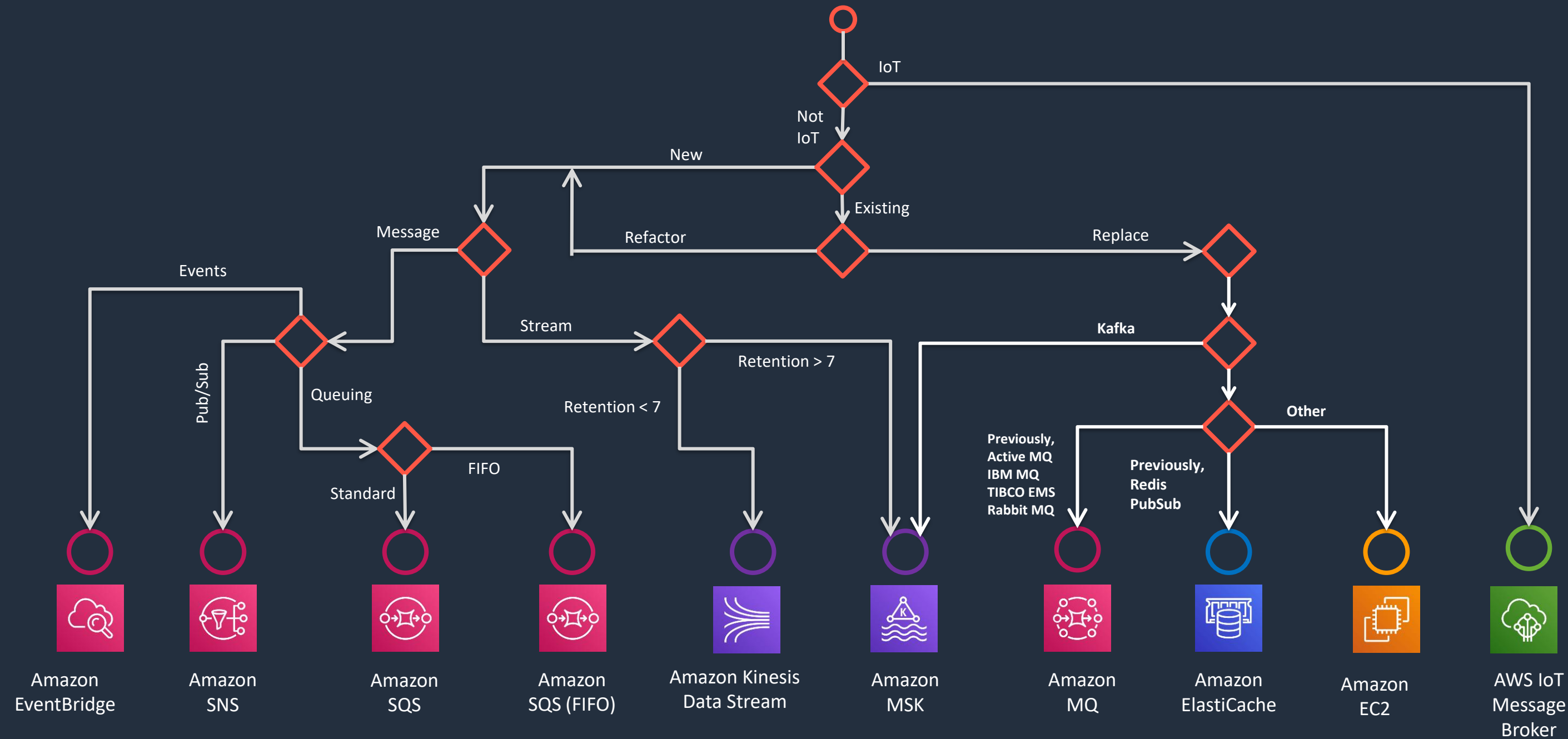


- Real-time data streams for analytics and machine learning
- At least once delivery
- Function receives batch of data and potentially batches of batches
- **Synchronous** Lambda invocation
- Service **long-polls** queues

Lambda function concurrency across models



A decision tree?



Event-Driven Use Cases

Processing file uploads

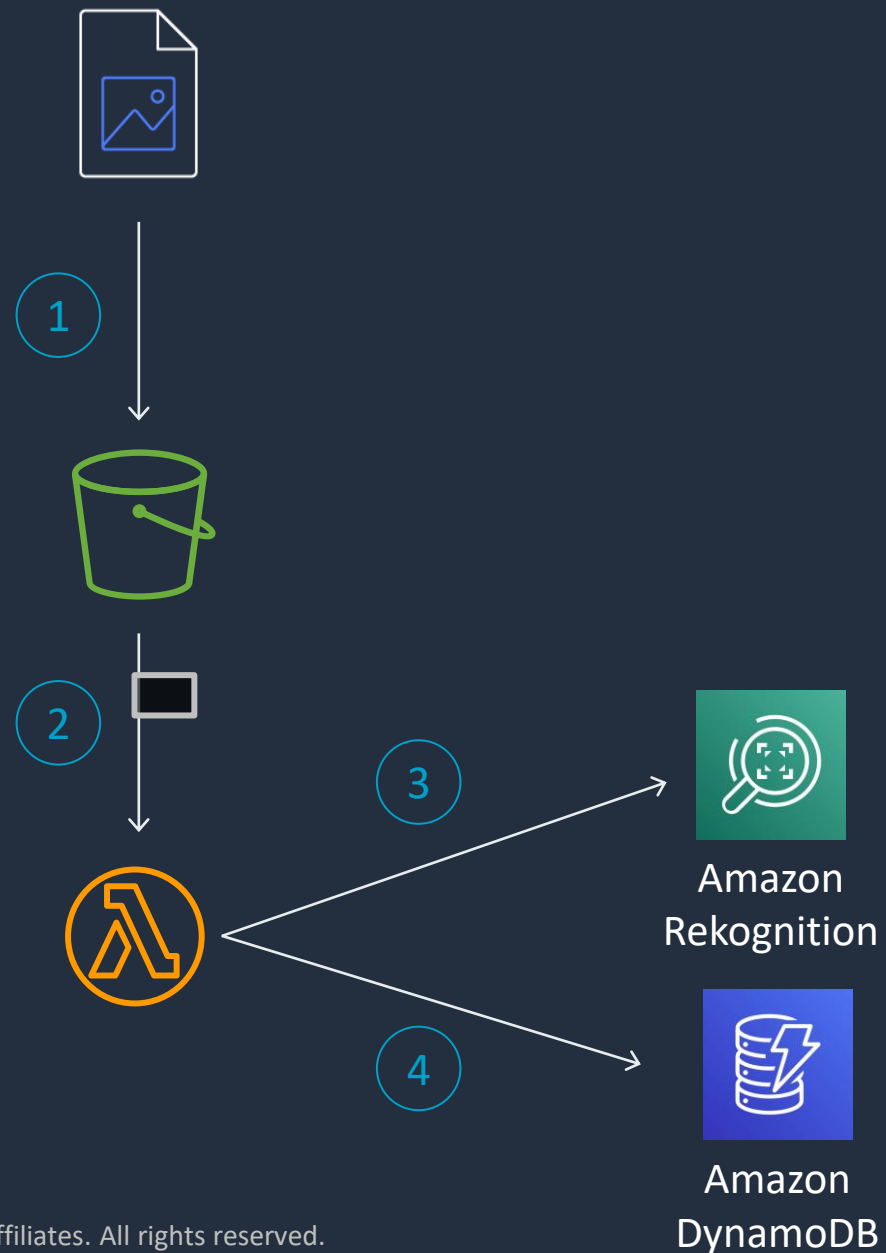
Resize photo, extract text, translate, etc.



1. Object uploaded to Amazon S3 Bucket
2. **Asynchronous** invoke of Lambda function, event payload includes:
 - Bucket name
 - Object key

Processing file uploads, quickly add new functionality

Add image analysis and metadata storage



3. Analyze photo with Amazon Rekognition

4. Store image details and results of analysis

Streaming data ingestion and storage

Consume, process, and store data

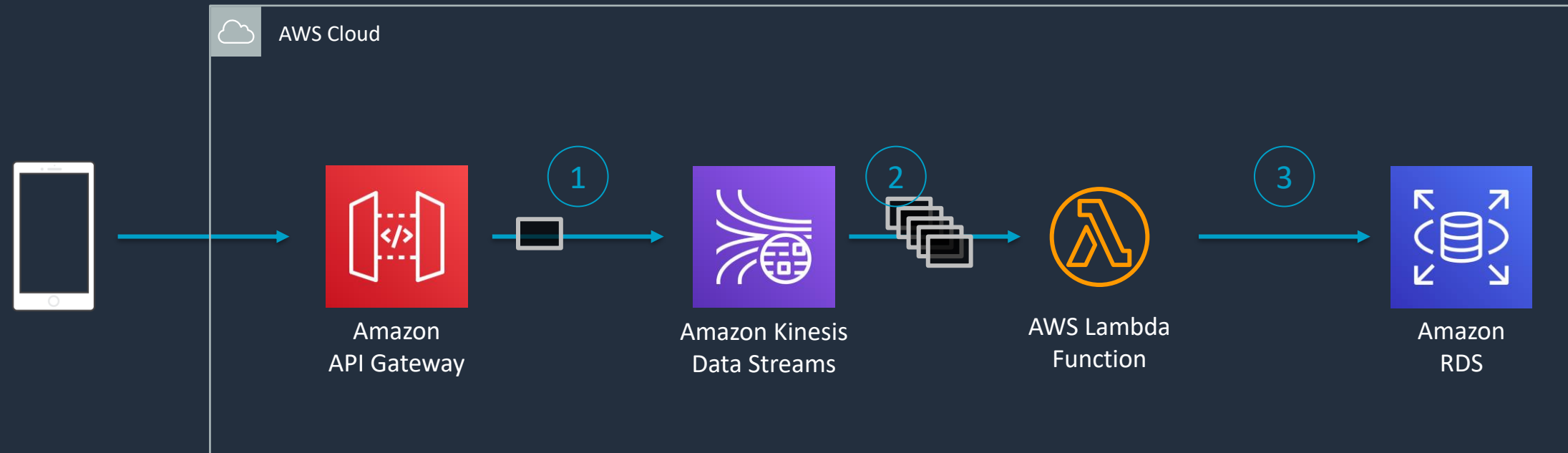


1. Lambda service polls Kinesis Data Stream for messages
2. Function is **synchronously** invoked with **batches** of messages

3. Function processes and/or pushes data to downstream data stores

“Semi-Serverless” Webhook

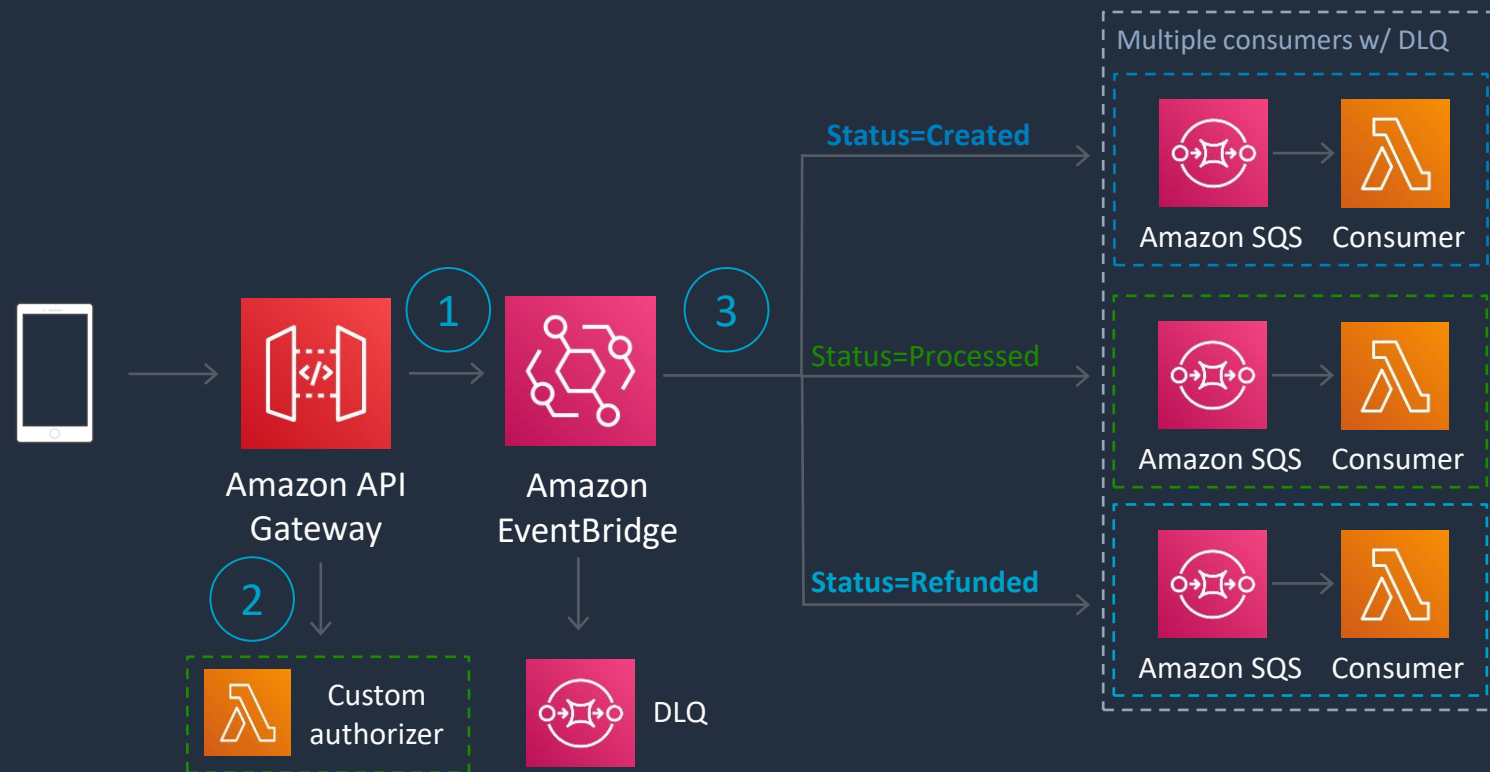
Scalable, resilient. Buffer downstream concurrency.



1. “Storage-first” pattern: API Gateway writes directly to Kinesis Data Stream
2. Kinesis as a buffer to limit concurrency downstream
3. Perform transaction(s) on batch

Fan out

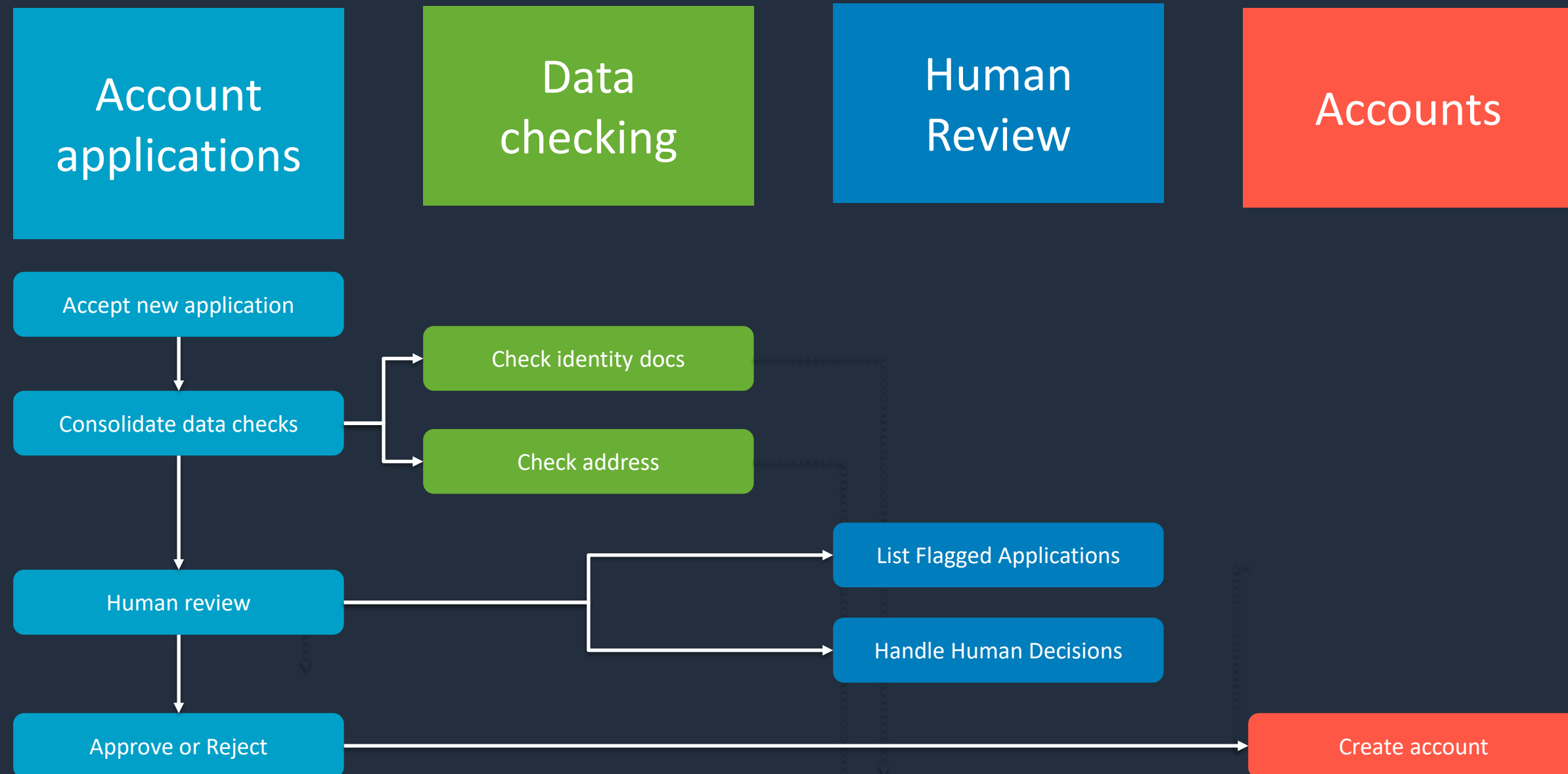
Push updates to multiple subscribers



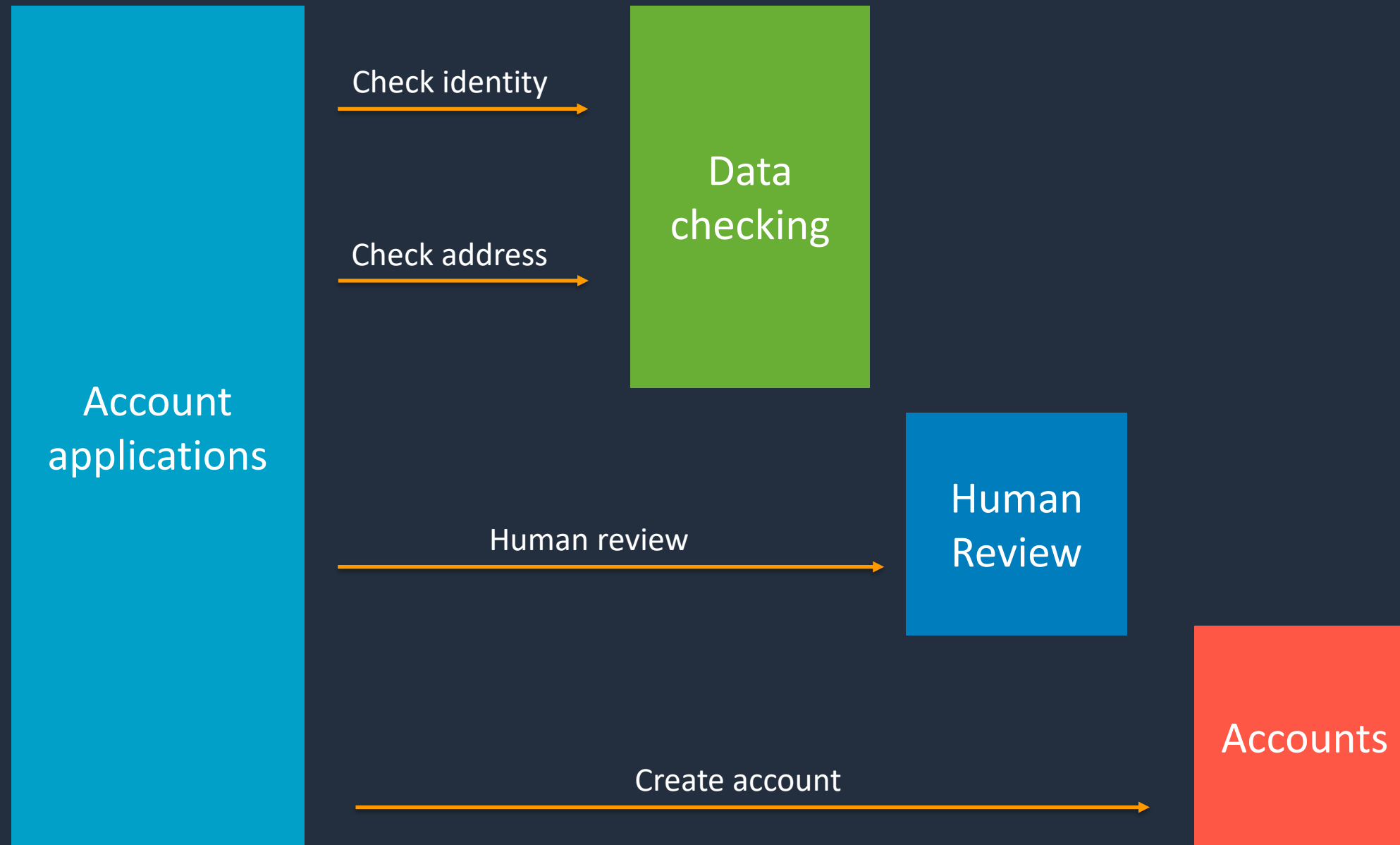
1. “Storage first”: integrate API Gateway directly to EventBridge
2. Enforce authorization
3. Use routing for efficient processing

Orchestration

Simplified banking system: Processing a new account requires coordination

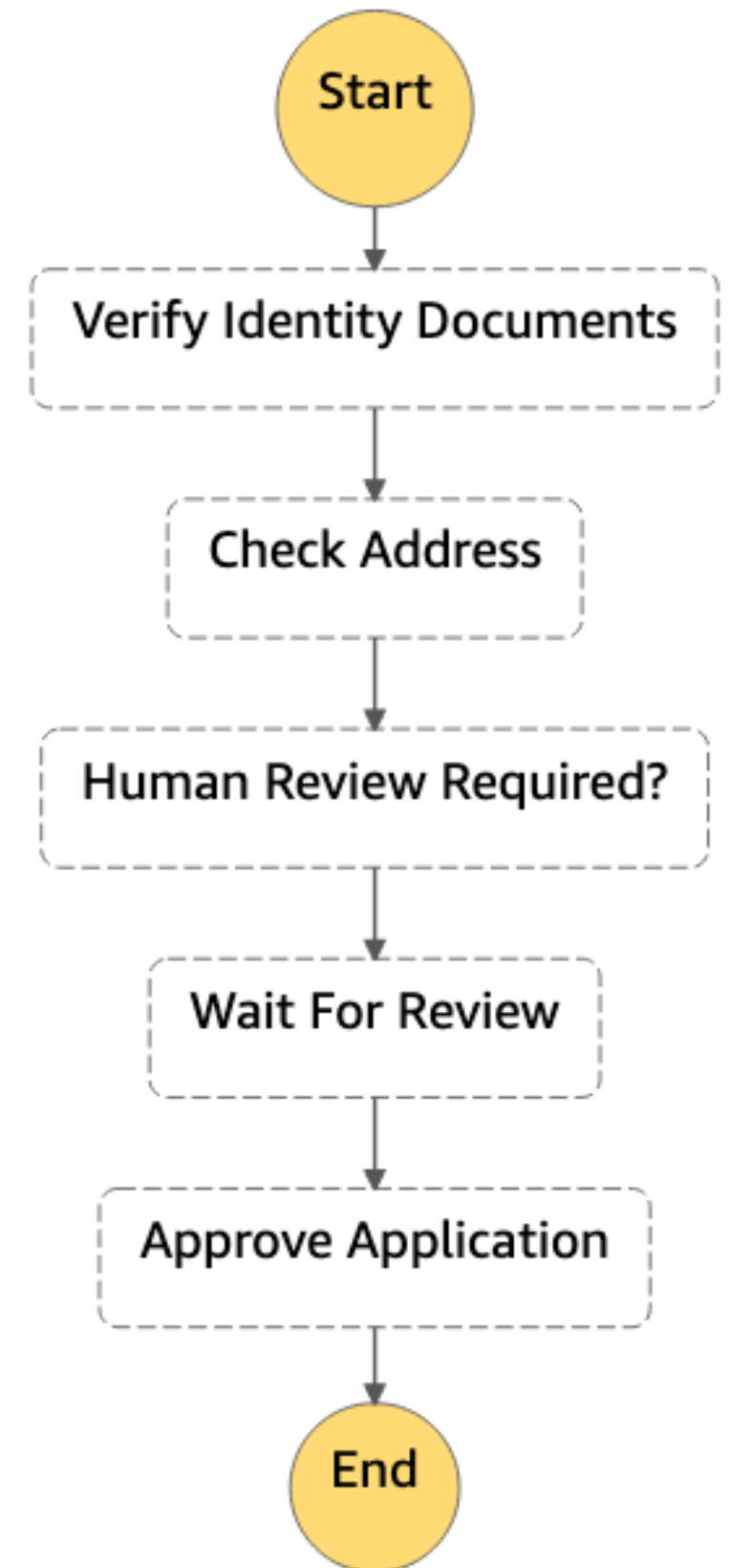


Orchestration: one process manages workflow state and calls appropriate services in turn



Modeling as a State Machine

- Describes a collection of computational steps split into discrete states
- Has one starting state and always one active state (while executing)
- The active state receives input, takes some action, and generates output
- Transitions between states are based on state outputs and rules that we define



AWS Step Functions

Serverless state machines

- Resilient workflow automation
- Built-in error handling
- Powerful AWS service integrations
- Integrate with your own services
- Auditable history, visual monitoring
- Defined with JSON

Visual workflow

Code

Step details

Name

Automated Checks Choice

Status

✔ Succeeded

Resource

-

▶ Input

▶ Output

▶ Exception

Execution event history

ID	Type	Step	Resource
▶ 1	ExecutionStarted		-
▶ 2	ParallelStateEntered	Perform Automated Checks	-
▶ 3	ParallelStateStarted	Perform Automated Checks	-
▶ 4	TaskStateEntered	Check Identity	-
▶ 5	LambdaFunctionScheduled	Check Identity	Lambda CloudWatch logs
▶ 6	PassStateEntered	Check Fraud Model	-

Hands-on Workshop

Get hands-on with serverless messaging

- SNS
- EventBridge
- Filtering
- Asynchronous processing

aws

Search...

1. Getting Started

2. Event-Driven with EventBridge

2.1 First event bus and targets

2.2 Working with EventBridge Rules

2.3 Scheduling Expressions for Rules

3. Event-Driven with Lambda

4. Event-Driven with SNS

5. Workshop Clean Up

Privacy | Site Terms | © 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

event-driven-architecture.workshop.aws

Building Event-Driven Architectures on AWS > Event-Driven with EventBridge > Worki...

WORKING WITH EVENTBRIDGE RULES

Rules match incoming events and routes them to targets for processing. A single rule can route to multiple targets, all of which are processed in parallel. Rules aren't processed in a particular order. A rule can customize the JSON sent to the target, by passing only certain parts or by overwriting it with a constant. EventBridge supports 20+ AWS service targets!

In this module, you will walk through the steps to create an **Orders** EventBus rule to match an event with a **com.amazonaws.orders** source and to send the event to an **Amazon Kinesis Firehose** delivery stream, **OrdersDelivery Stream**, storing the events in an **Amazon S3 Bucket**. Afterwards, you will be challenged to create two additional rules without instruction. See the diagram below:

Info

The EventBus targets (Kinesis Firehose, Step Functions, and SNS Topic) have been provisioned for you. The goal is for you to write EventBus rules to match events and verify delivery to the appropriate target.

Rule Matching Basics

Events in Amazon EventBridge are represented as JSON objects and have the following envelope signature:

```
{  "version": "0",  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",  "detail-type": "EC2 Instance State-change Notification",  "source": "aws.ec2",  "account": "111122223333",  "time": "2017-12-22T18:43:48Z",  "region": "us-west-1",  "resources": [    "arn:aws:ec2:us-west-1:123456789012:instance/ i-1234567890abcdef0"  ],  "detail": {    "instance-id": "i-1234567890abcdef0",    "state": "terminated"  }}
```

Rules use event patterns to select events and route them to targets. A pattern either matches an event or it doesn't. Event patterns are