PUBLIC SECTOR SYMPOSIUM

BRUSSELS | MARCH 28, 2023

BTT204

Building next-gen applications using event-driven architectures

Michael Hume
Senior Solutions Architect
AWS

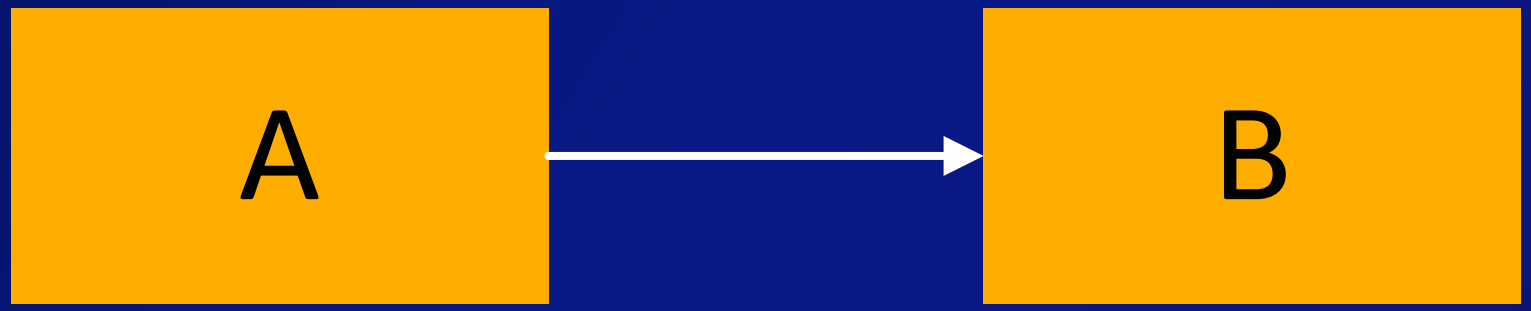
Kevin Azijn
Senior Solutions Architect
AWS

Agenda

- Enterprise Integration Patterns
- Even-driven Architecture (EDA)
- Handling event duplication using idempotency
- Design considerations, an example: Storage First Pattern

Enterprise Integration Patterns

Coupling – Integration's magic word



Coupling is a measure of independent variability between connected systems

Decoupling has a cost, both at design and runtime

Coupling isn't binary

Coupling isn't one-dimensional

Source: EnterpriseIntegrationPatterns.com

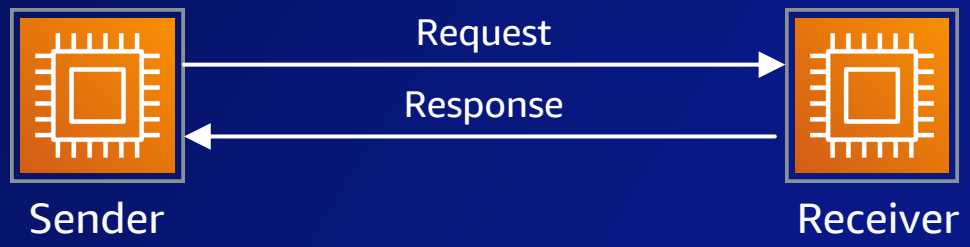
The many facets of coupling

Technology dependency:	Java vs. C++
Location dependency:	IP addresses, DNS
Data format dependency:	Binary, XML, JSON, ProtoBuf, Avro
Data type dependency:	int16, int32, string, UTF-8, null, empty
Semantic dependency:	Name, middle name, ZIP
Temporal dependency:	sync, async
Interaction style dependency:	messaging, RPC, query-style (GraphQL)
Conversation dependency:	pagination, caching, retries

Source: EnterpriseIntegrationPatterns.com

Synchronous request-response model

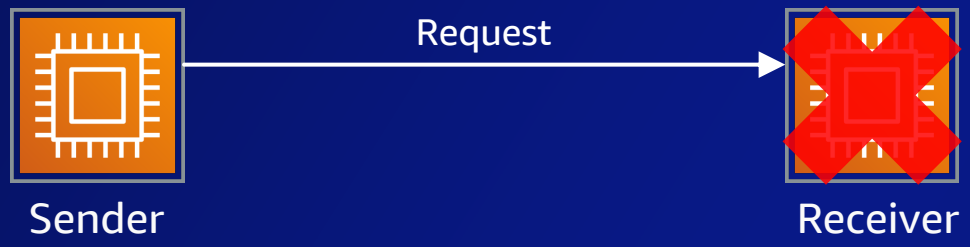
Synchronous request-response model



Advantages

- Low latency
- Simple
- Fail fast

Synchronous request-response model



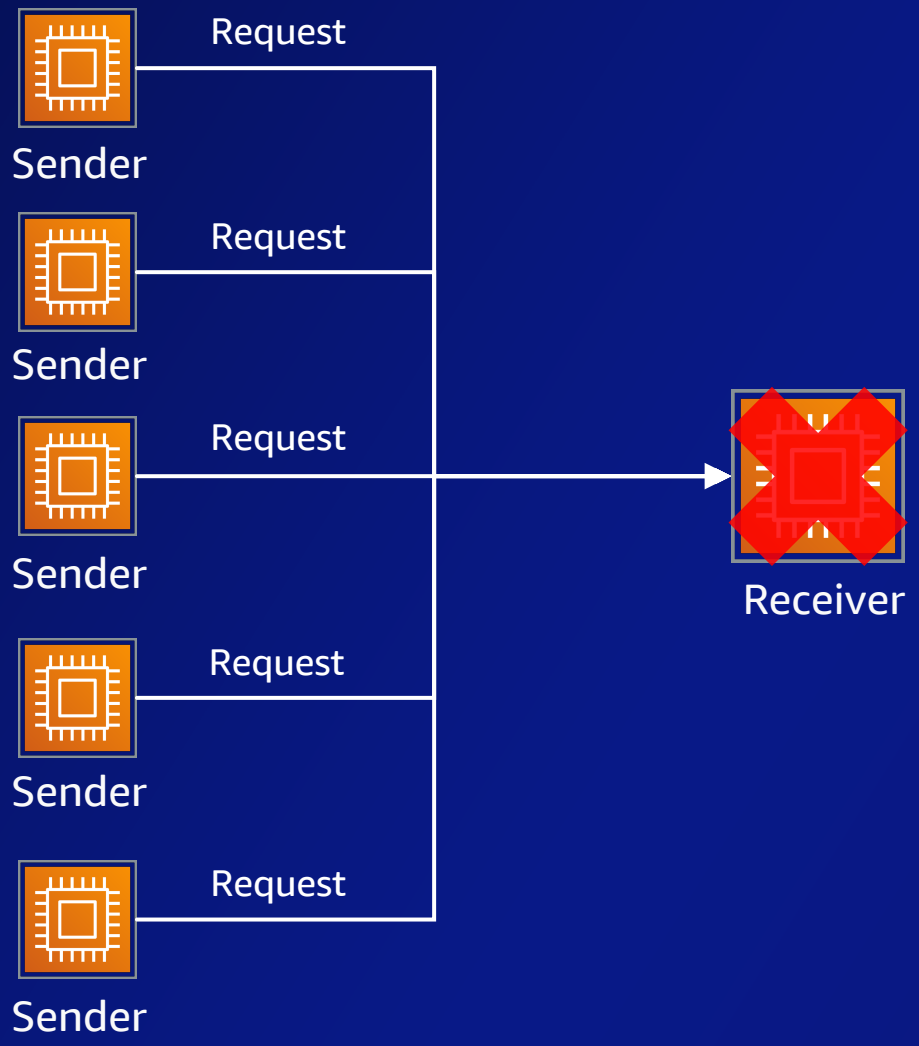
Advantages

- Low latency
- Simple
- Fail fast

Disadvantages

- Receiver failure

Synchronous request-response model



Advantages

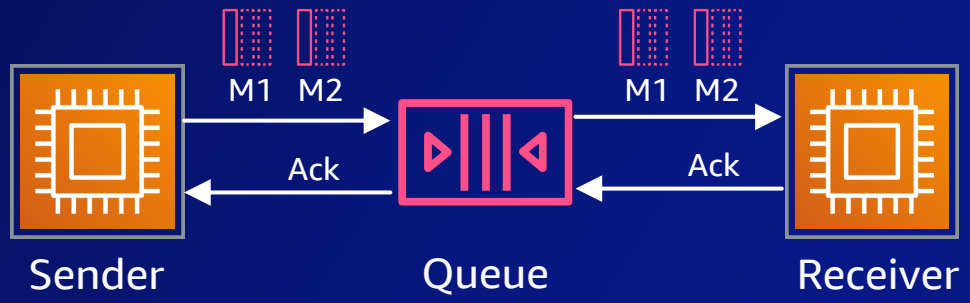
- Low latency
- Simple
- Fail fast

Disadvantages

- Receiver failure
- Receiver throttled

Asynchronous point-to-point model (queue)

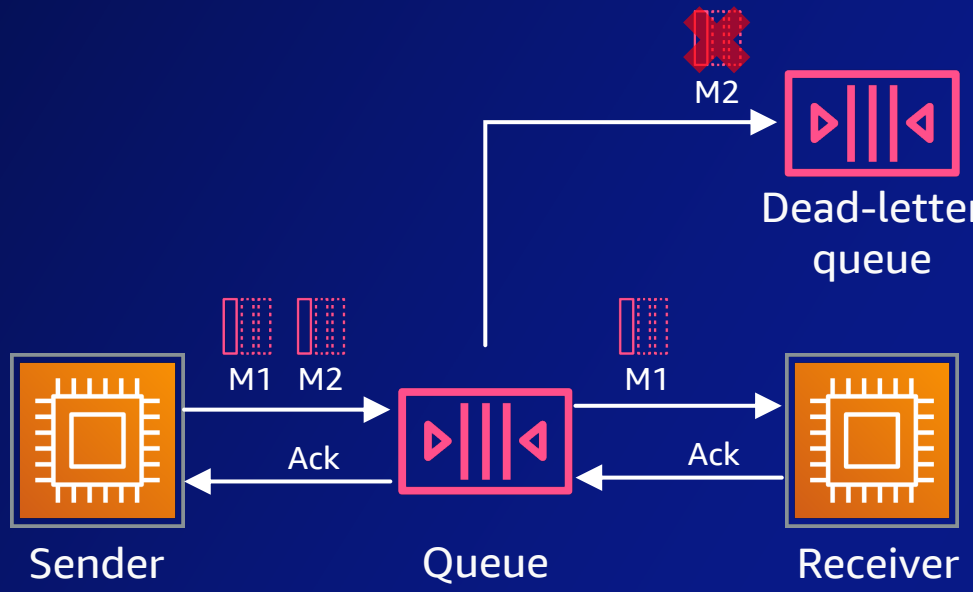
Asynchronous point-to-point model (queue)



Advantages

- Decreases **temporal coupling**
- Resilient to receiver failure
- Receiver controls consumption rate

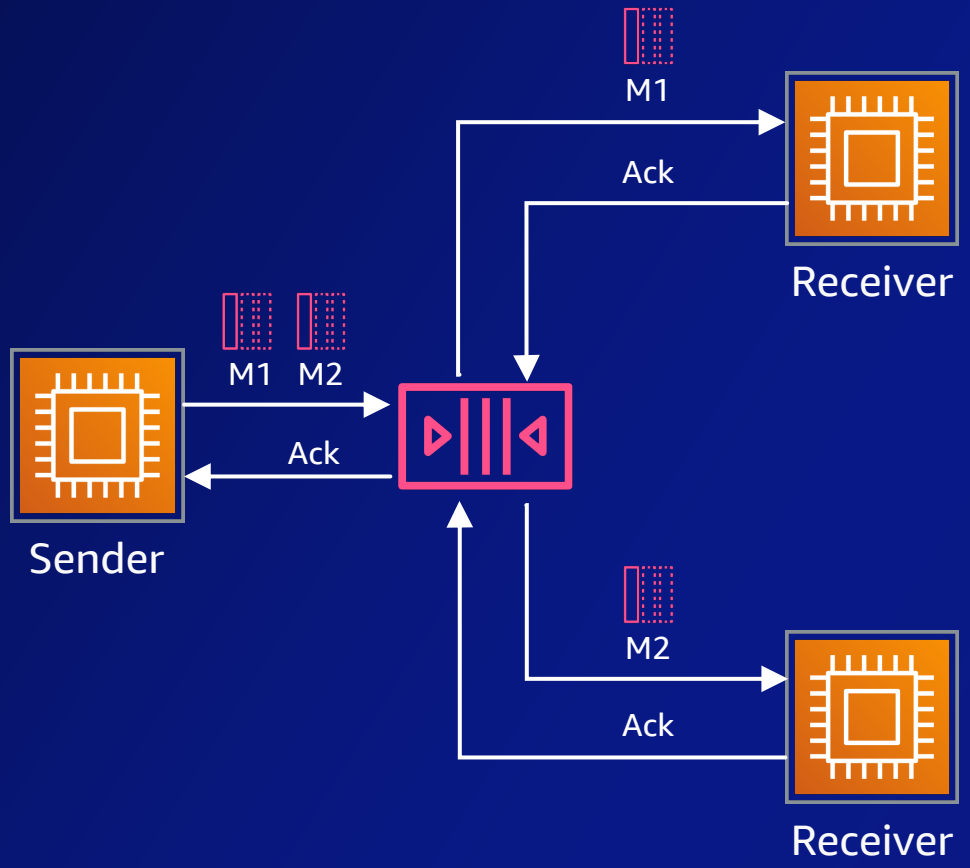
Asynchronous point-to-point model (queue)



Advantages

- …
- Dead-letter queue (DLQ) for errors

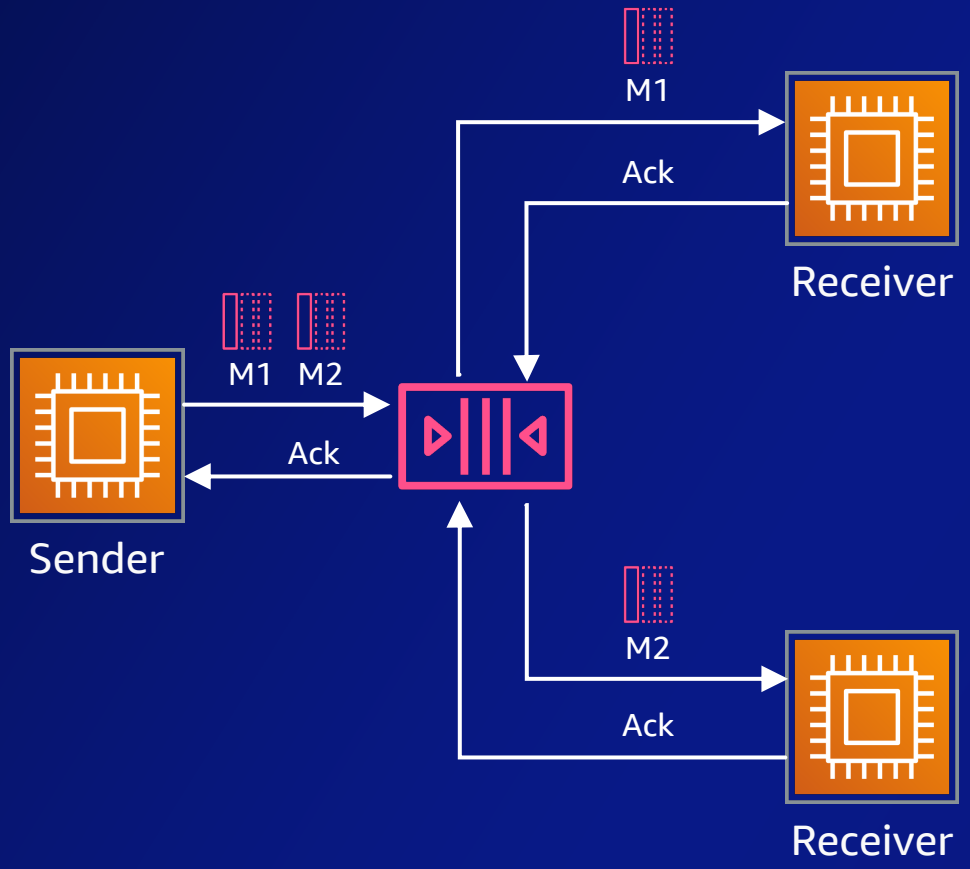
Asynchronous point-to-point model (queue)



Advantages

- …
- Only one receiver can consume each message

Asynchronous point-to-point model (queue)



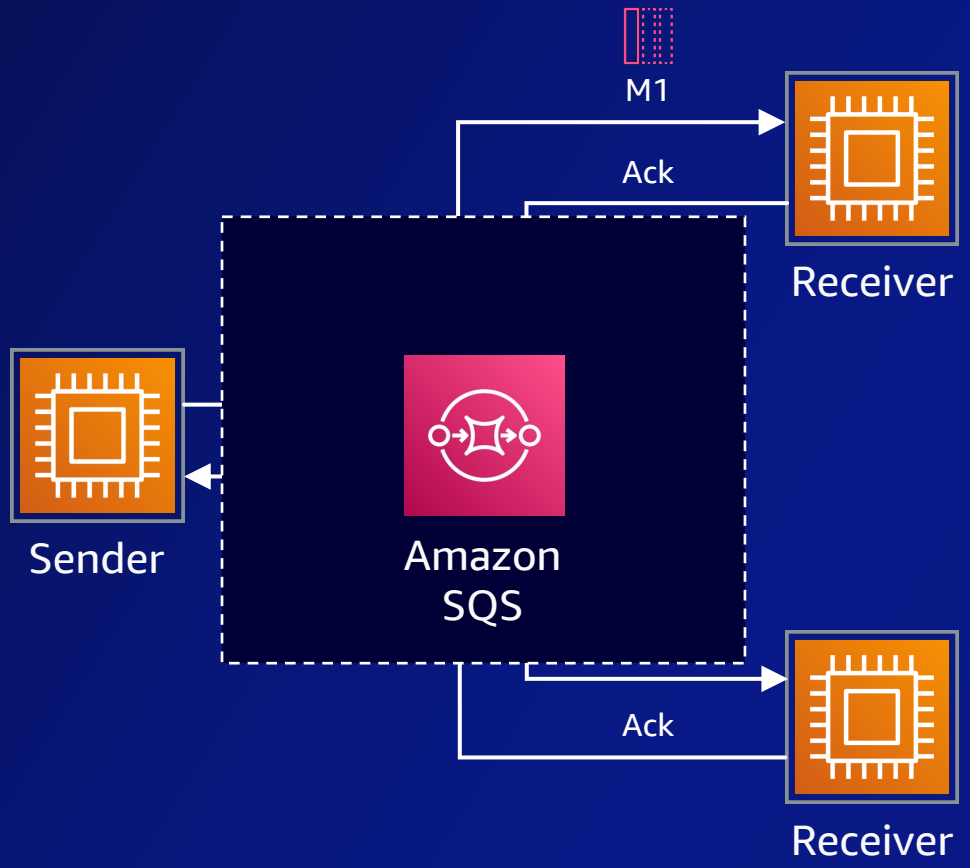
Advantages

- …
- Only one receiver can consume each message

Disadvantages

- Response correlation
- Backlog recovery time
- Fairness in multi-tenant systems

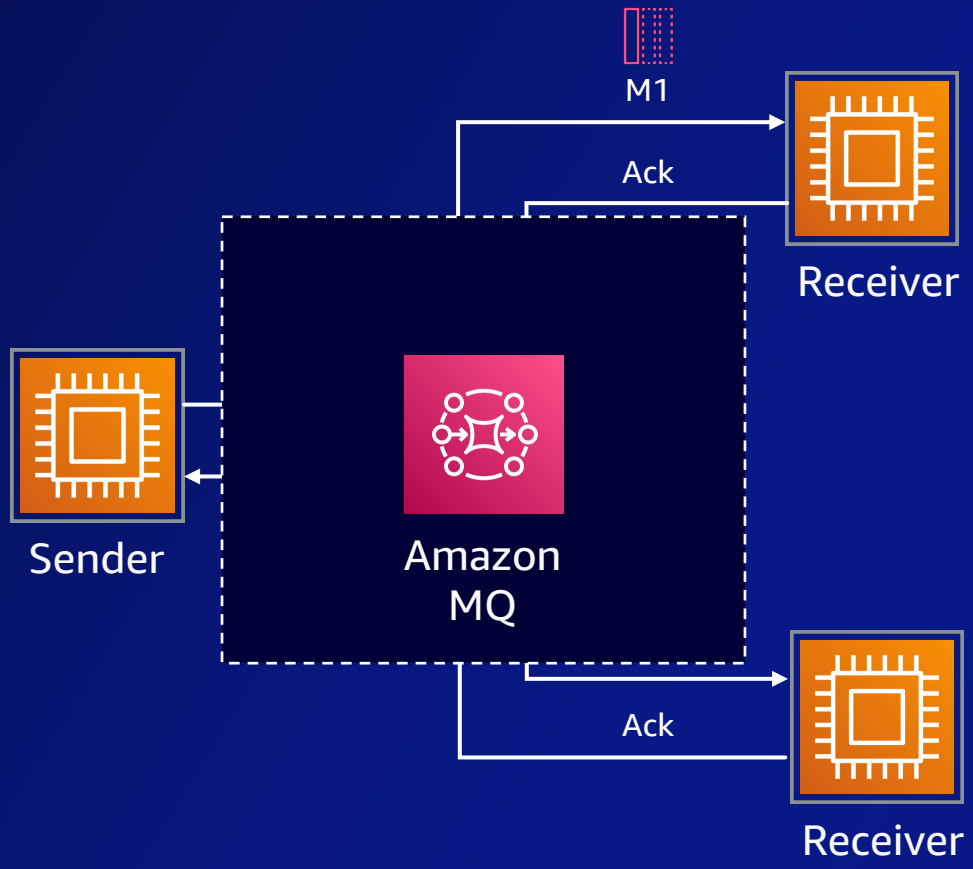
Asynchronous point-to-point model (queue)



Amazon Simple Queue Service (Amazon SQS)

- Fully managed message queue
- Scales almost infinitely
- Simple, easy-to-use API
- DLQ support
- Standard and FIFO options

Asynchronous point-to-point model (queue)

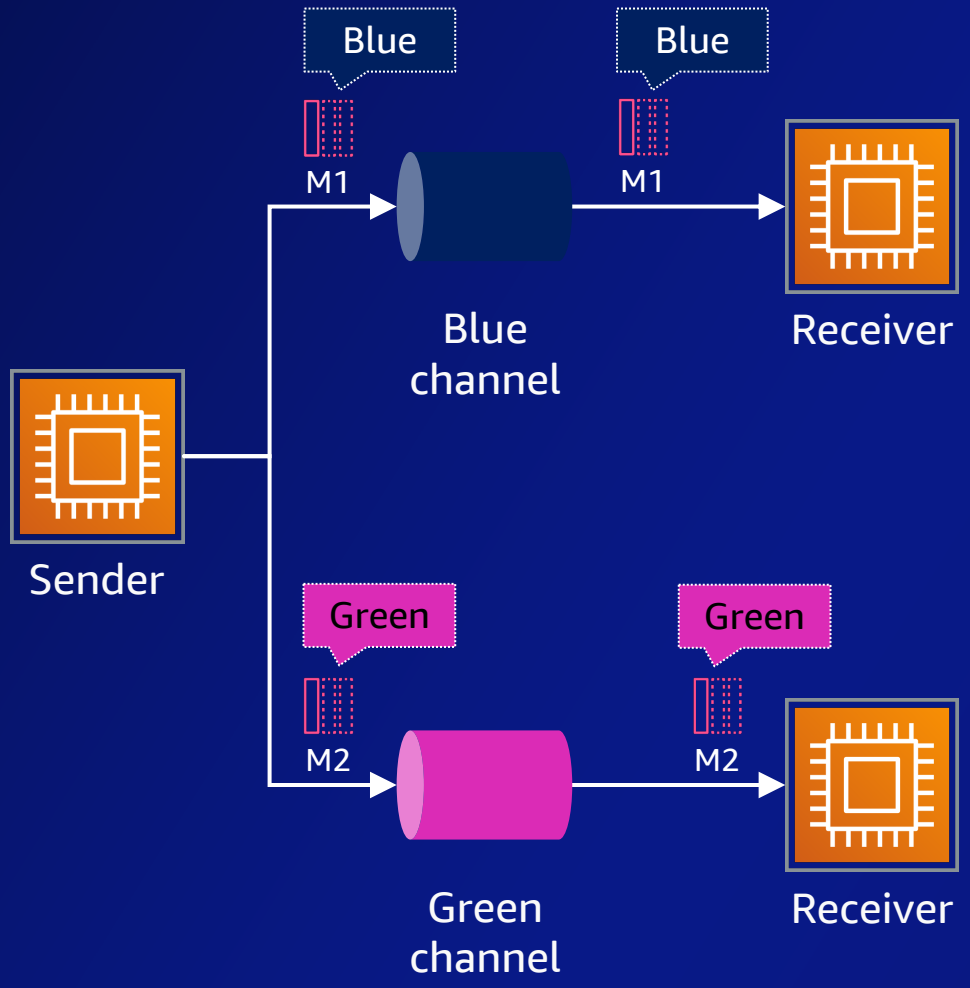


Amazon MQ

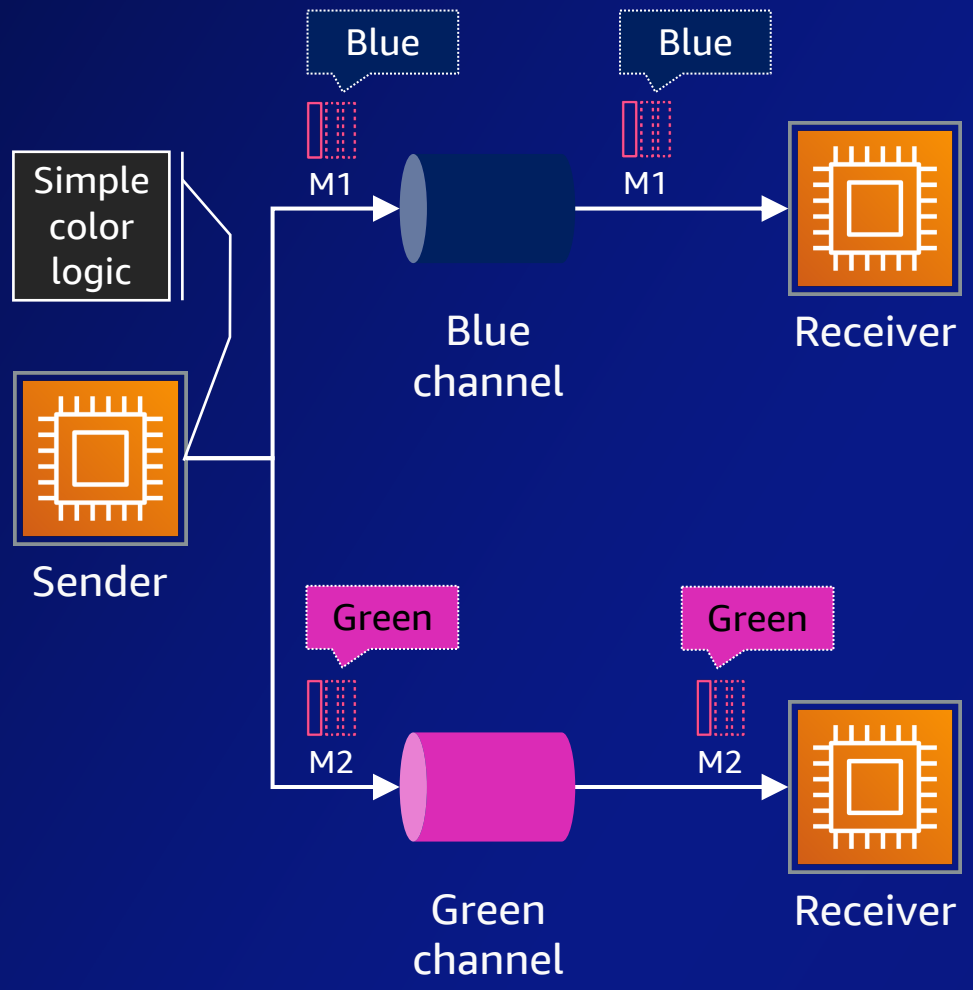
- Active MQ and RabbitMQ broker engine options
- Manages the provisioning, setup, and maintenance of message brokers
- Connects to current applications with industry-standard APIs and protocols

Asynchronous point-to-point model (router)

Asynchronous point-to-point model (router)



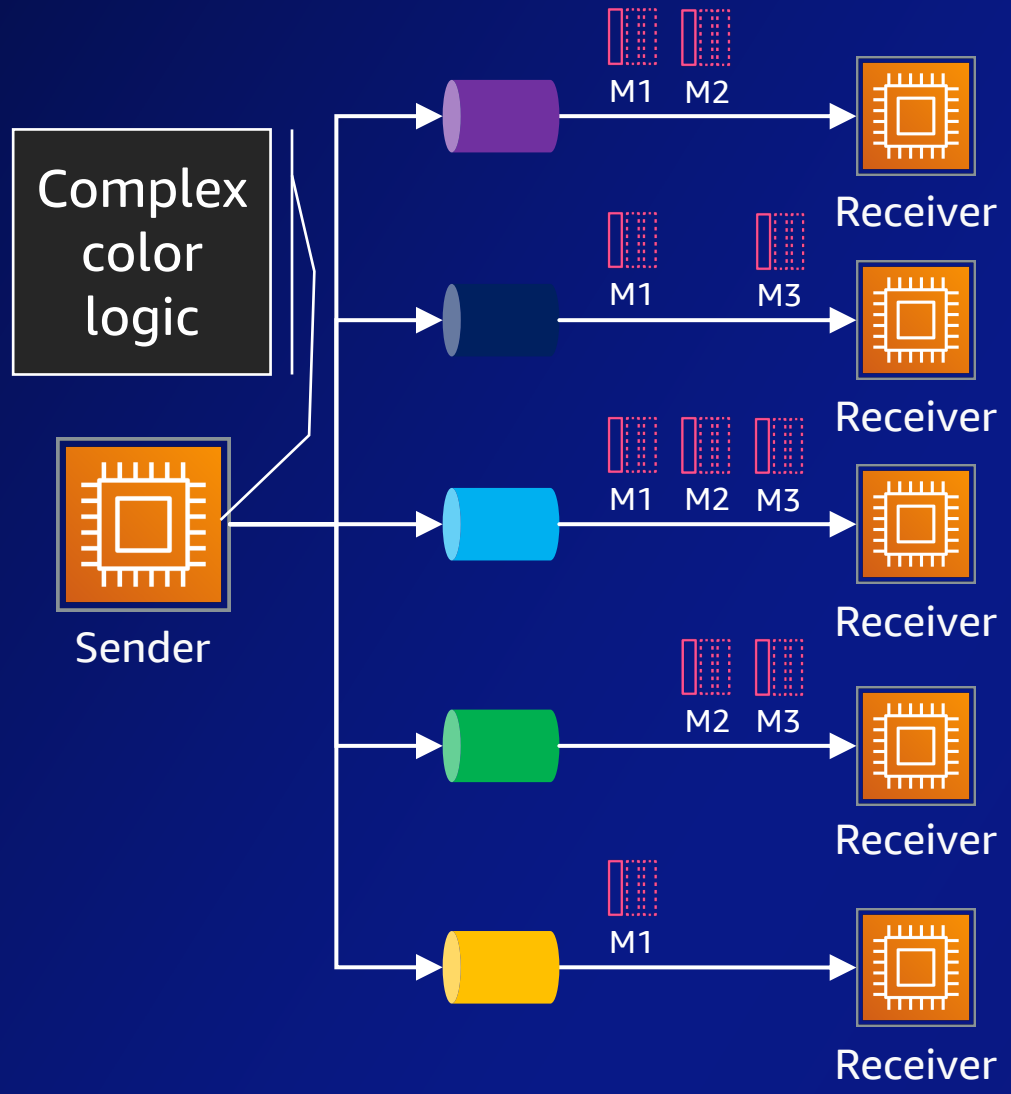
Asynchronous point-to-point model (router)



Disadvantages

- Increases **location coupling**
- Sender maintains routing logic

Asynchronous point-to-point model (router)

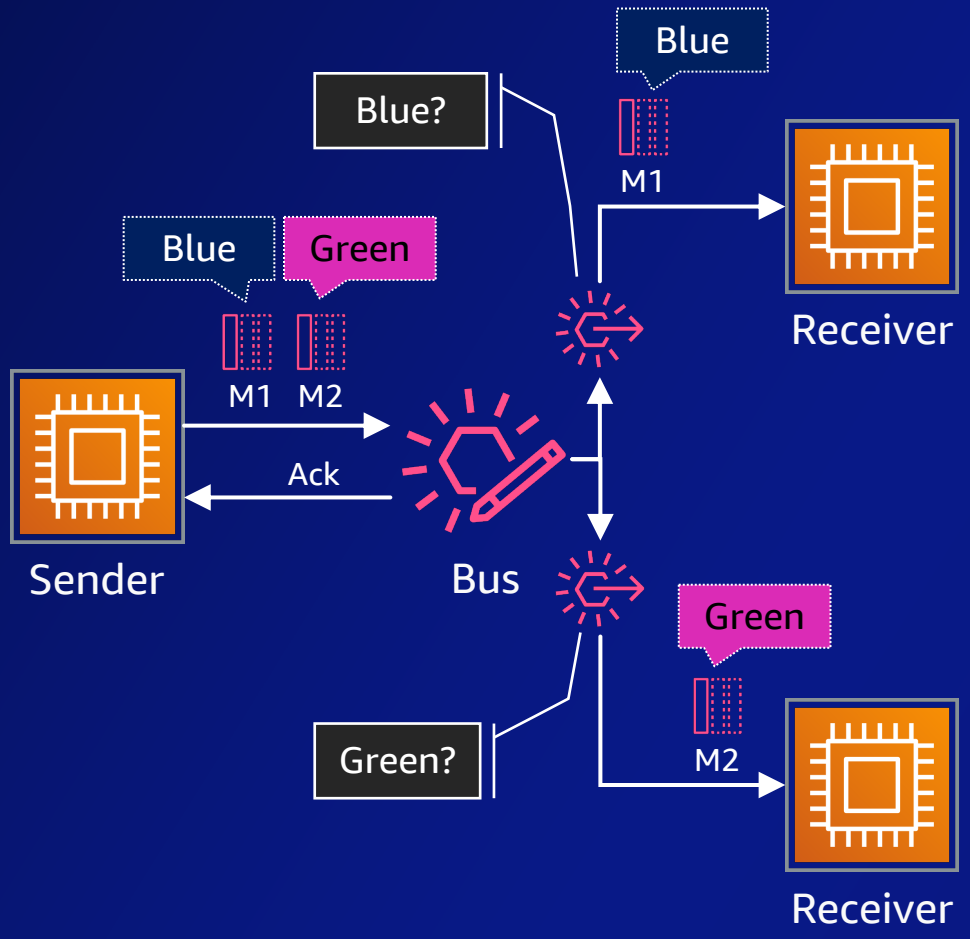


Disadvantages

- Increases **location coupling**
- Sender maintains routing logic
- Sender complexity increases with time

Asynchronous message-router model (bus)

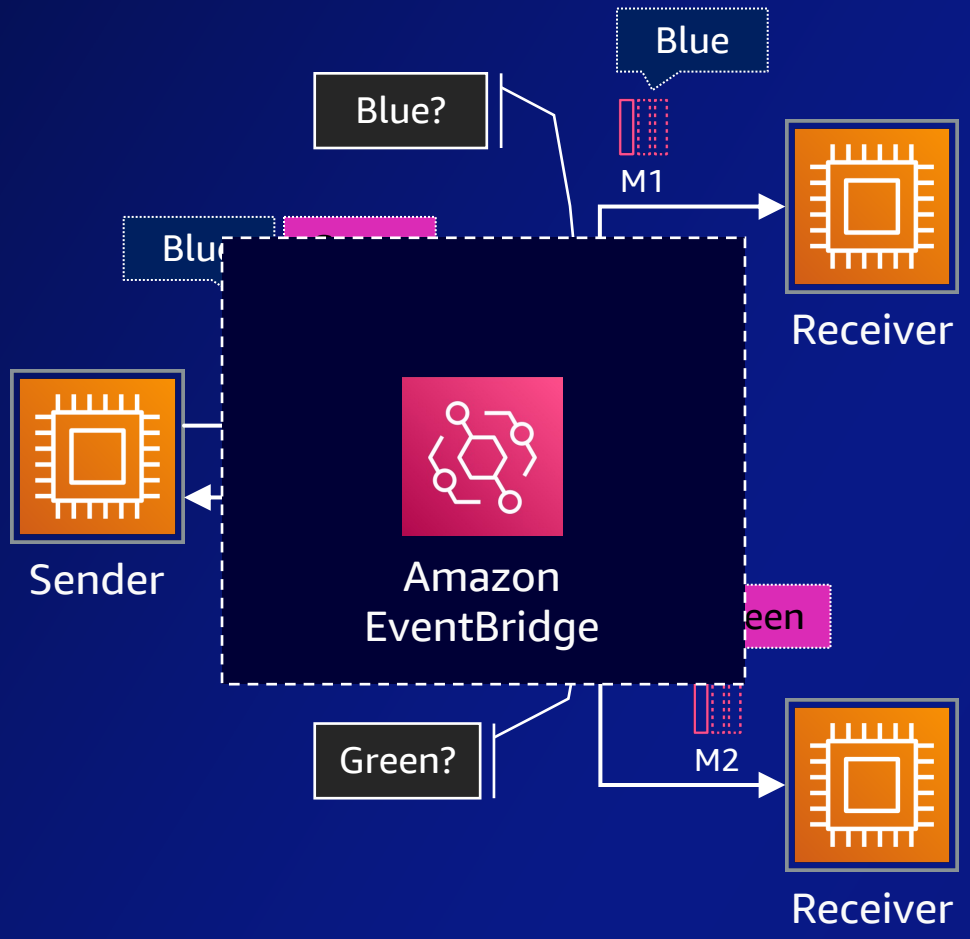
Asynchronous message-router (bus)



Advantages

- Reduces **location coupling**
- Efficient for senders and receivers

Asynchronous message-router (bus)

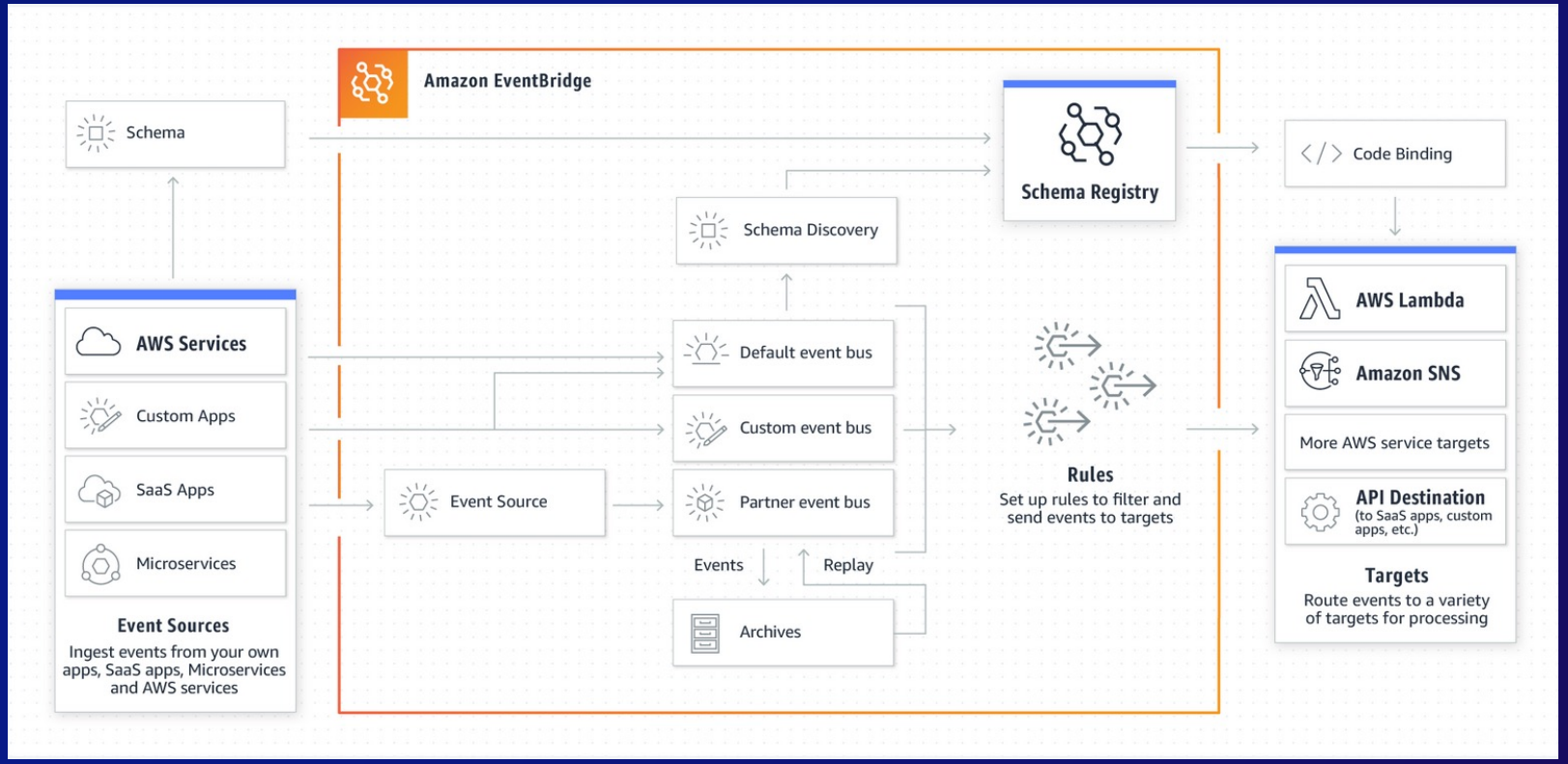


Advantages

- Reduces **location coupling**
- Efficient for senders and receivers

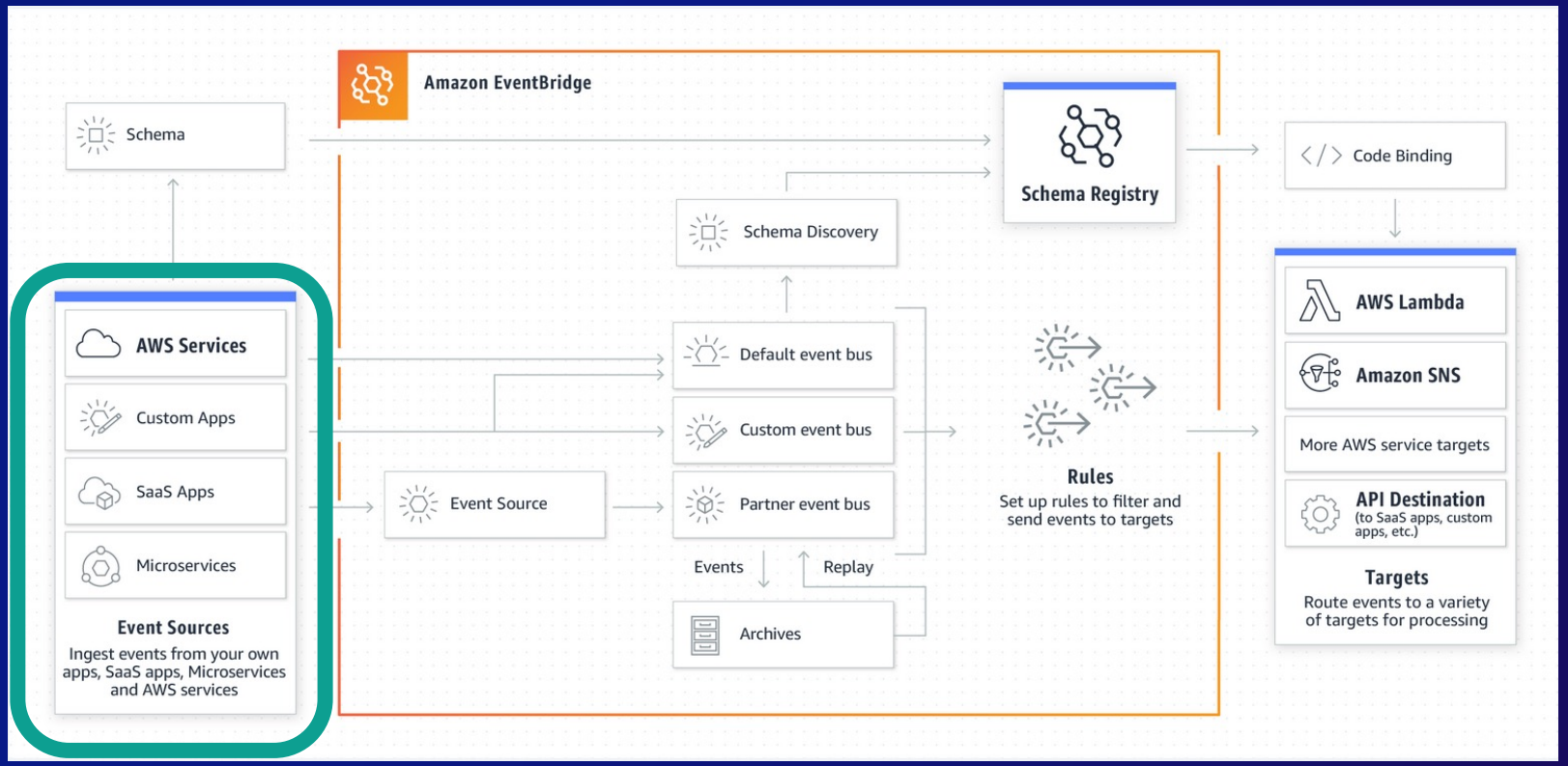
EventBridge

EventBridge is a simple, flexible, fully managed, pay-as-you-go **event bus service** that makes it easy to ingest and process data from **AWS services, your own applications**, and **SaaS applications**



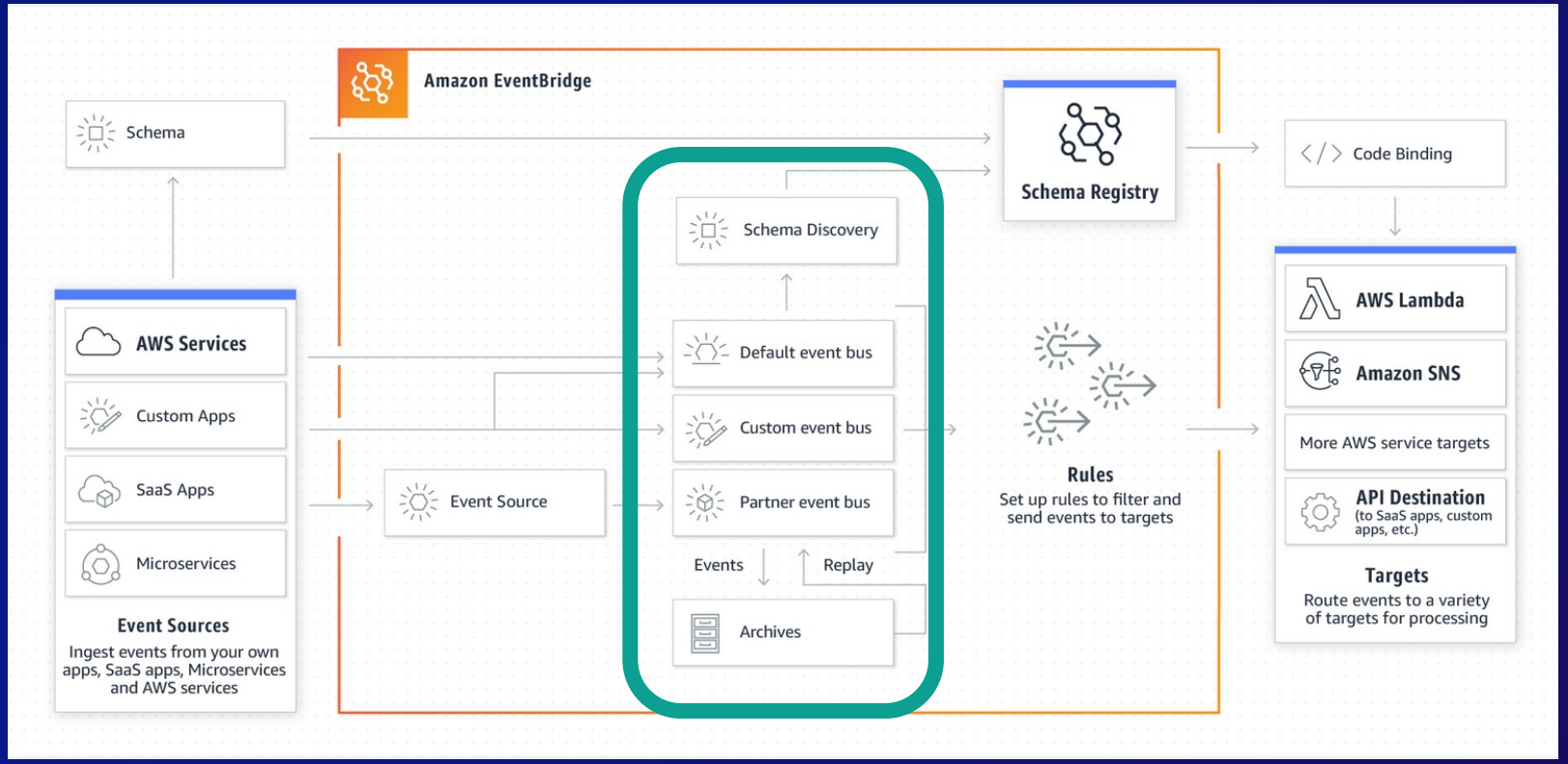
EventBridge

EventBridge is a simple, flexible, fully managed, pay-as-you-go **event bus service** that makes it easy to ingest and process data from **AWS services, your own applications**, and **SaaS applications**



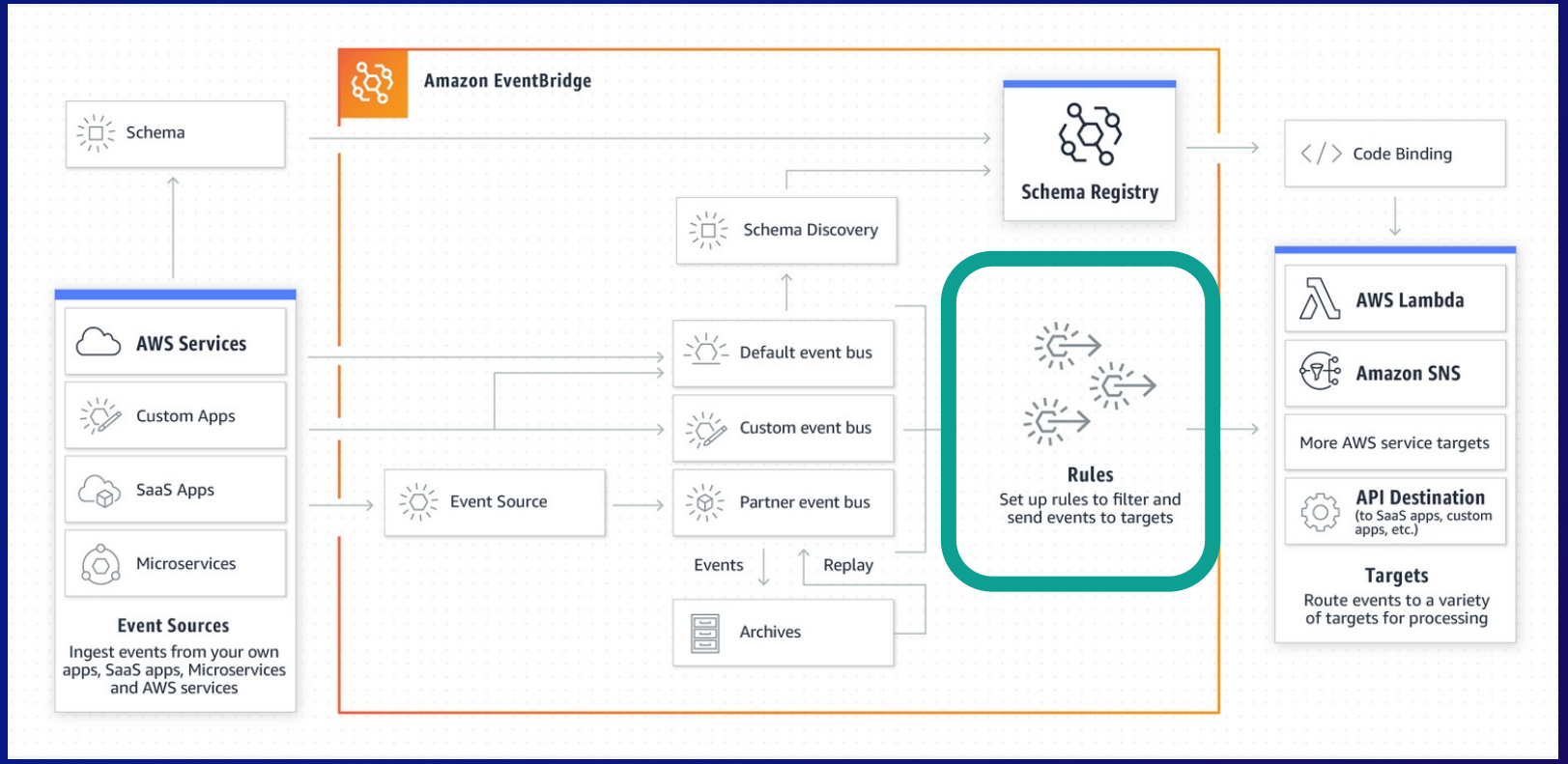
EventBridge

EventBridge is a simple, flexible, fully managed, pay-as-you-go **event bus service** that makes it easy to ingest and process data from **AWS services, your own applications**, and **SaaS applications**



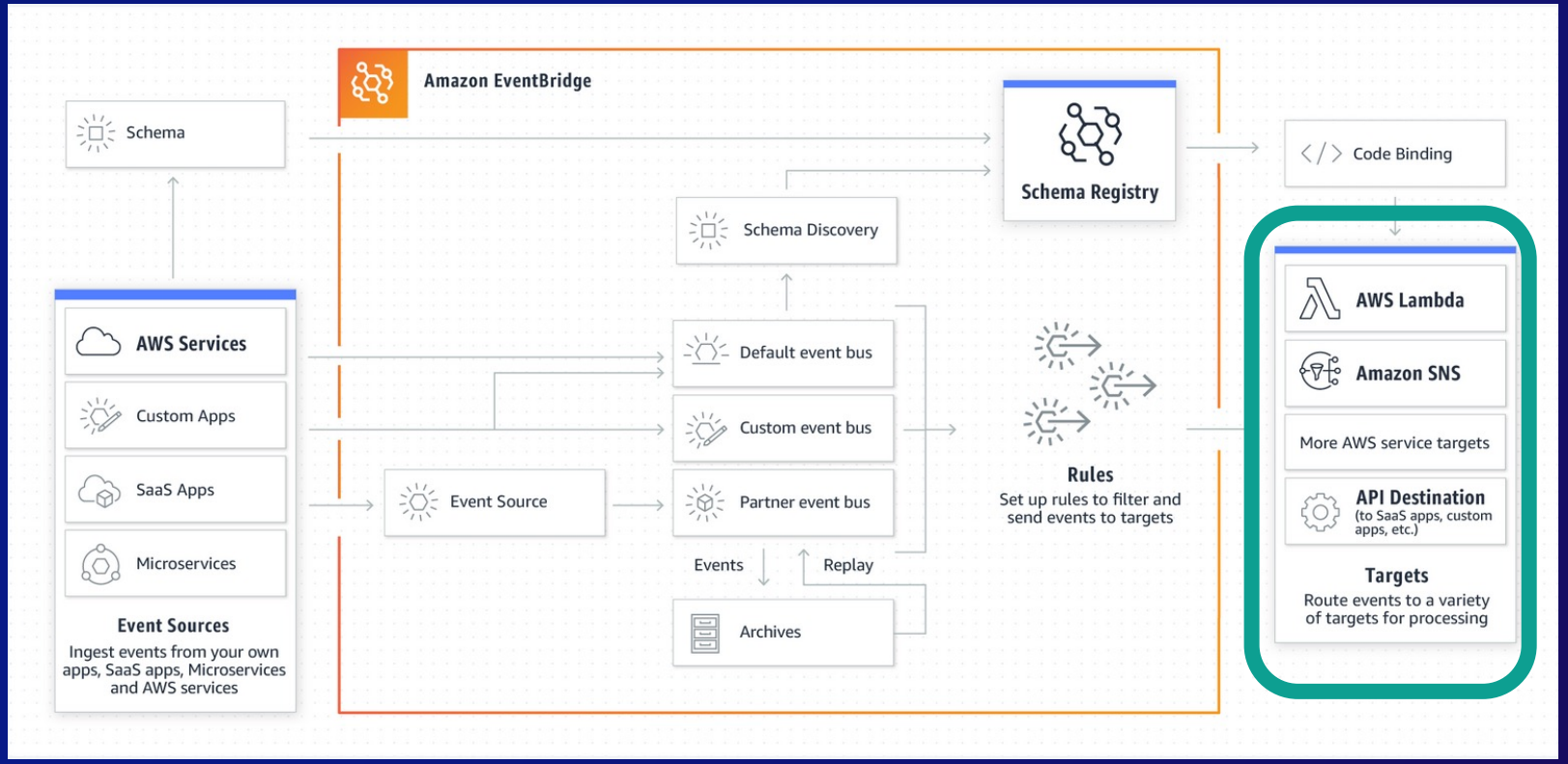
EventBridge

EventBridge is a simple, flexible, fully managed, pay-as-you-go **event bus service** that makes it easy to ingest and process data from **AWS services, your own applications**, and **SaaS applications**



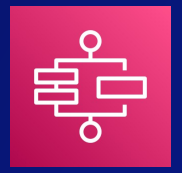
EventBridge

EventBridge is a simple, flexible, fully managed, pay-as-you-go **event bus service** that makes it easy to ingest and process data from **AWS services, your own applications**, and **SaaS applications**



Target 20+ AWS services and API destinations


AWS Lambda


AWS Step Functions

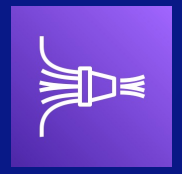

Amazon Kinesis Data Streams


Amazon CloudWatch


AWS CodePipeline


Amazon ECS


Amazon SNS


Amazon Kinesis Data Firehose


AWS Systems Manager


AWS CodeBuild


Amazon EC2


Amazon SQS


AWS Glue


Incident Manager, a capability of AWS Systems Manager


Amazon SageMaker


AWS Batch


Amazon API Gateway


Amazon Redshift


Amazon Inspector


API destinations

EventBridge content-based routing rules

EventBridge example event

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
    },
    "data": {
      "order-id": "1073459984",
      "created-at": "2021-11-26T16:05:09-04:00",
      "price": 24.62
      "currency": "AU",
    }
  }
}
```

EventBridge content-based routing rules

EventBridge example event

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
    },
    "data": {
      "order-id": "1073459984",
      "created-at": "2021-11-26T16:05:09-04:00",
      "price": 24.62
      "currency": "AU",
    }
  }
}
```

EventBridge example rule

```
{
  "detail": {
    "data":  {
      "currency": ["AU", "NZ"]
    }
  }
}
```

EventBridge content-based routing rules

EventBridge example event

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
    },
    "data": {
      "order-id": "1073459984",
      "created-at": "2021-11-26T16:05:09-04:00",
      "price": 24.62
      "currency": "AU",
    }
  }
}
```

EventBridge example rule

```
{
  "detail": {
    "data":  {
      "currency": ["AU", "NZ"]
    }
  }
}
```

EventBridge content-based routing rules

EventBridge example event

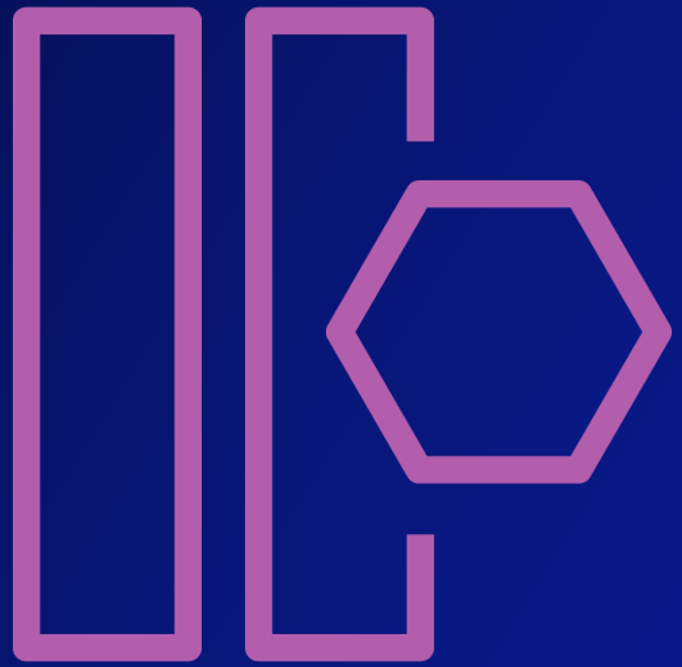
```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
    },
    "data": {
      "order-id": "1073459984",
      "created-at": "2021-11-26T16:05:09-04:00",
      "price": 24.62
      "currency": "AU",
    }
  }
}
```

EventBridge example rule

```
{
  "detail": {
    "data":  {
      "currency": ["AU", "NZ"]
    }
  }
}
```

✅ Match!

Event-driven architecture - EDA



event

[i-’vent] noun

A signal that a system’s state has changed

Properties of events

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
      "idempotency-key": "c1b95b88"
    },
    "data": {
      "order-id": "1073459984"
    }
}
```

- Events are signals that a system's state has changed
- Events occur in the past (e.g. OrderCreated)
- Events cannot be changed (immutable)
- Decrease semantic coupling by restricting information to key data

Sparse events vs. full state descriptions

Order 123 was created at 10:47 a.m. by customer 456

Sparse events

Order 123 was created at 10:47 a.m. by customer 456. The current status is Open, the total was $237.51, …

Full state description

Considerations with sparse events

Order 123 was created at 10:47 a.m. by customer 456

What are the details for order 123?

Events

Considerations with full state descriptions

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
      "idempotency-key": "c1b95b88"
    },
    "data": {
      "order-id": "1073459984",
      "status": "Open",
      "total": "237.51"
    }
}
```

- Event schemas should be backwards compatible

Considerations with full state descriptions

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
      "idempotency-key": "c1b95b88"
    },
    "data": {
      "order-id": "1073459984",
      "status": "Open",
      "total": "237.51"
    }
}
```

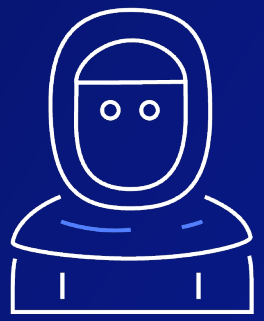
- Event schemas should be backwards compatible
- Cost to calculate values can increase over time

Considerations with full state descriptions

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
      "idempotency-key": "c1b95b88"
    },
    "data": {
      "order-id": "1073459984",
      "status": "Open",
      "total": "237.51"
    }
}
```

- Event schemas should be backwards compatible
- Cost to calculate values can increase over time

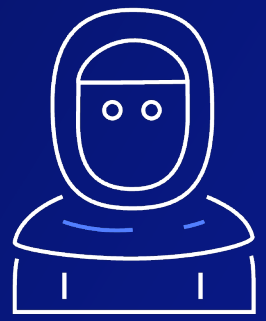
Choreograph events *between domains* using subscriptions



Retail

Fulfillment

Choreograph events *between domains* using subscriptions



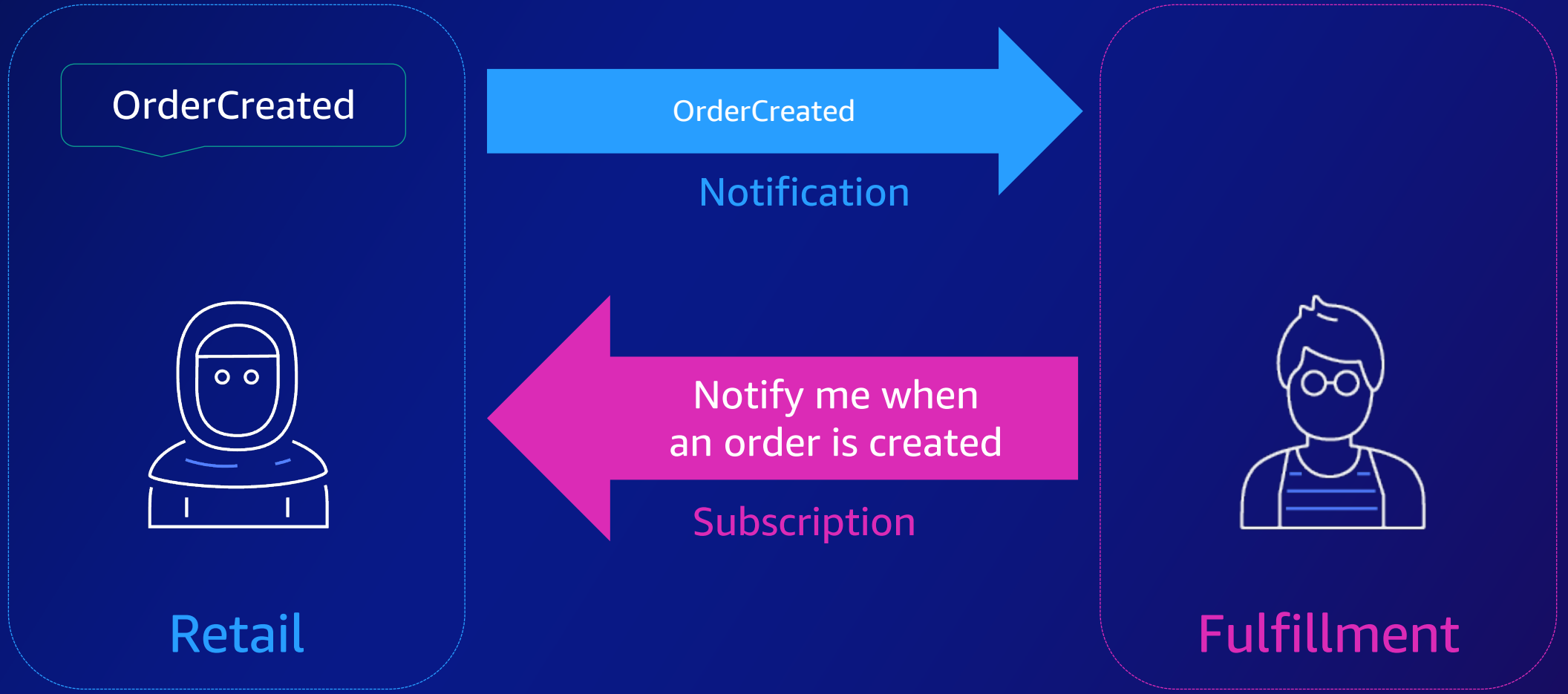
Retail

Notify me when an order is created

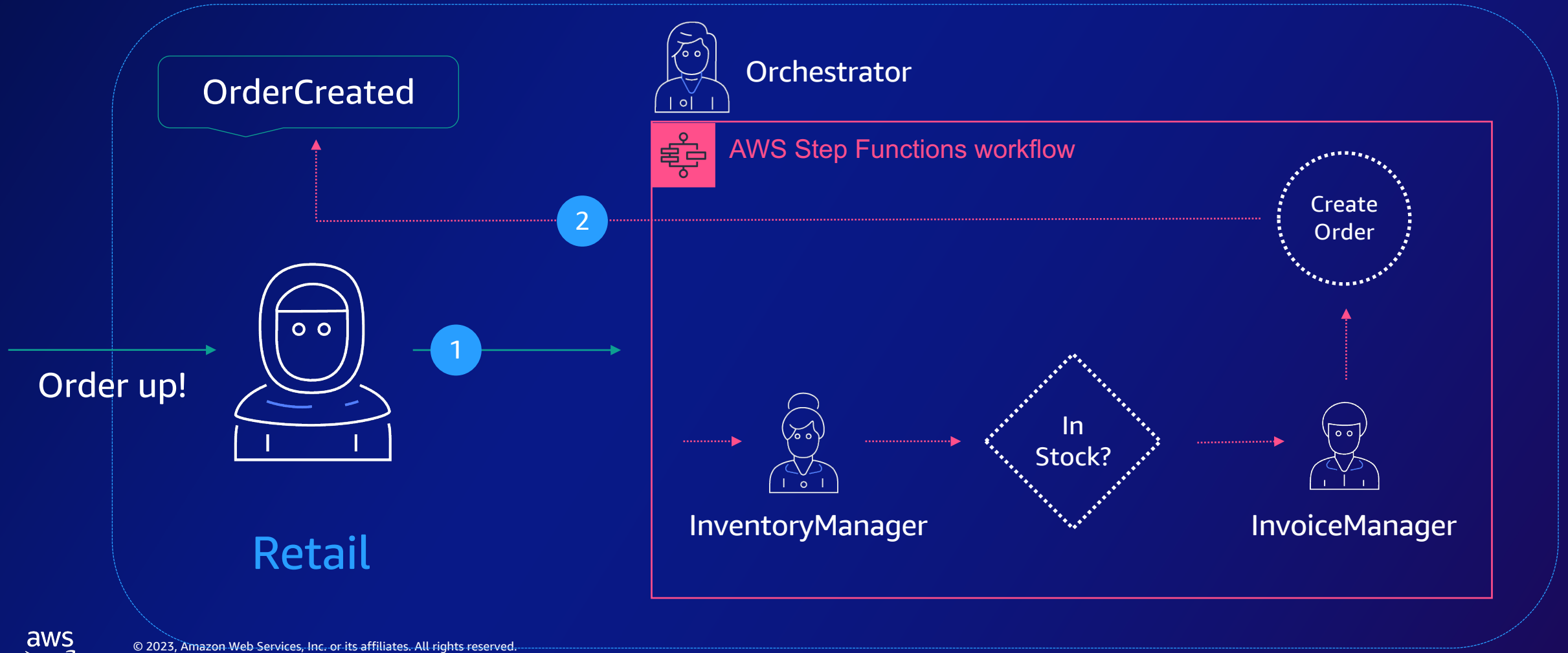
Subscription

Fulfillment

Choreograph events *between domains* using subscriptions

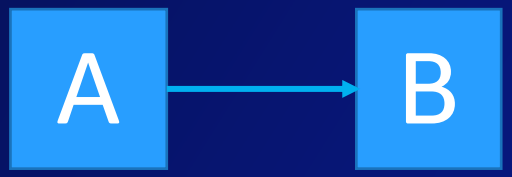


Orchestrate a business process *within a domain*, resulting in a published event



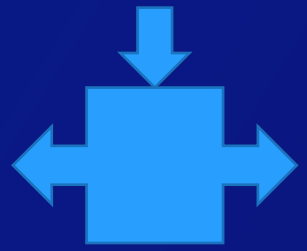
What can we do with orchestration?

"I want to sequence tasks"

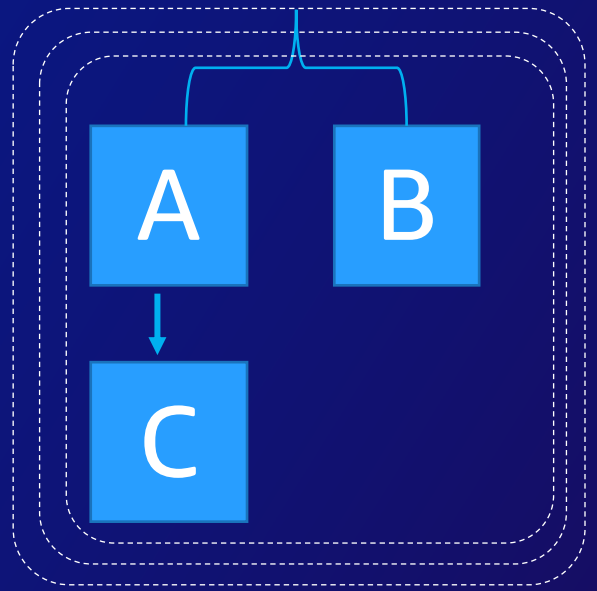


"I want to retry failed tasks"

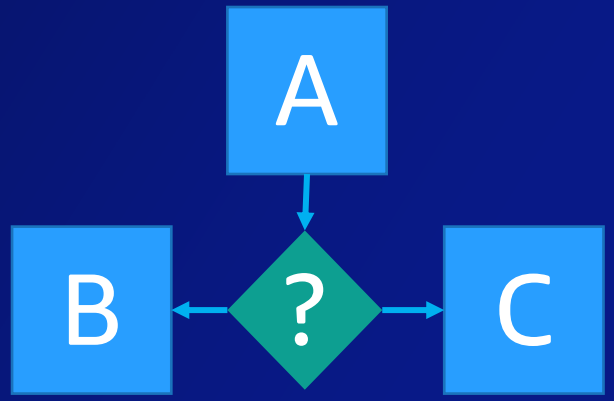
"I want try/catch/finally"



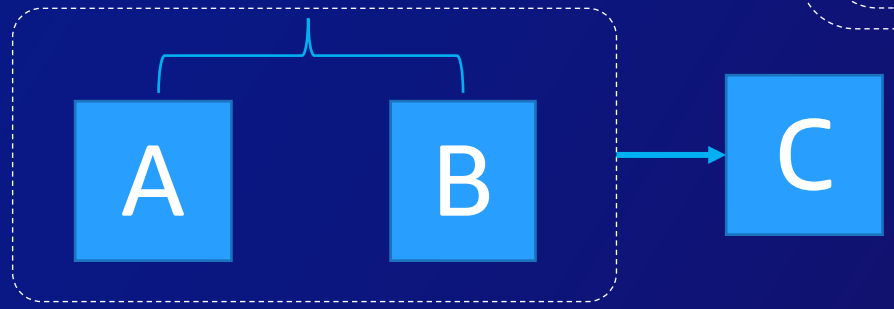
"I want concurrent and iterative tasks"



"I want to select tasks based on data"



"I want to run tasks in parallel"

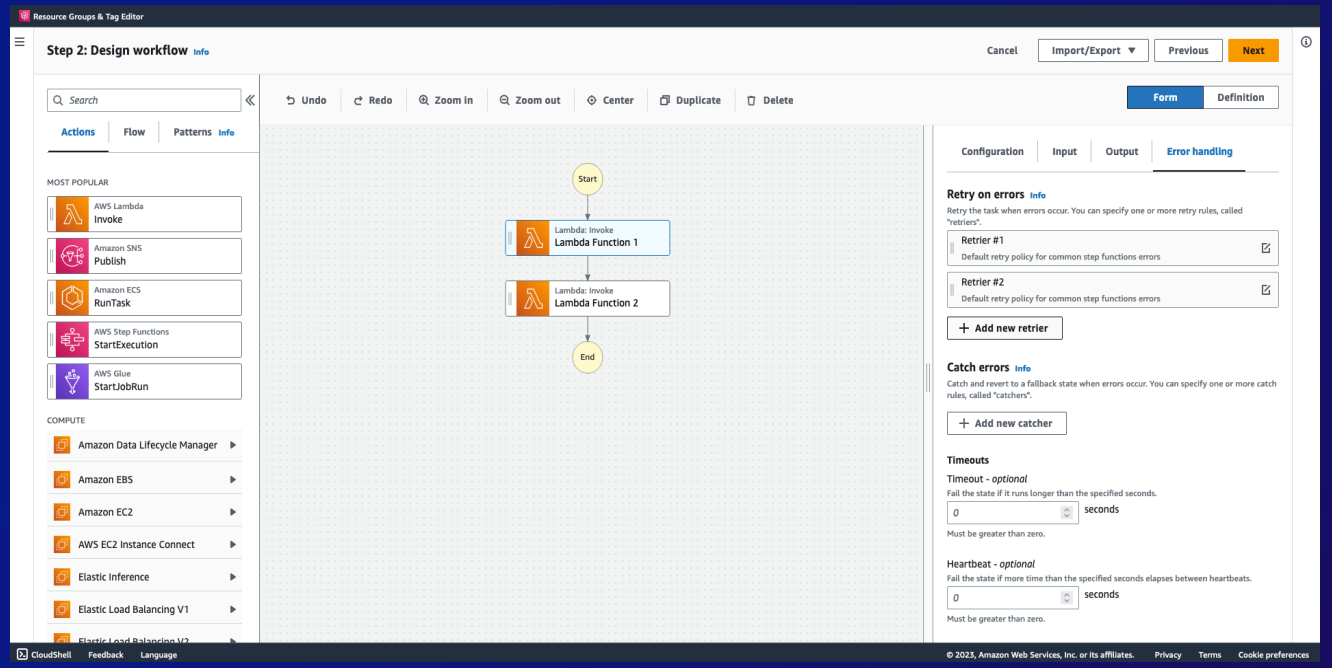


AWS Step Functions

The workflows you build with Step Functions are called state machines, and each step of your workflow is called a state.

When you execute your state machine, each move from one state to the next is called a state transition.

You can reuse components, easily edit the sequence of steps or swap out the code called by task states as your needs change.



AWS STEP FUNCTIONS WORKFLOW STUDIO

Visual Workflows

Define

JSON - Amazon States Language

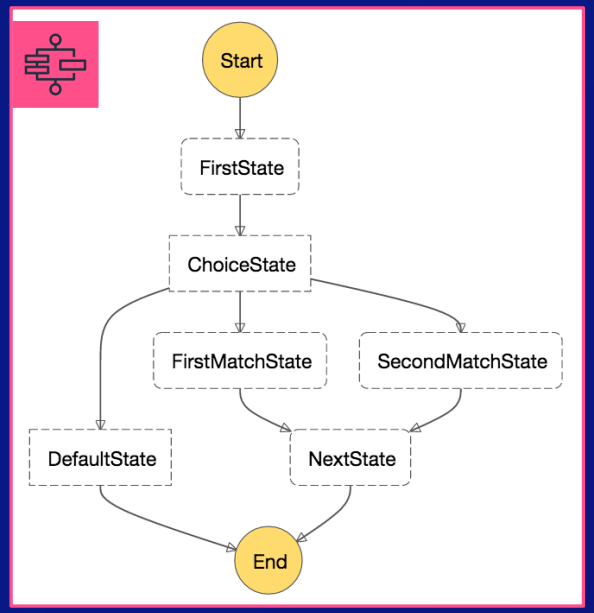
```
Code
{
  "Comment": "An AWL example using a choice state.",
  "StartAt": "FirstState",
  "States": {
    "FirstState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:FUNCTION_NAME",
      "Next": "ChoiceState"
    },
    "ChoiceState": {
      "Type" : "Choice",
      "Choices": [
        {
```

CDK TypeScript, JavaScript, Python, Java, C#

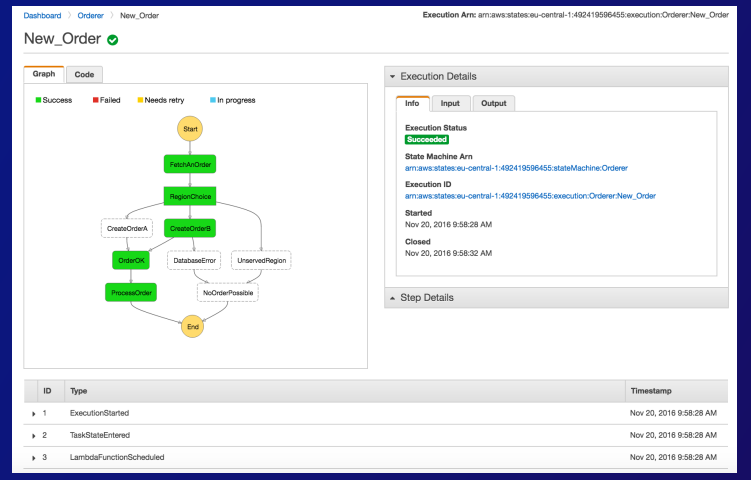
Data Science SDK

Python

Visualize



Execute and Monitor



Step Functions integration types

Optimized integrations

Customized to simplify the usage of 17 AWS services

Supported Integration patterns:

- Request Response
- Run a Job (.sync)
- Wait for a Callback (.waitForTaskToken)

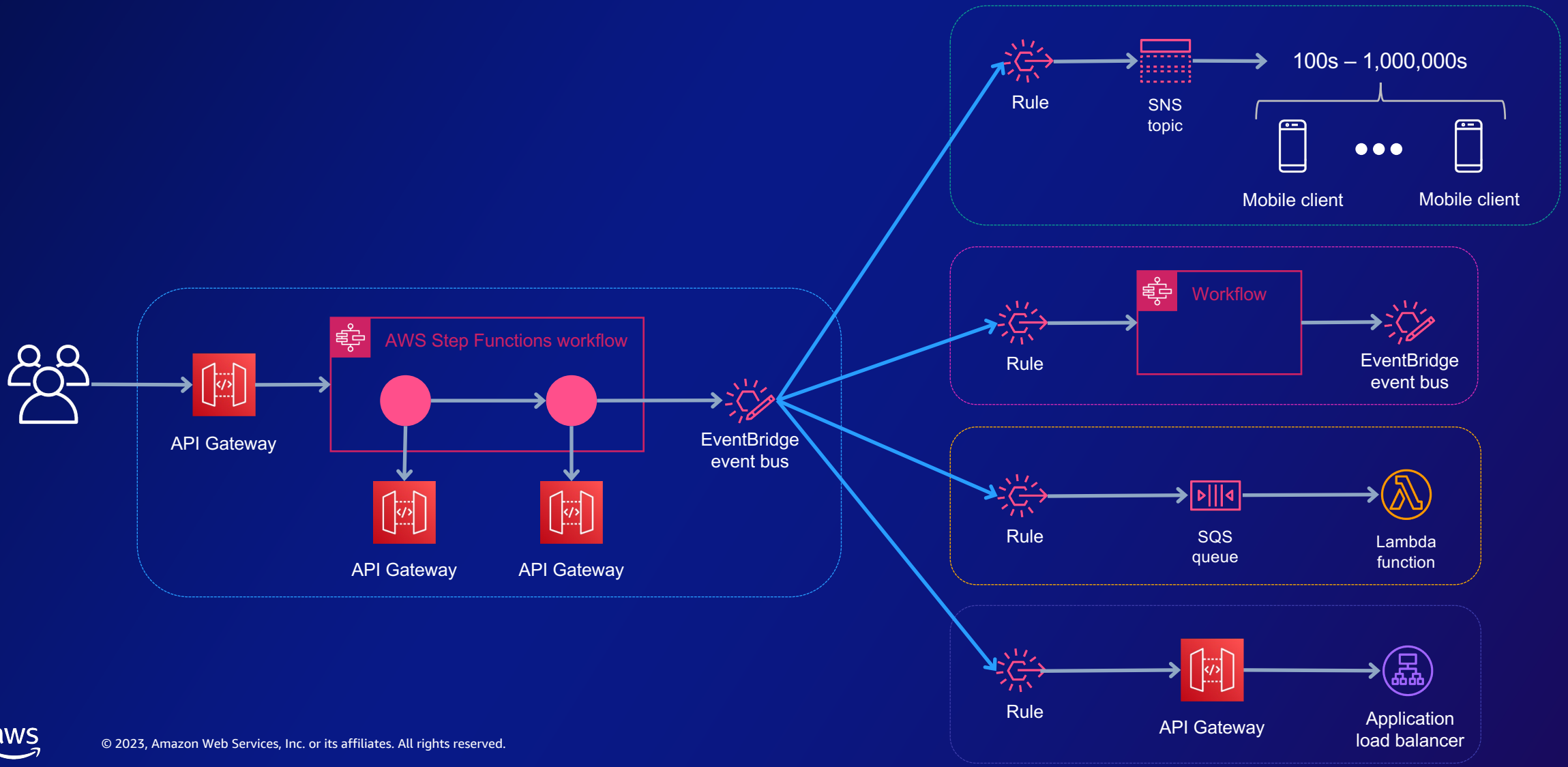
AWS SDK integrations

Call 200 AWS services directly (9000+ API actions)

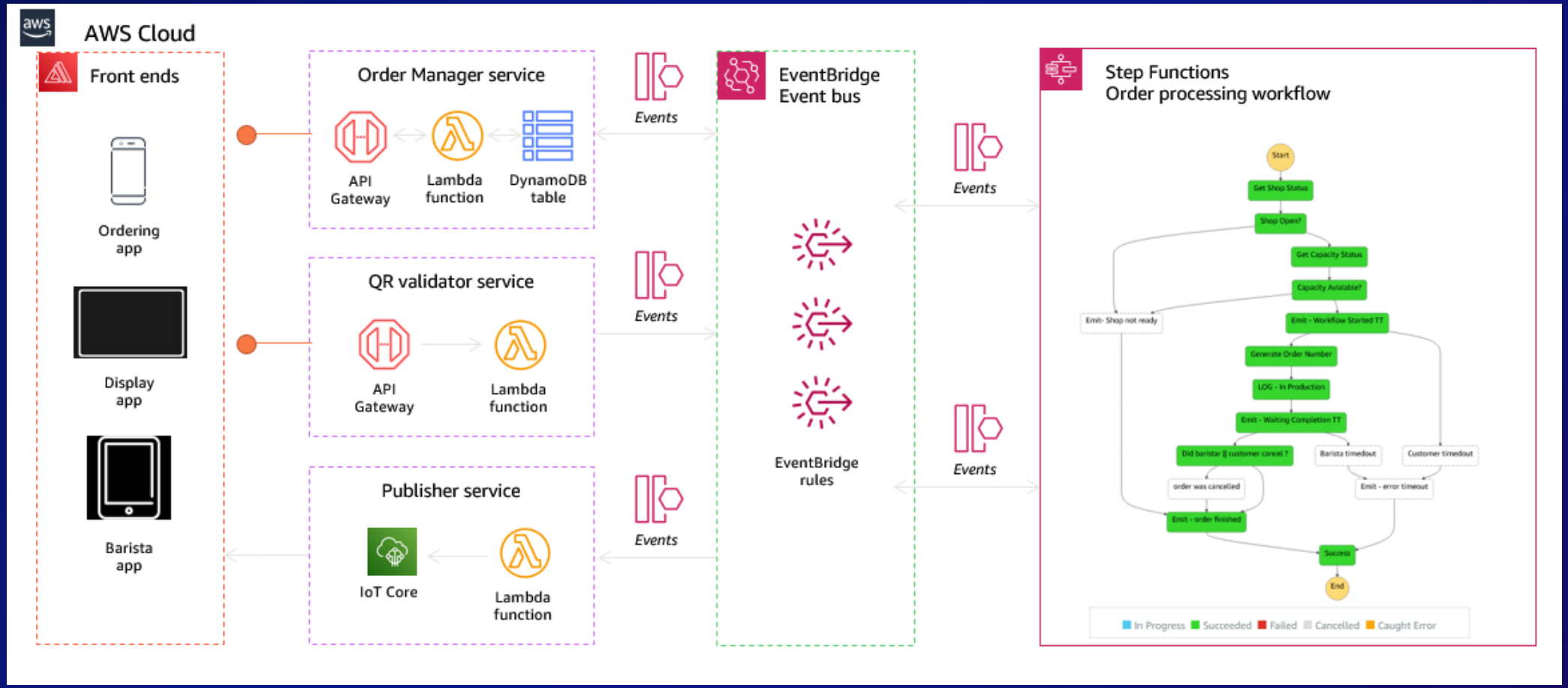
Supported Integration patterns:

- Request Response
- Wait for a Callback (.waitForTaskToken)

Better together: Orchestration + Choreography



Serverlesspresso: Building an event-driven application from the ground up



Handling event duplication using idempotency

What is idempotence, idempotency, idempotent?

"[Mathematical operations] that can be applied multiple times without changing the result." – Wikipedia

```
f(x) = x + 0 = x
f(x) = x * 1 = x
```

"A message that has the same effect whether it is received once or multiple times." – "Enterprise Integration Patterns" (Hohpe, Woolf)

"Did the internet charge my credit card twice?" – Text message (Mom)

Event uniqueness

Idempotency

Operation will return the same results whether it is called once or multiple times (EIP).

Idempotency Key

Assigned to the message by the sender to simplify deduplication by the receiver.

```
{
  "source": "com.orders",
  "detail-type": "OrderCreated",
  "detail": {
    "metadata": {
      "idempotency-key": "c1b95b88"
    },
    "data": {
      "order-id": "1073459984"
    }
}
```

Example: EventBridge idempotency identifier

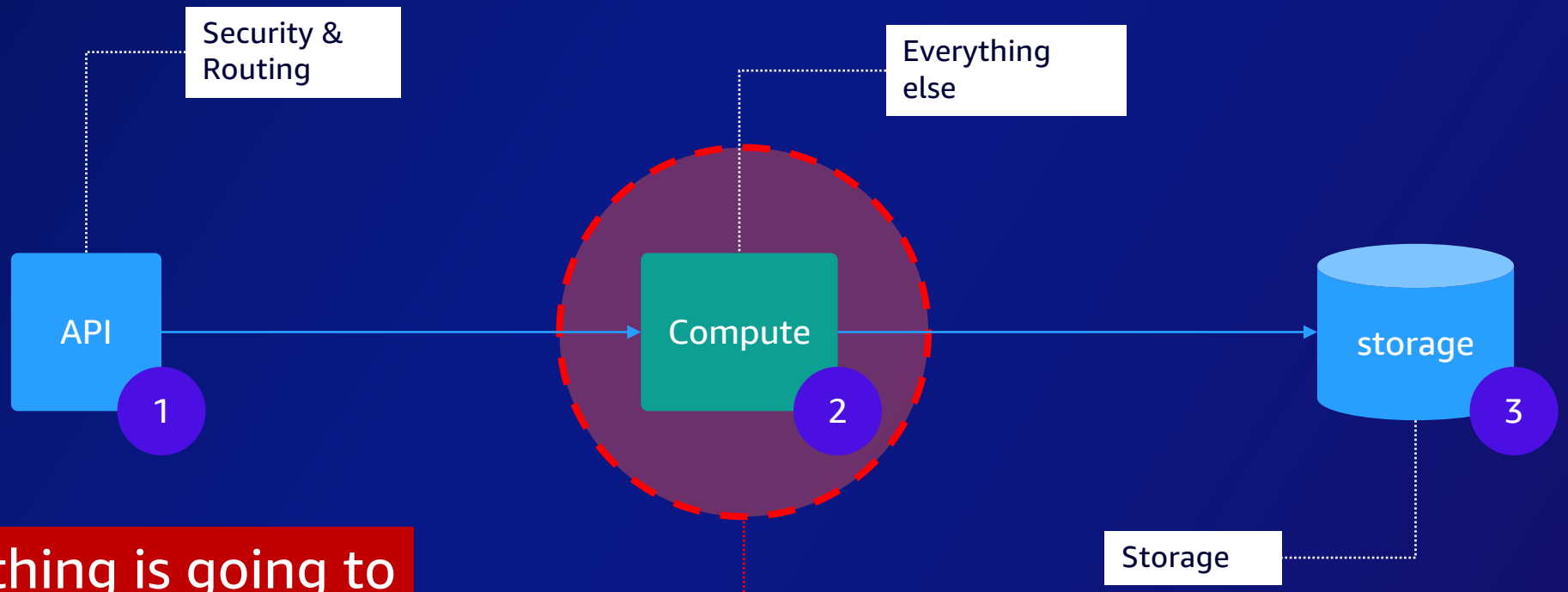
```
{
  "version": "0",
  "id": "61a15356-f8d3-4b6e-7da9-5bfccde8016d",
  "detail-type": "OrderCreated",
  "source": "com.orders",
  "account": "068896461592",
  "time": "2022-05-01T22:15:20Z",
  "region": "us-east-1",
  "detail": {
    "metadata": {
      "idempotency-key": "AF8074B2-3C23-415B-B465-71A849C63452"
    },
    "data": {
      "order-id": "1073459984"
    }
}
```

!! Event.id is not an idempotency ID !!

Client-provided idempotency ID

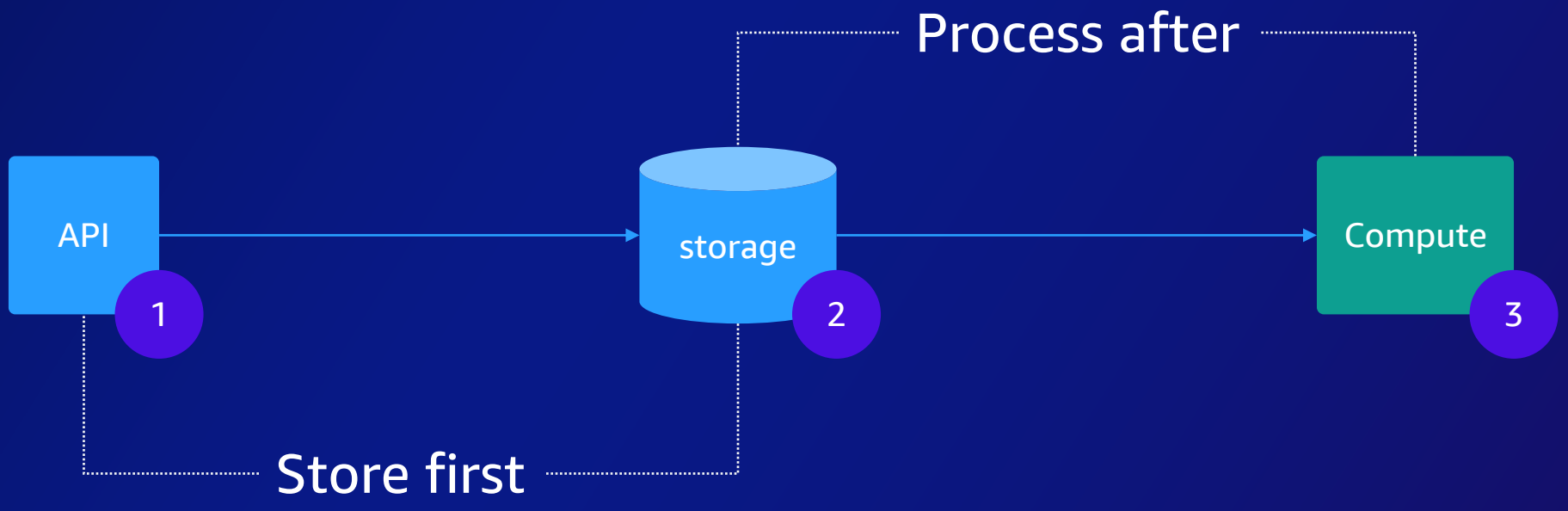
Design considerations – storage first pattern

Thinking synchronously

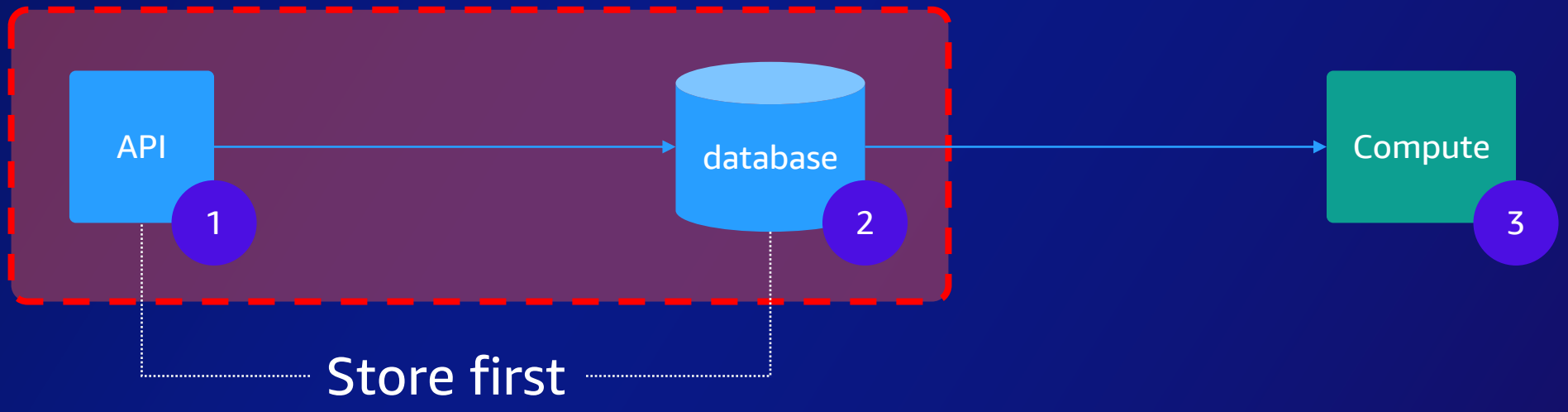


If something is going to go wrong, this is the most probable place

Thinking asynchronously



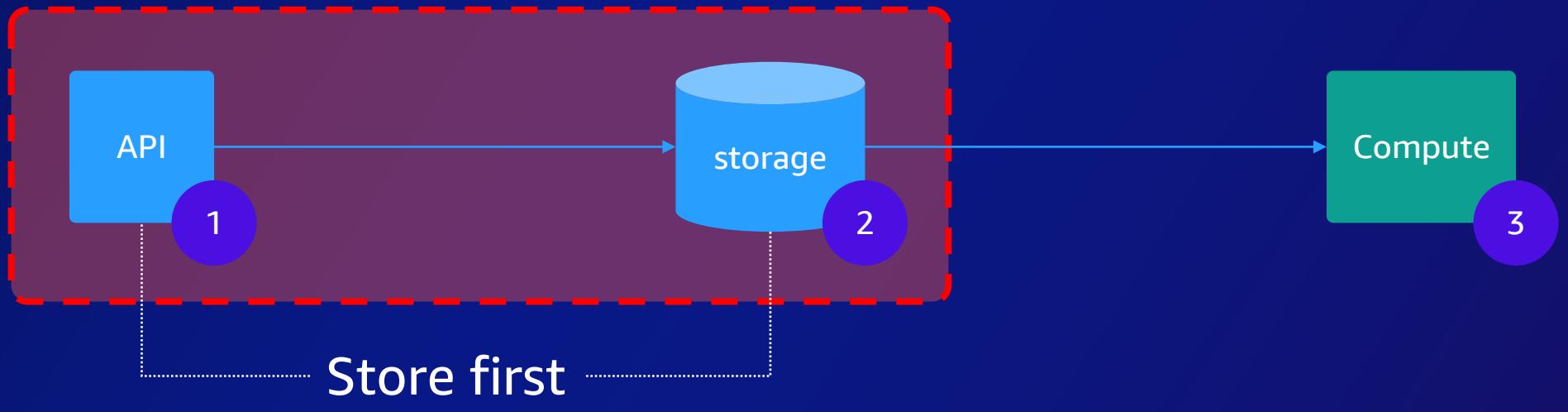
Thinking asynchronously



Greater reliability

Data is stored before my code gets a hold of it

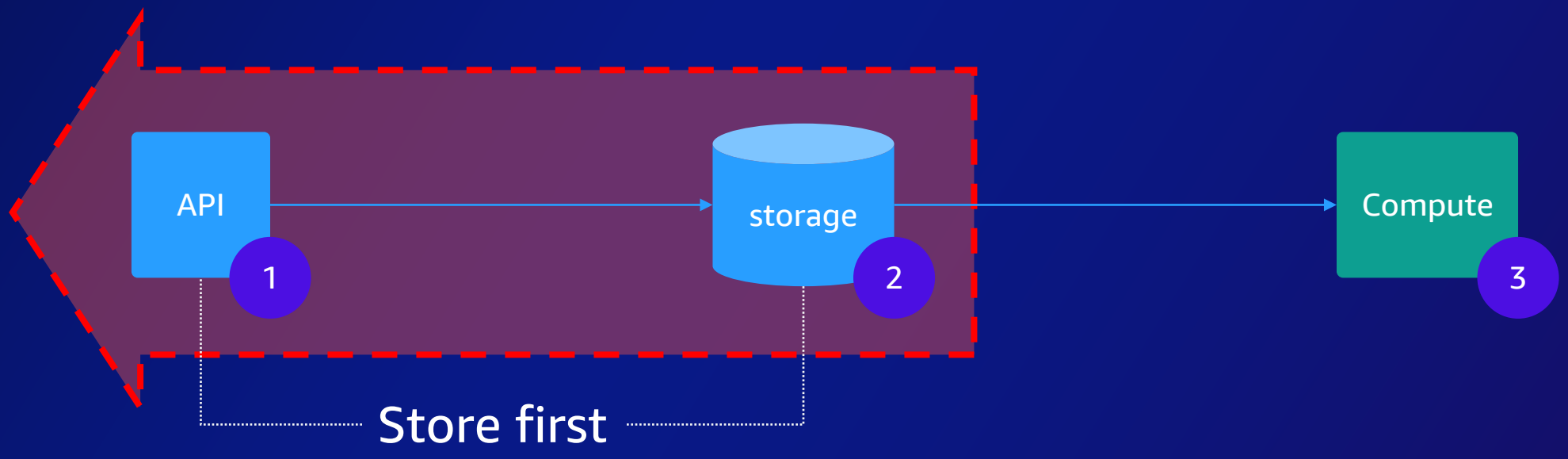
Thinking asynchronously



Less code

Reduce code through direct service integrations

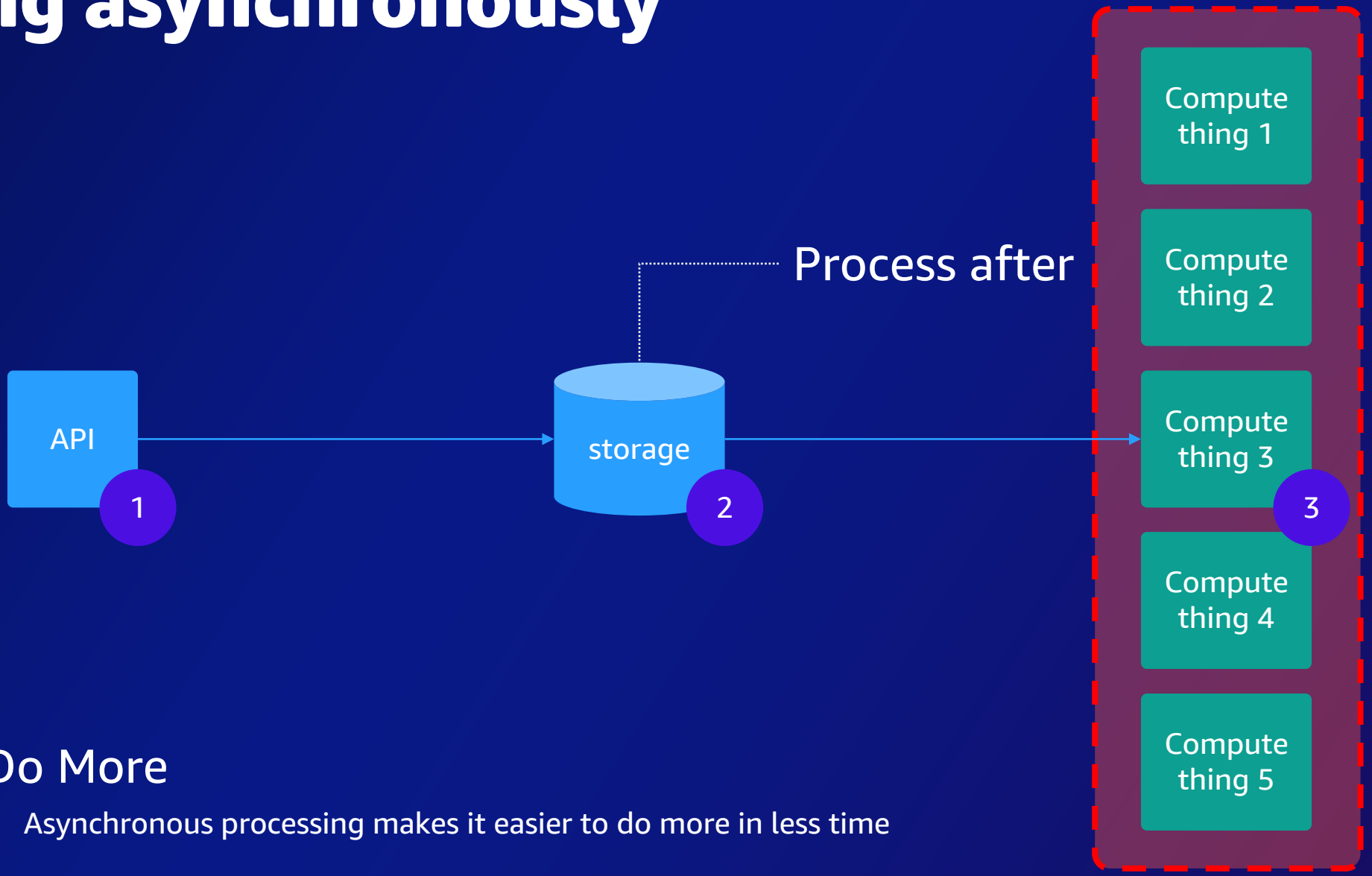
Thinking asynchronously



Faster response time

Client receives receives acknowledgement and can poll for further data if needed.

Thinking asynchronously



Do More

Asynchronous processing makes it easier to do more in less time

skillbuilder.aws

Your time is now

Build in-demand cloud skills your way

aws

What is an Event-Driven Architecture?
https://aws.amazon.com/event-driven-architecture/

Choosing between messaging services for serverless applications.
https://aws.amazon.com/blogs/compute/choosing-between-messaging-services-for-serverless-applications/

Building Event Driven Architectures.
https://serverlessland.com/

Get started on serverless training (Free !)
https://aws.amazon.com/training/learn-about/serverless/

Serverlesspresso Workshop
https://workshop.serverlesscoffee.com/

AWS re:Invent 2022 - Keynote with Dr. Werner Vogels (the world is asynchronous)
https://www.youtube.com/watch?v=RfvL_423a-I

Thinking asynchronously
https://www.youtube.com/watch?v=2Dp8KFLK4Vg

New Blogs Videos Learn Events Patterns About Search Search

Welcome to **Serverless** Land

This site brings together all the latest blogs, videos, and training for AWS Serverless. Learn to use and build apps that scale automatically on low-cost, fully-managed serverless architecture.

Learn More

For more serverless learning resources, visit:
https://serverlessland.com

Thank you!

Please complete the session survey in the mobile app

Michael Hume
mjhume@amazon.co.uk

Kevin Azijn
kevazijn@amazon.com