AWS re:Invent

DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

SVS339

Building event-driven architectures using Amazon ECS with AWS Fargate

Eric Johnson

Principal Developer Advocate

AWS

Uma Ramadoss

she/her

Principal Specialist Solutions Architect

AWS

Agenda

01 Event-driven architecture (EDA)

02 Integration patterns

03 What is AWS Fargate?

04 EDA patterns for AWS Fargate

05 Wrap-up

Event-driven architecture (EDA)

What is an event?

- Events are signals that a system’s state has changed
- Events occur in the past (e.g., ChannelCreated)
- Events cannot be changed (immutable)
- Contract between producer and consumer

```
{
  "source": ”pizza.service.com",
  "detail-type": ”OrderCreated",
  "detail": {
    "metadata": {
      ”idempotency-key”: ”837uy4erje”
    },
    "data": {
      ”channel-id": ”983u4ejrhewio9039oi4kerj",
      "created-at": "2021-11-26T16:05:09-04:00",
      ”name": “noPineappleOnPizza”
      “description”: “All about real pizza”,
      “Tags”: [“Food”,”Proper Pizza”],
      “OrderType”: [“Pickup”],
      ”city”: “Melbourne”
      ”region": "AU",
    }
  }
}
```

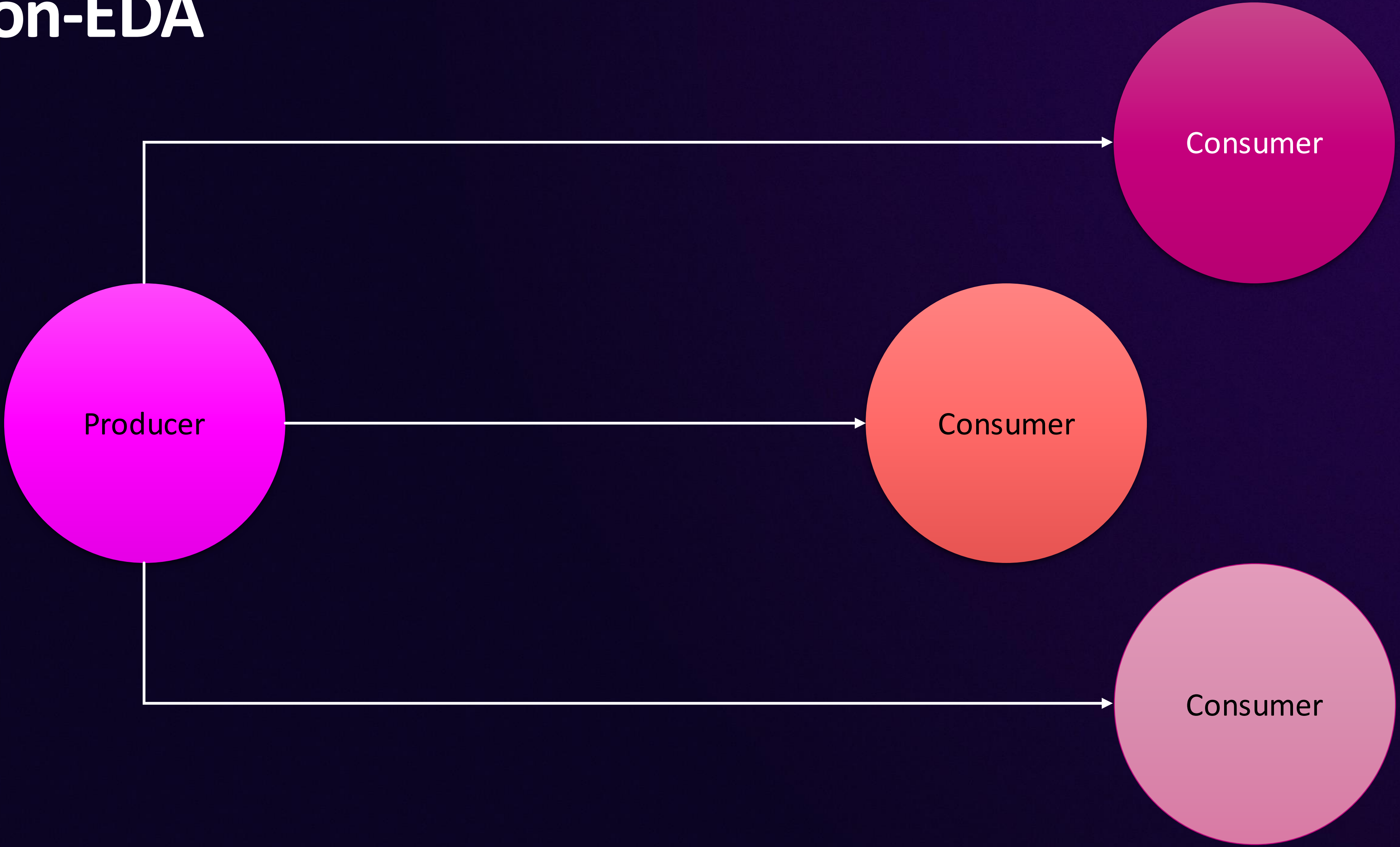
Defining event-driven architecture

Something happens

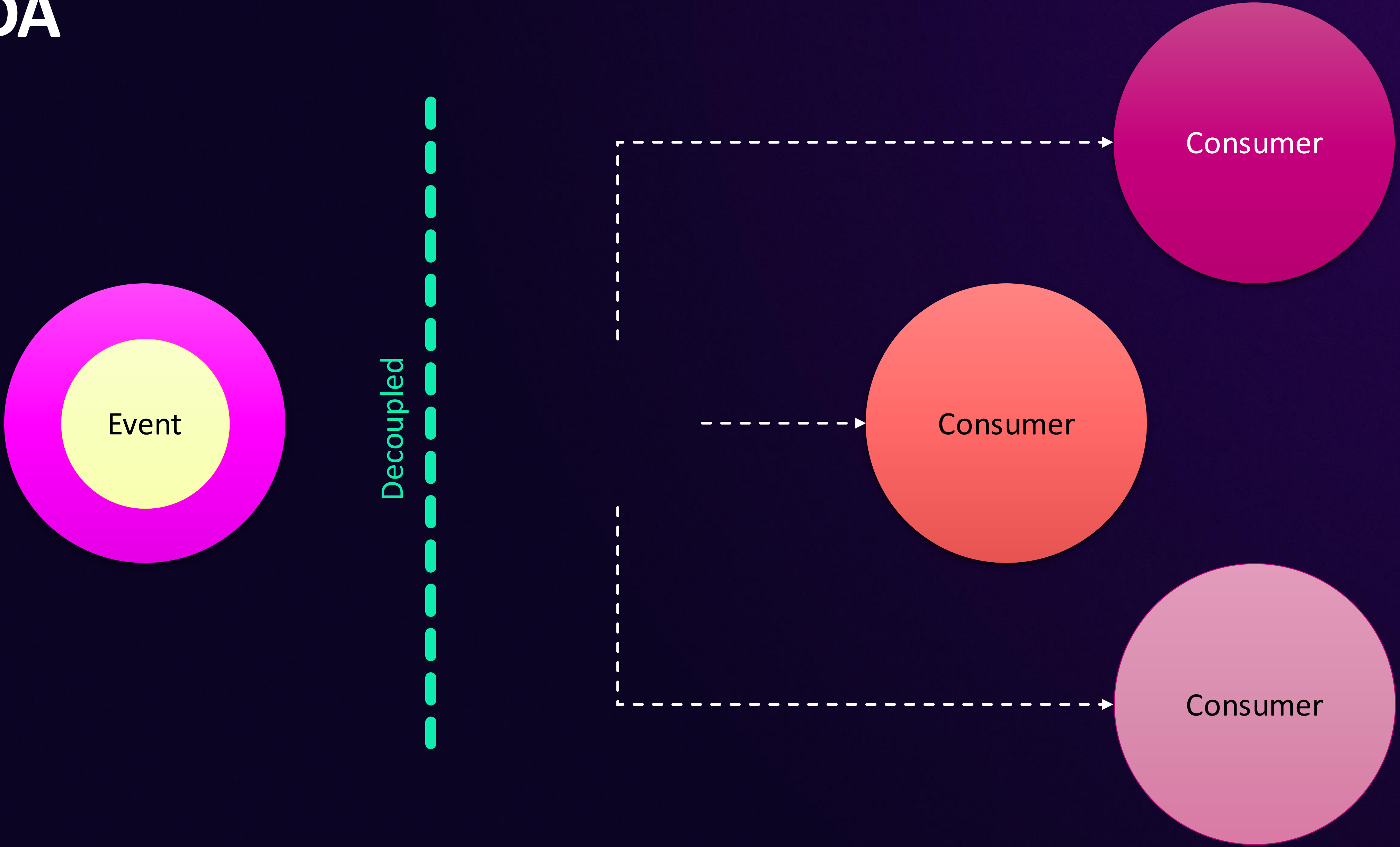
↓

We react

Non-EDA



EDA

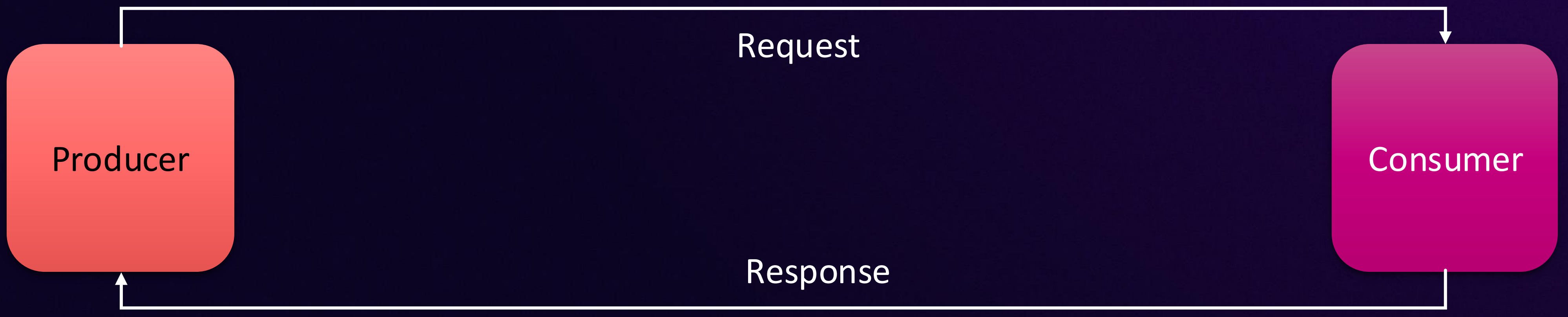


Integration patterns

Synchronous request response model

Advantages

- Low latency
- Simple
- Fail fast



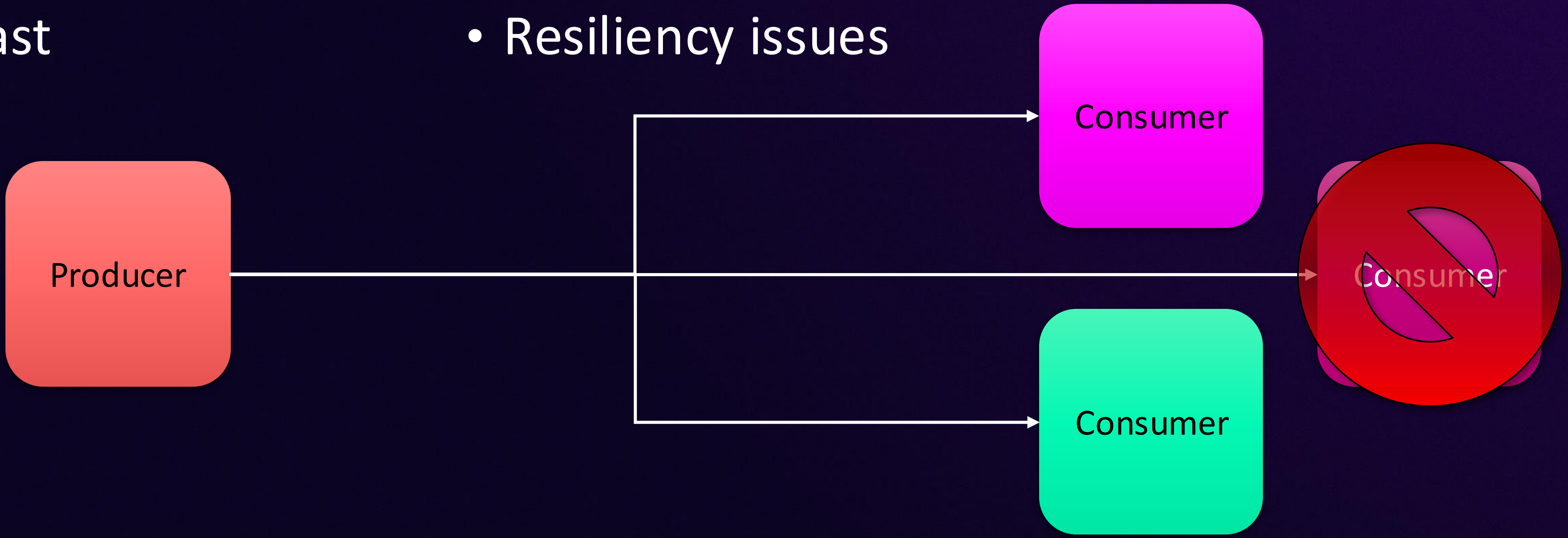
Synchronous request response model

Advantages

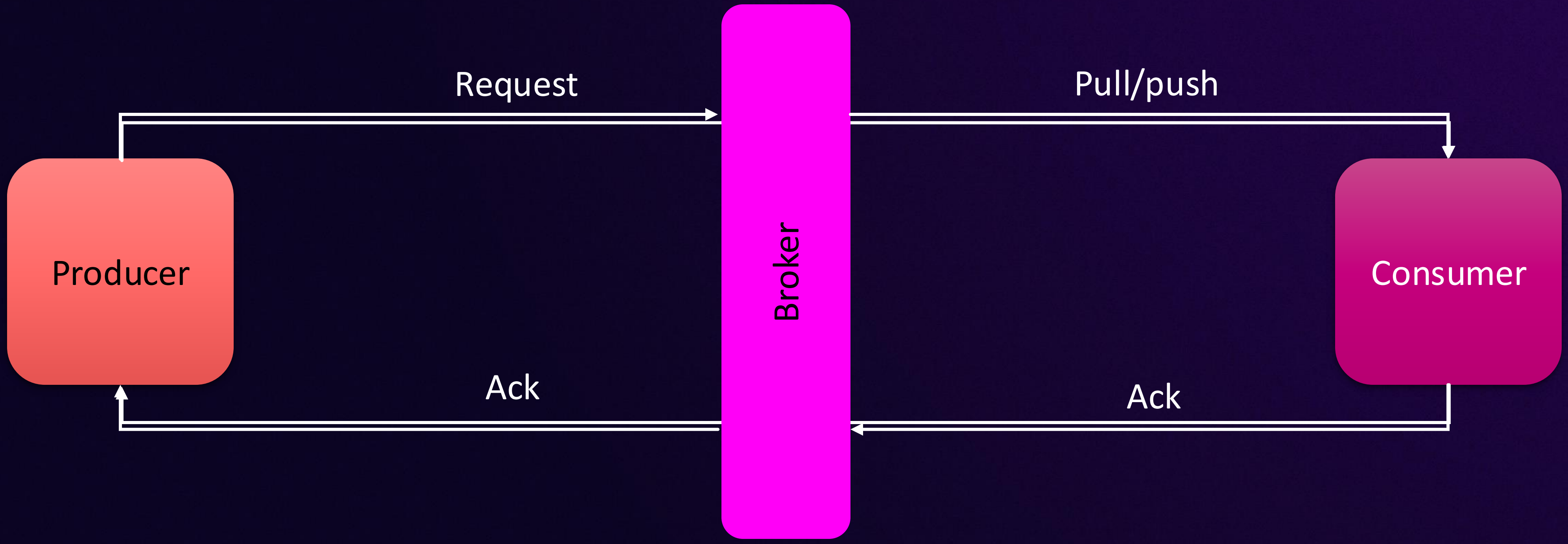
- Low latency
- Simple
- Fail fast

Disadvantages

- Throttling
- Complex
- Resiliency issues



Moving to asynchronous communication

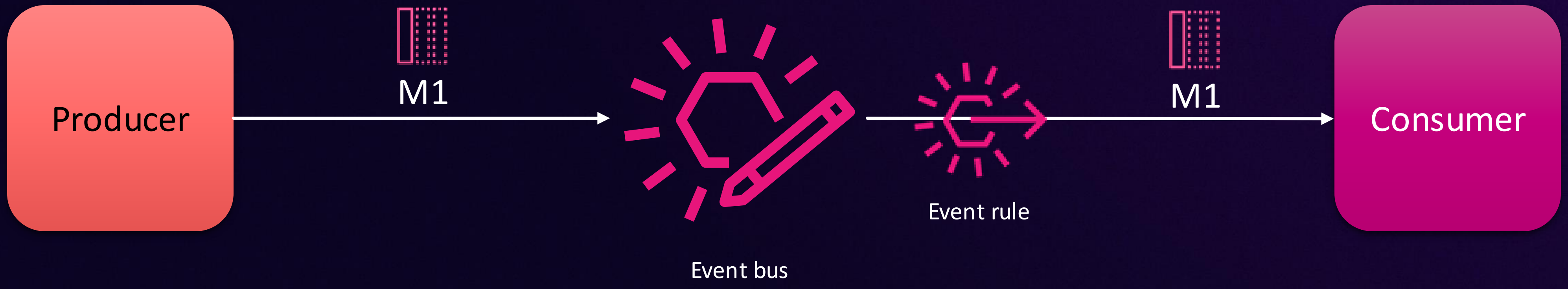


Event routers

- Route one event at a time
- Target multiple endpoints and protocols
- Filter at the router
- Configurable retry
- Configurable DLQ

Event routers – Event buses

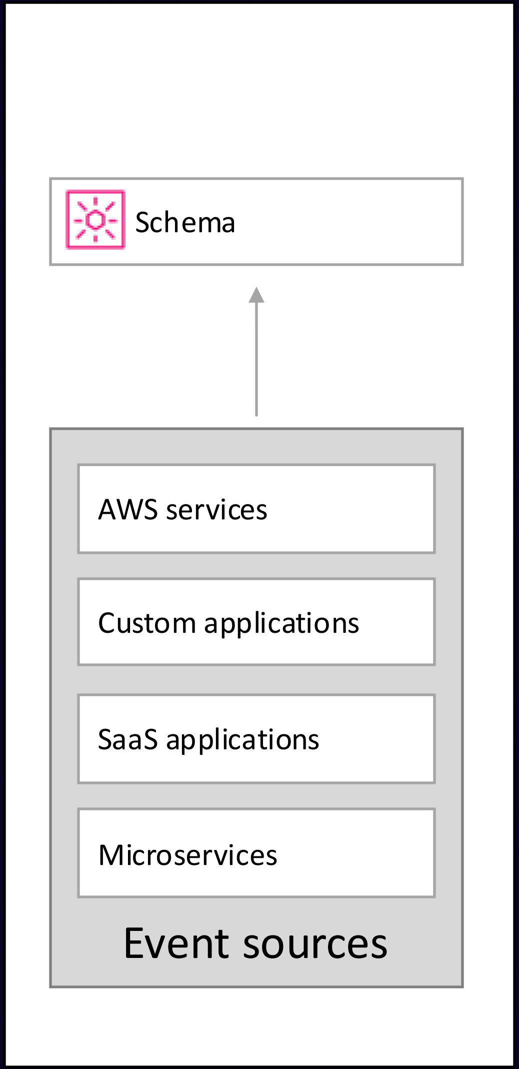
- Reduces location coupling
- Efficient for sender and receiver
- Routing logic maintained in rule



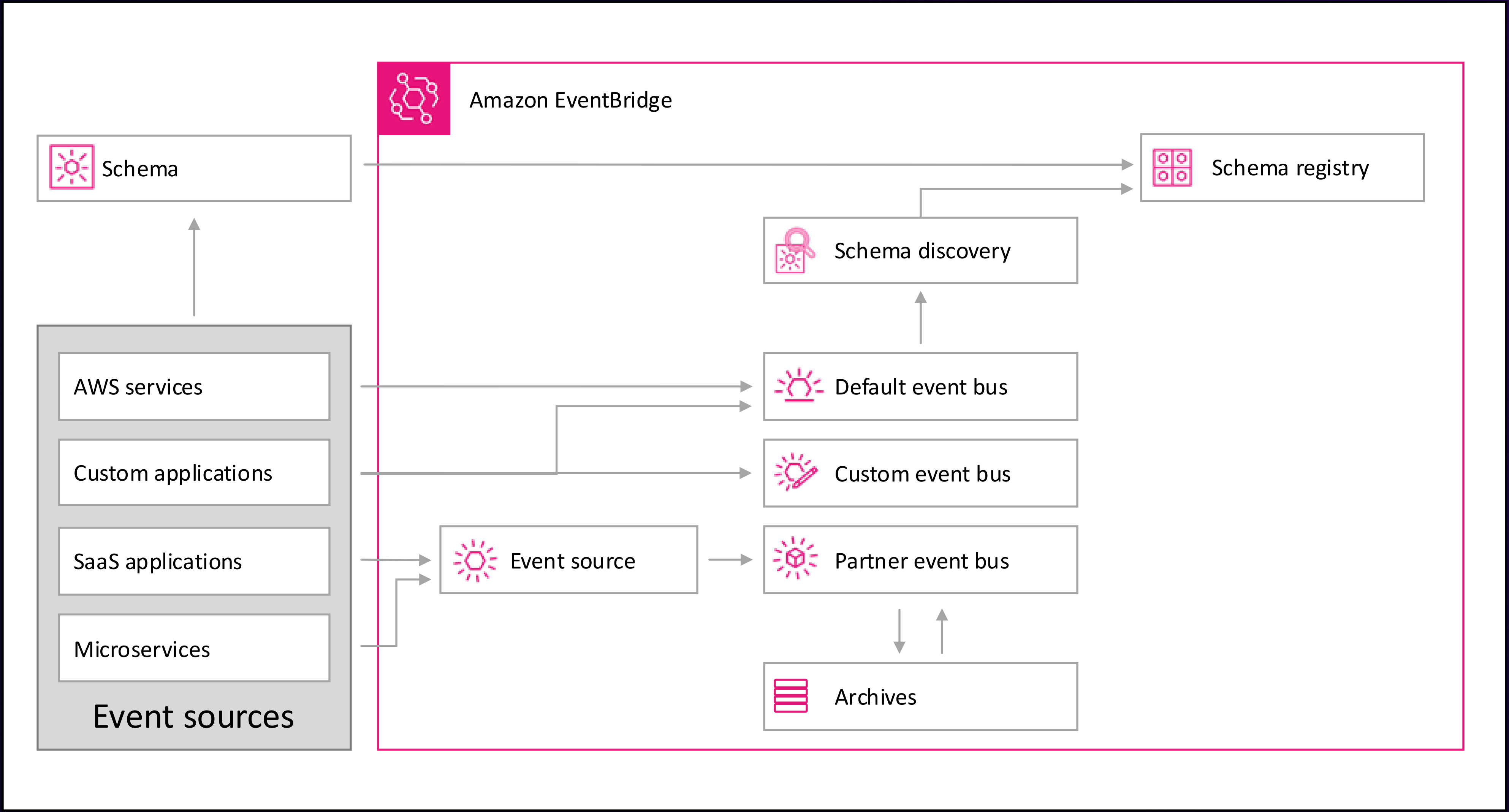
Amazon EventBridge

- Serverless event bus service
- Native integration to AWS and AWS Partner services
- Global endpoints

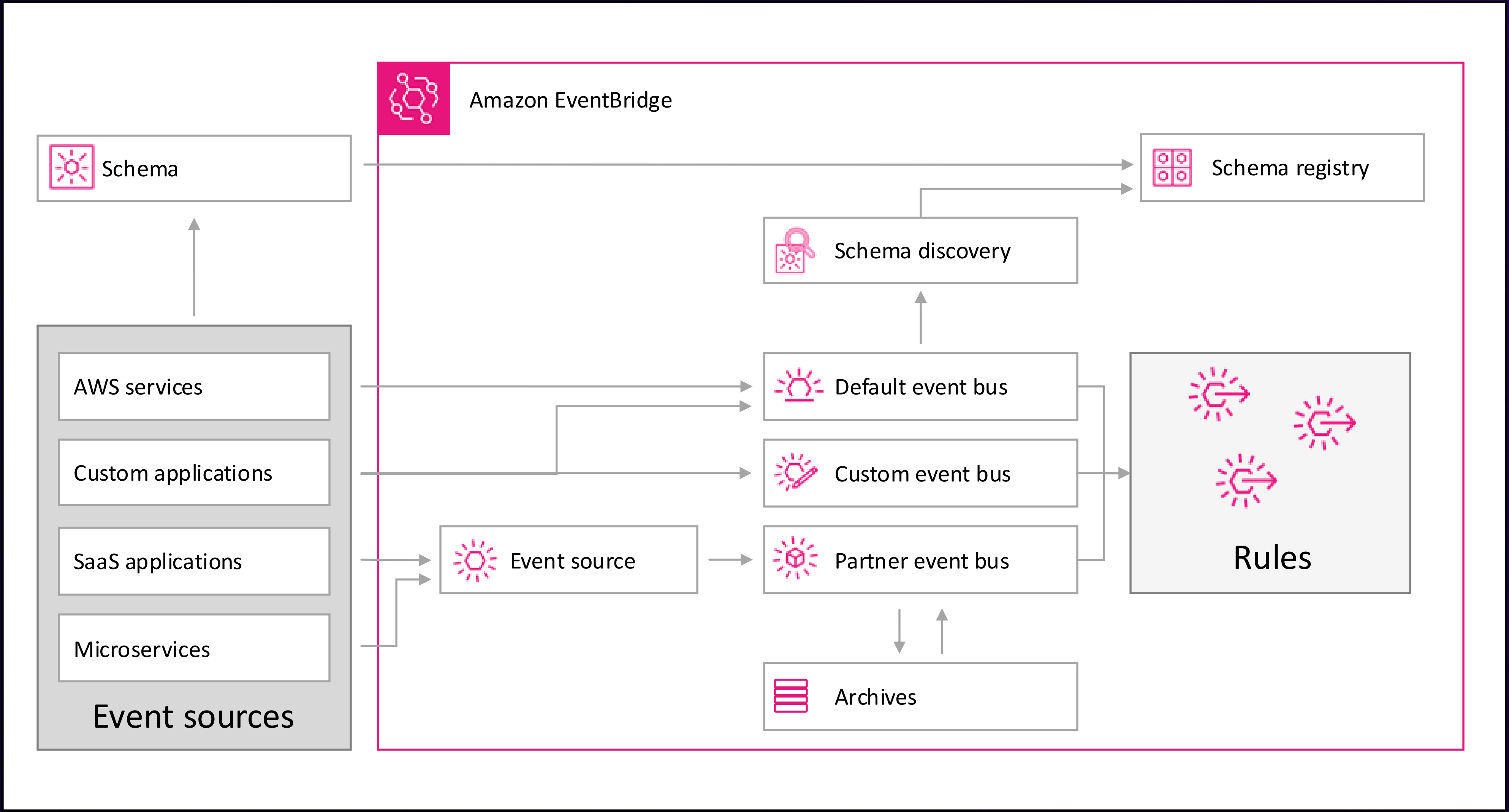
Amazon EventBridge



Amazon EventBridge



Amazon EventBridge



EventBridge content-based routing rules

EventBridge event

```
{
  "source": "com.flix",
  "detail-type": ”ChannelCreated",
  "detail": {
    "metadata": {
      ”idempotency-key”: ”837uy4erje”
    },
    "data": {
      ”channel-id": ”983u4ejrhewio9039oi4kerj",
      "created-at": "2021-11-26T16:05:09-04:00",
      ”name": “noPineappleOnPizza”
      “description”: “All about real pizza”,
      “Tags”: [“Food”,”Proper Pizza”],
      ”city”: “Melbourne”
      ”region": "AU",
    }
  }
}
```

EventBridge content-based routing rules

EventBridge event

```
{
  "source": "com.flix",
  "detail-type": ”ChannelCreated",
  "detail": {
    "metadata": {
      ”idempotency-key”: ”837uy4erje”
    },
    "data": {
      ”channel-id": ”983u4ejrhewio9039oi4kerj",
      "created-at": "2021-11-26T16:05:09-04:00",
      ”name": “noPineappleOnPizza”
      “description”: “All about real pizza”,
      “Tags”: [“Food”,”Proper Pizza”],
      ”city”: “Melbourne”
      ”region": "AU",
    }
  }
}
```

EventBridge rule

```
{
  “source”: [“com.flix”]
  "detail": {
    "data":  {
      ”region": ["AU", "NZ"]
    }
  }
}
```

EventBridge content-based routing rules

EventBridge event

```
{
  "source": "com.flix",
  "detail-type": ”ChannelCreated",
  "detail": {
    "metadata": {
      ”idempotency-key”: ”837uy4erje”
    },
    "data": {
      ”channel-id": ”983u4ejrhewio9039oi4kerj",
      "created-at": "2021-11-26T16:05:09-04:0[illegible]
      ”name": “noPineappleOnPizza”
      “description”: “All about real [illegible]za”,
      “Tags”: [“Food”,”Proper Pi[illegible]],
      ”city”: “Melbourne”
      ”region": "AU",
    }
  }
}
```

EventBridge rule

```
{
  “source”: [“com.flix”]
  "detail": {
    "data":  {
      ”region"[illegible] ["AU", "NZ"]
    }
  }
}
```

EventBridge content-based routing rules

EventBridge event

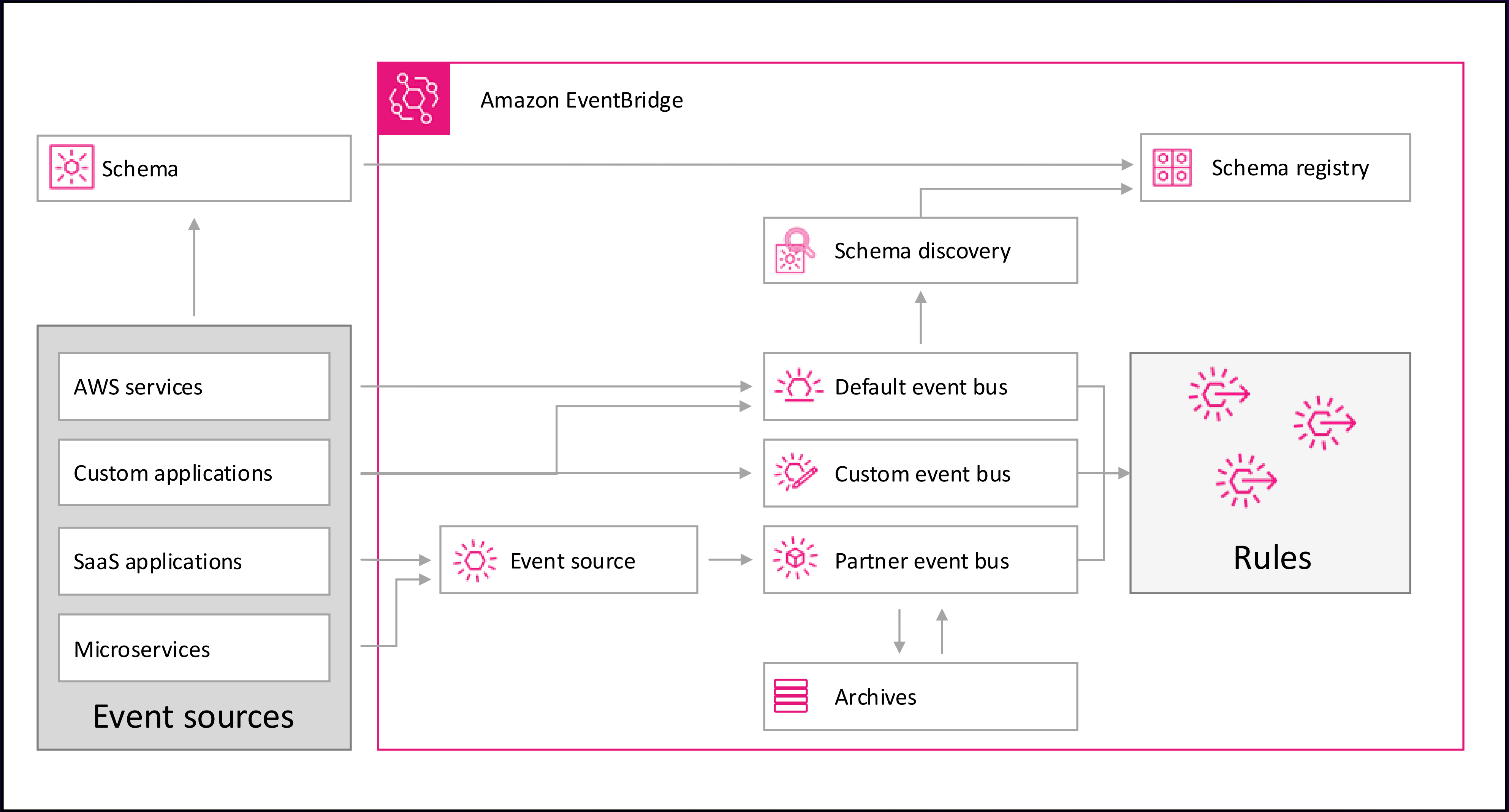
```
{
  "source": "com.flix",
  "detail-type": ”ChannelCreated",
  "detail": {
    "metadata": {
      ”idempotency-key”: ”837uy4erje”
    },
    "data": {
      ”channel-id": ”983u4ejrhewio9039oi4kerj",
      "created-at": "2021-11-26T16:05:09-04:0[illegible]
      ”name": “noPineappleOnPizza”
      “description”: “All about real [illegible]za”,
      “Tags”: [“Food”,”Proper Pi[illegible]],
      ”city”: “Melbourne”
      ”region": "AU",
    }
  }
}
```

EventBridge rule

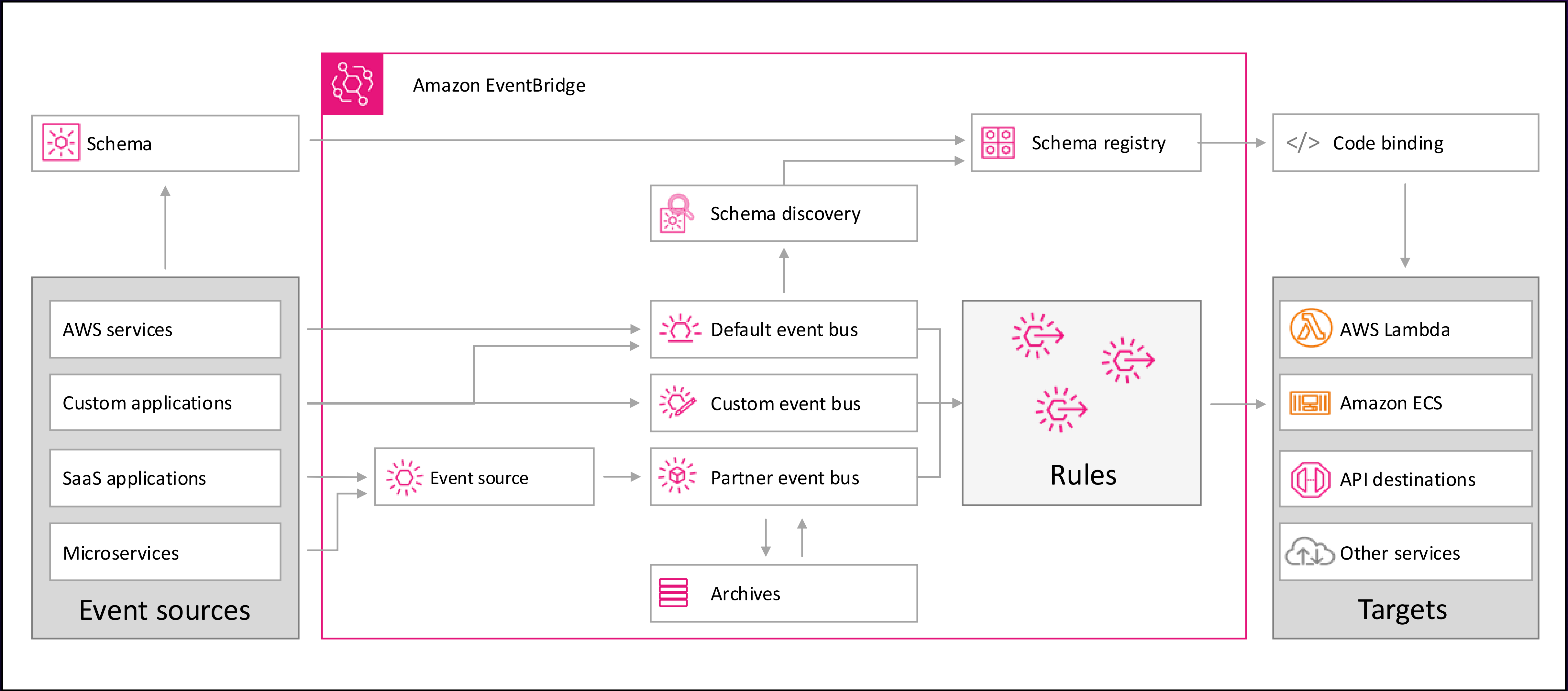
```
{
  “source”: [“com.flix”]
  "detail": {
    "data":  {
      ”region" [illegible] ["AU", "NZ"]
    }
  }
}
```

MATCH

Amazon EventBridge

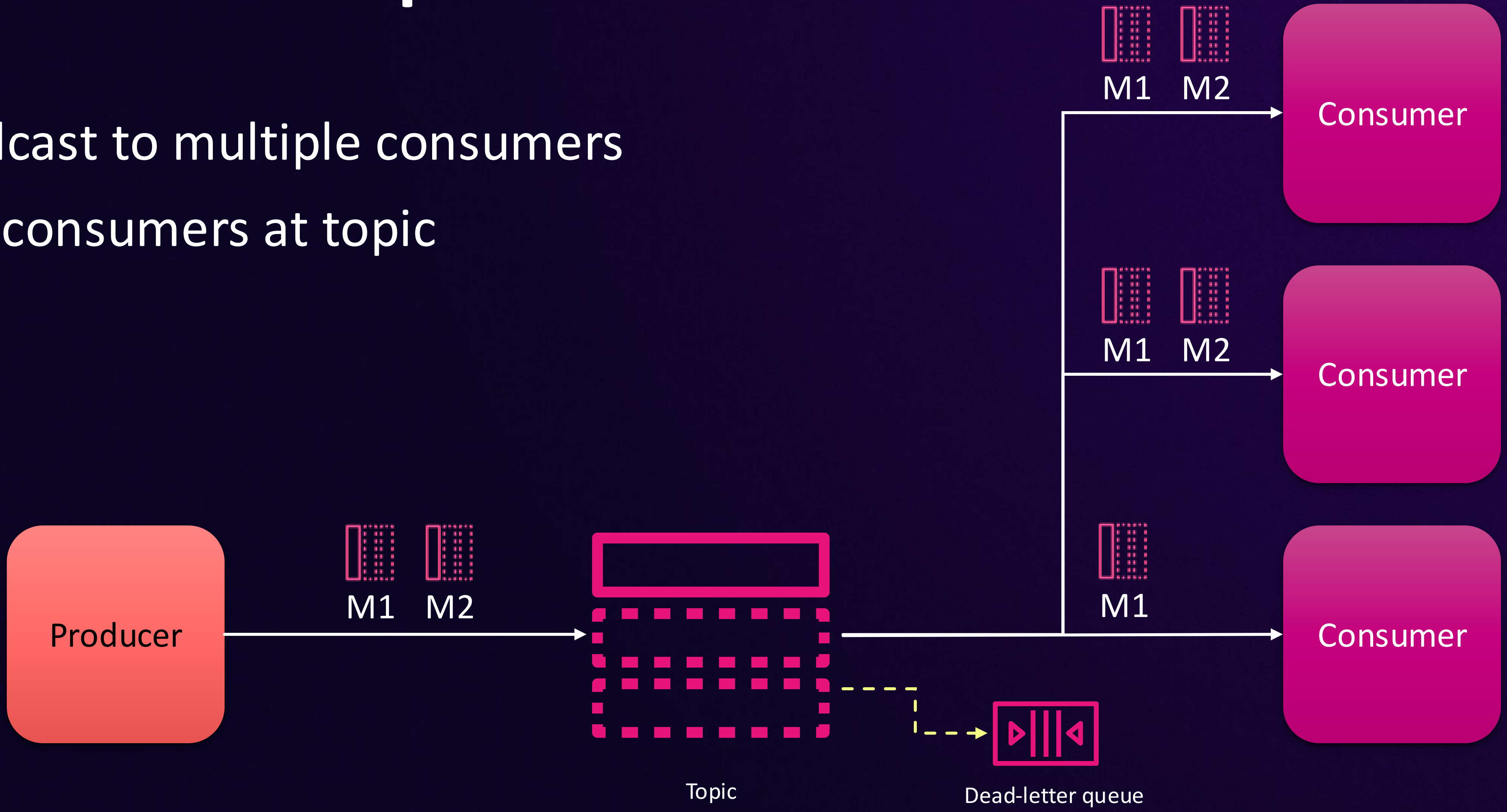


Amazon EventBridge



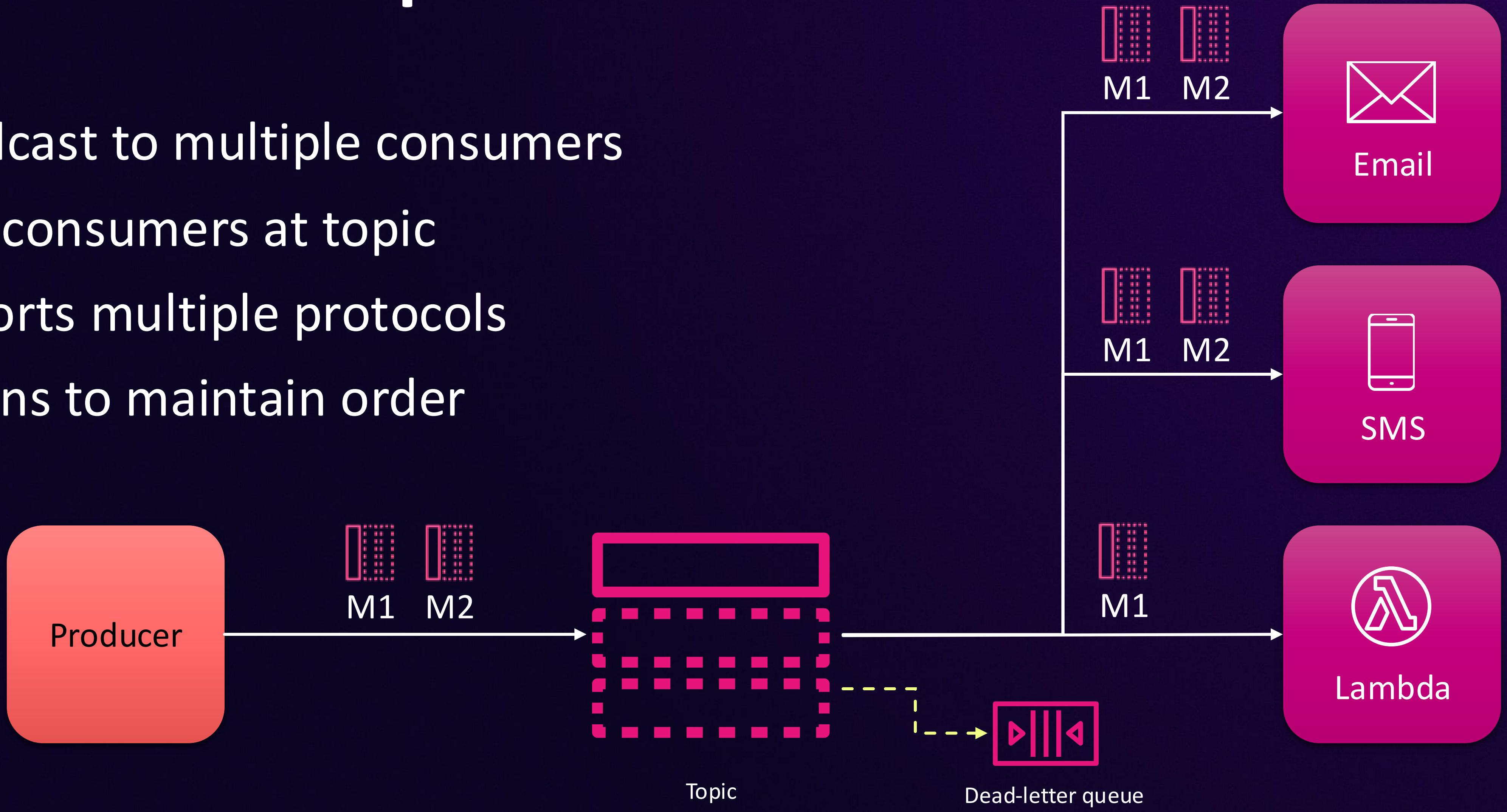
Event routers – Topics

- Broadcast to multiple consumers
- Filter consumers at topic



Event routers – Topics

- Broadcast to multiple consumers
- Filter consumers at topic
- Supports multiple protocols
- Options to maintain order

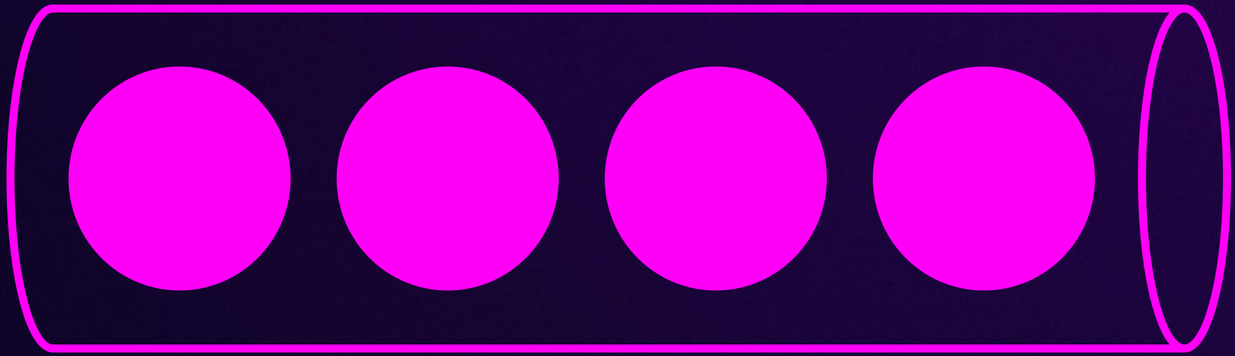


Amazon SNS

- Fully managed pub/sub service for A2A and A2P messaging
- Standard topics support a nearly unlimited number of messages per second
- Each topic supports up to 12.5 million subscriptions
- FIFO topics when strict ordering is required

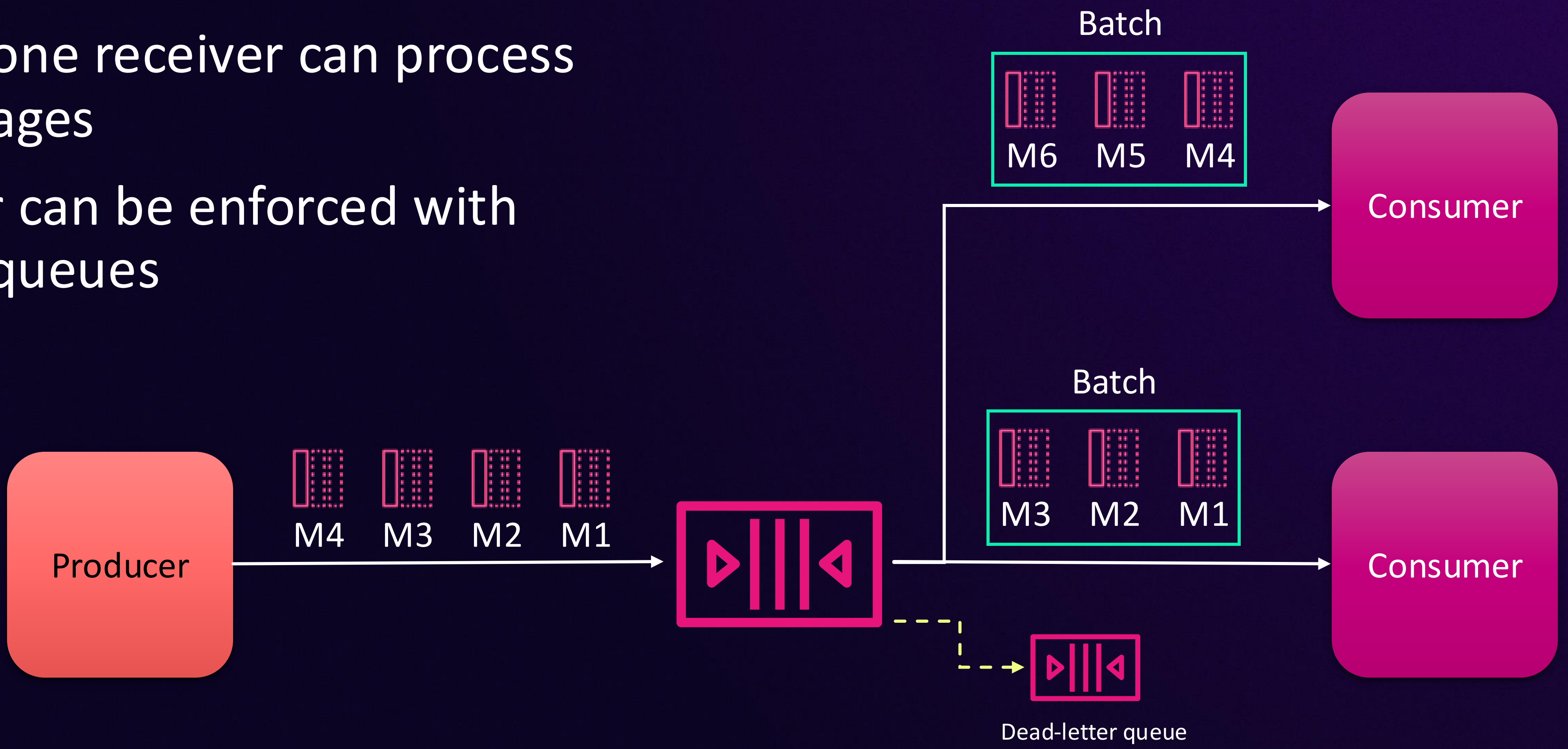
Event stores

- Enable receiver to process events in batches
- Control batch size and process timing
- Options for controlling order
- Configurable retry
- Configurable DLQ



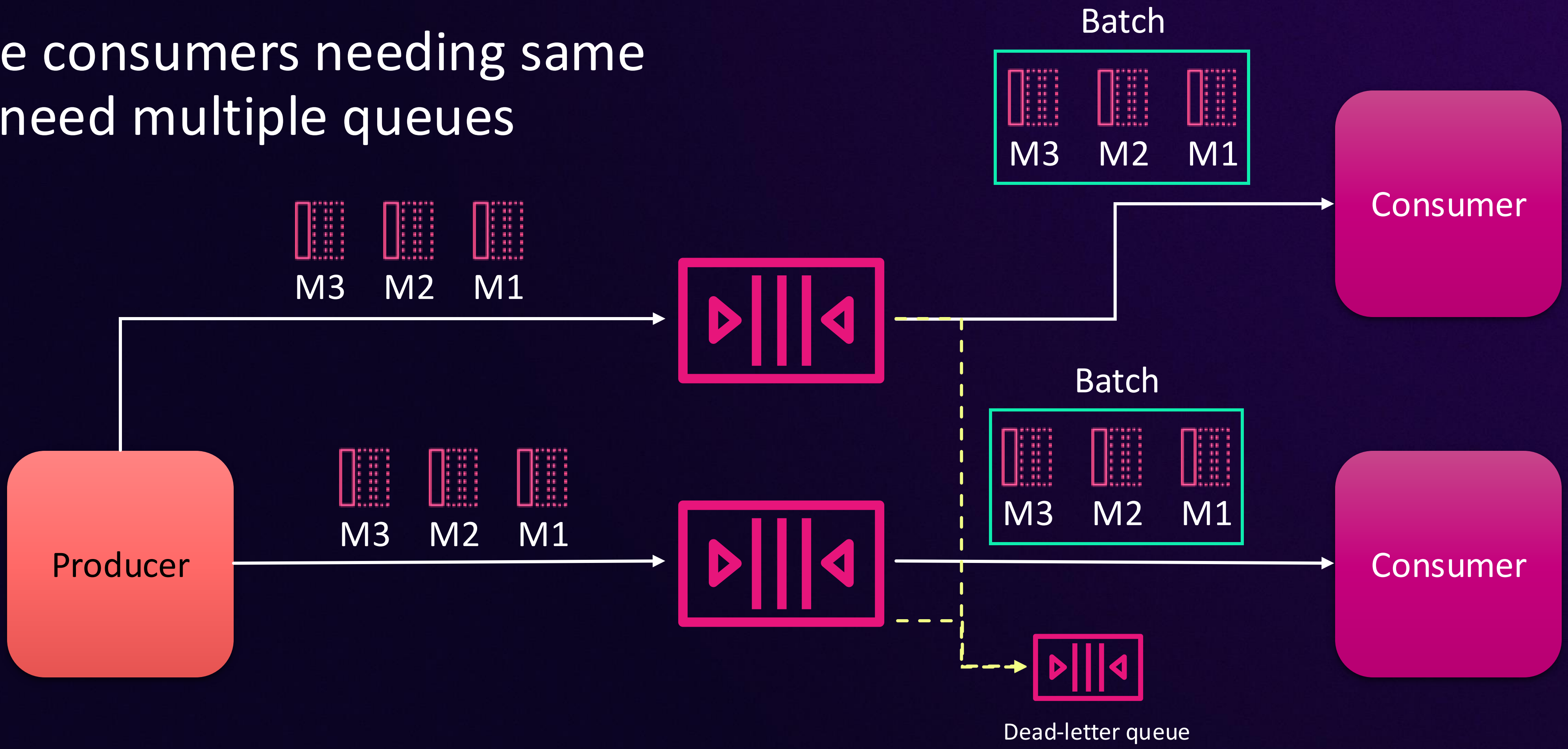
Event store – Queues

- Only one receiver can process messages
- Order can be enforced with FIFO queues



Event store – Queues

Multiple consumers needing same events need multiple queues



Amazon SQS

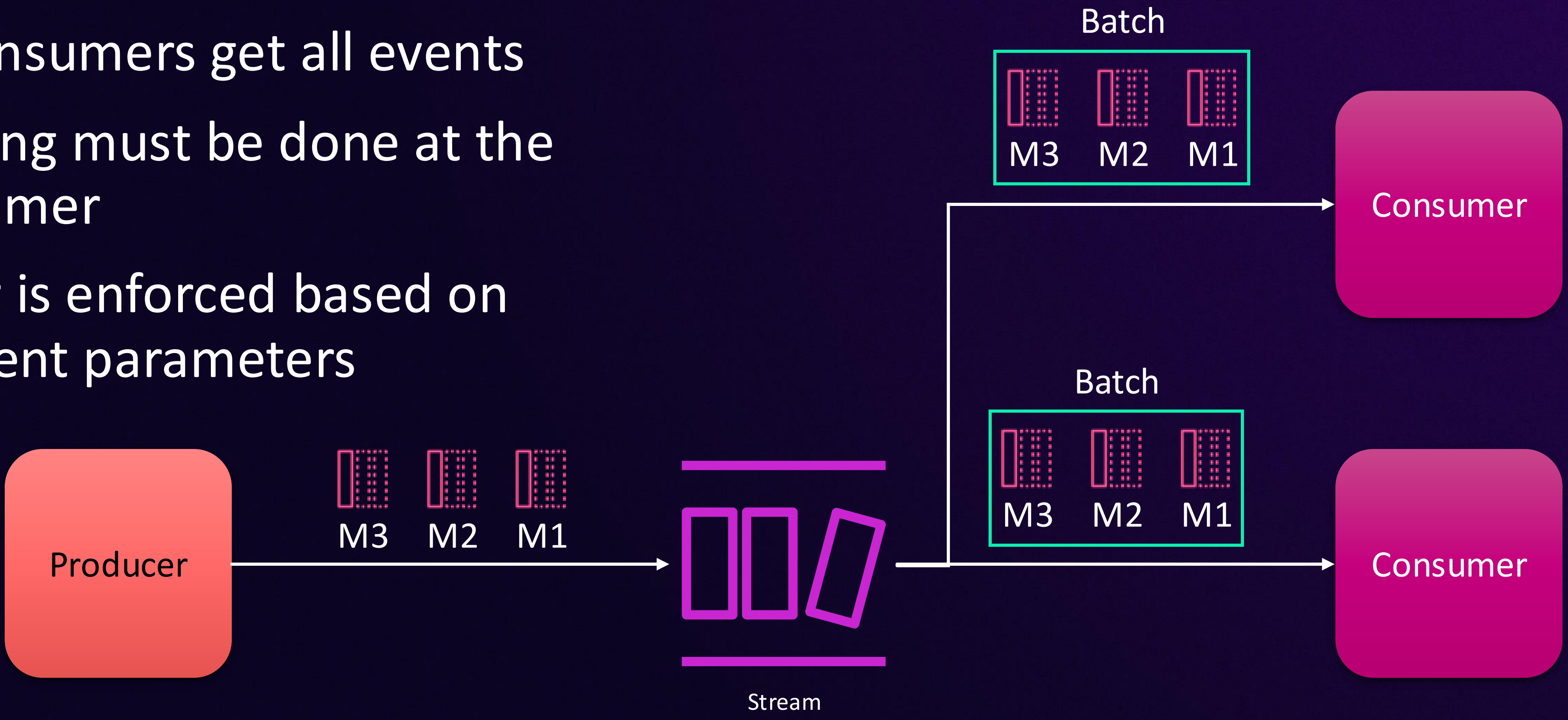
- Fully managed message queue
- Scales almost infinitely
- Simple, easy-to-use API
- DLQ support
- Standard and FIFO options

Amazon MQ

- ActiveMQ and RabbitMQ broker engine options
- AWS manages the provisioning, setup, and maintenance of message brokers
- Connects to current applications with industry-standard APIs and protocols

Event store – Stream

- All consumers get all events
- Filtering must be done at the consumer
- Order is enforced based on different parameters



Amazon Kinesis Data Streams

- Serverless streaming data service
- GB/sec streaming
- Automatic provisioning with on demand
- Built-in integration with other AWS services

Amazon MSK

- Fully managed Apache Kafka
- AWS manages the provisioning, setup, and high availability of Apache Kafka service
- Pay-as-you-go pricing

What is AWS Fargate?

Amazon ECS on AWS Fargate

Serverless

Managed by AWS; no EC2 instances to provision, scale, or manage – simple operations model

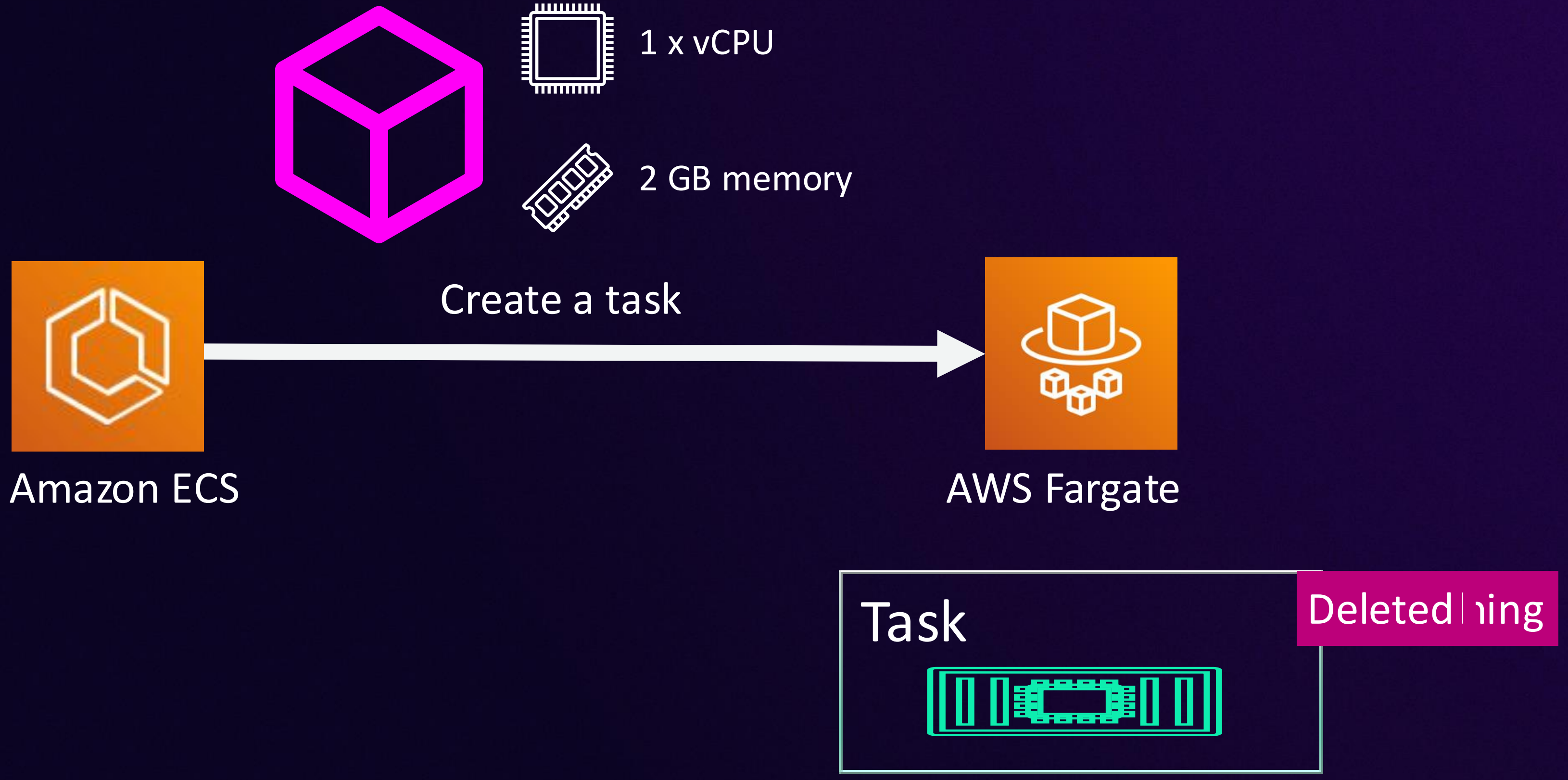
Secure

Isolated, patched, and compliant for running the most sensitive workloads

Savings

Deploy quickly, scale efficiently, and automatically allocate compute as needed

Running as standalone task

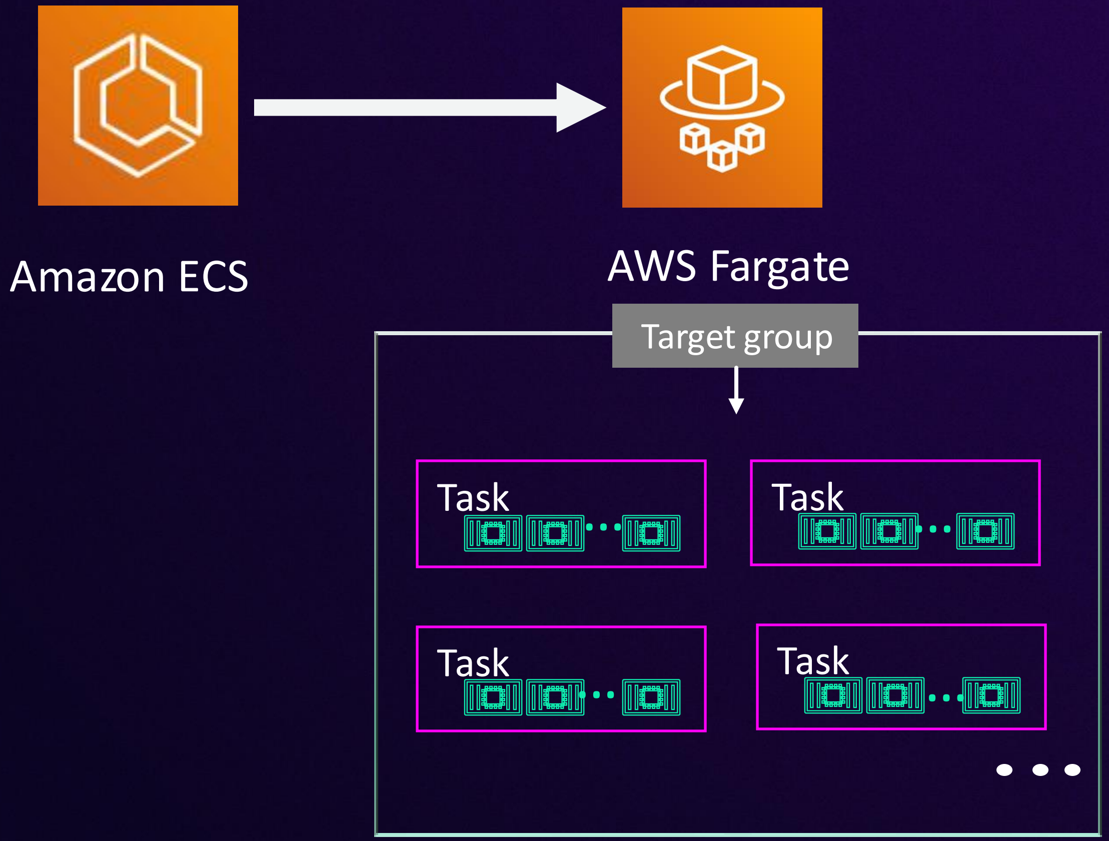


Running as service

1 x vCPU

2 GB memory

Here is my application container, and I want to launch it as a service with 4 copies.



Why AWS Fargate for EDA?

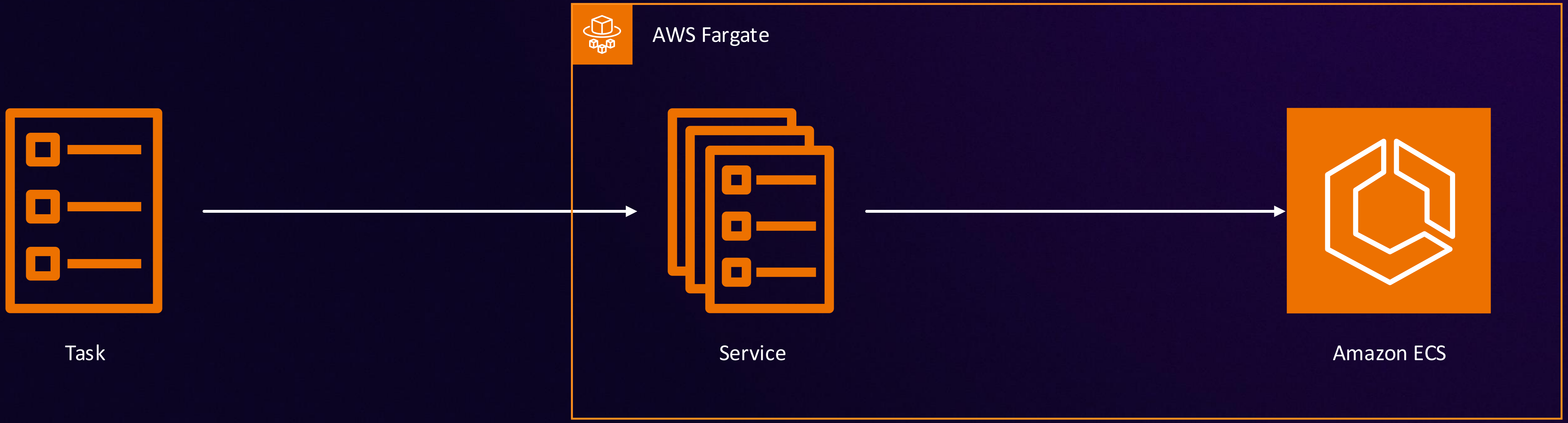
How AWS Fargate enables building EDA

Scalability	Microservices	Integrations	Cost savings
Auto scaling with target tracking	Flexible config, containerized, frequent deployments	Native integration with Amazon EventBridge, AWS Step Functions …	Integrated with Spot for cost savings

EDA patterns for AWS Fargate

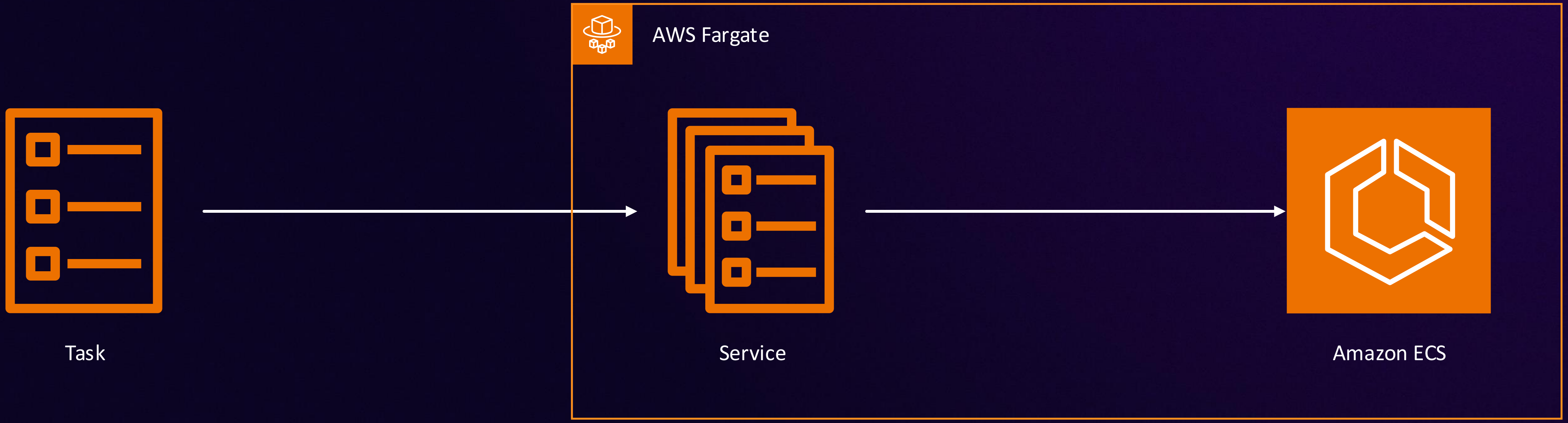
Traditional Amazon ECS on AWS Fargate

Provides persistent compute service that runs **all the time**



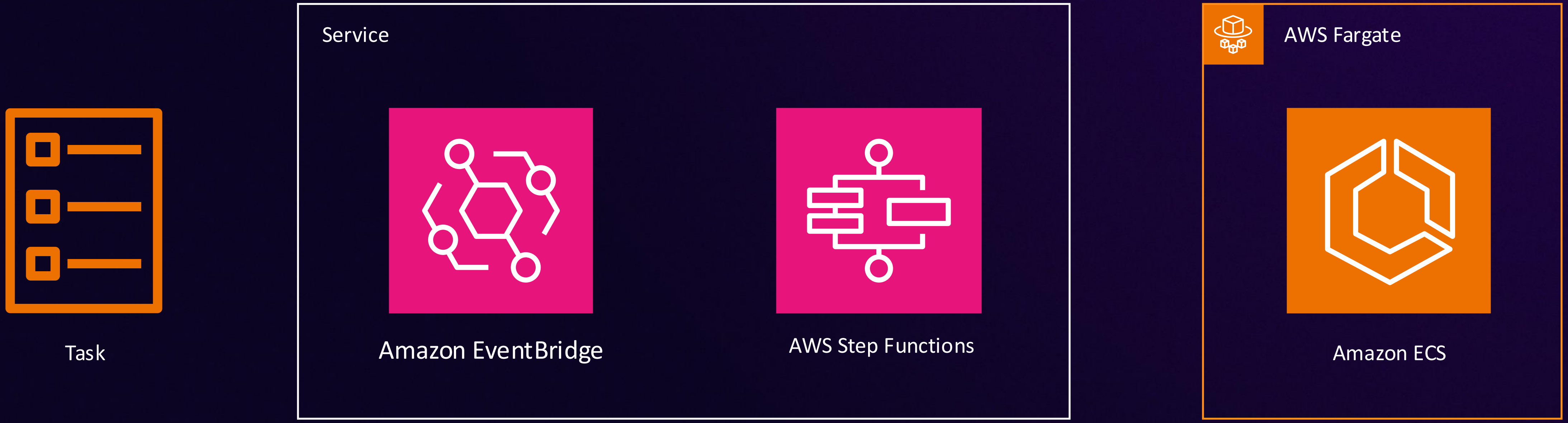
Traditional Amazon ECS on Fargate

What if I only need it sometimes, like when an event happens?



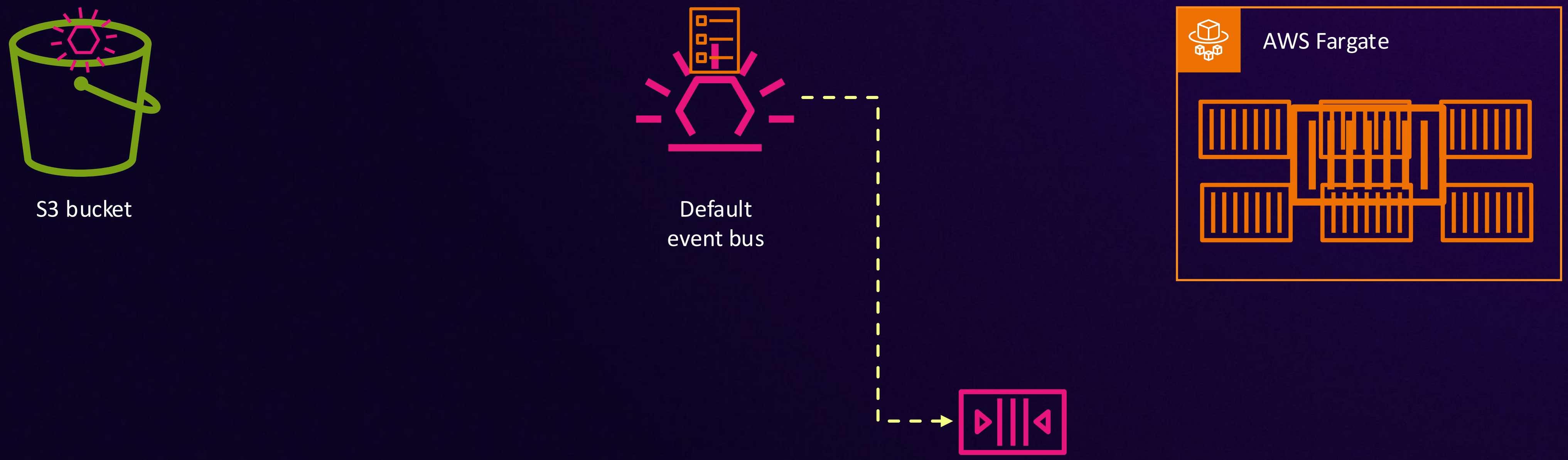
Event-based Amazon ECS on Fargate

I swap out the service to run tasks as needed.

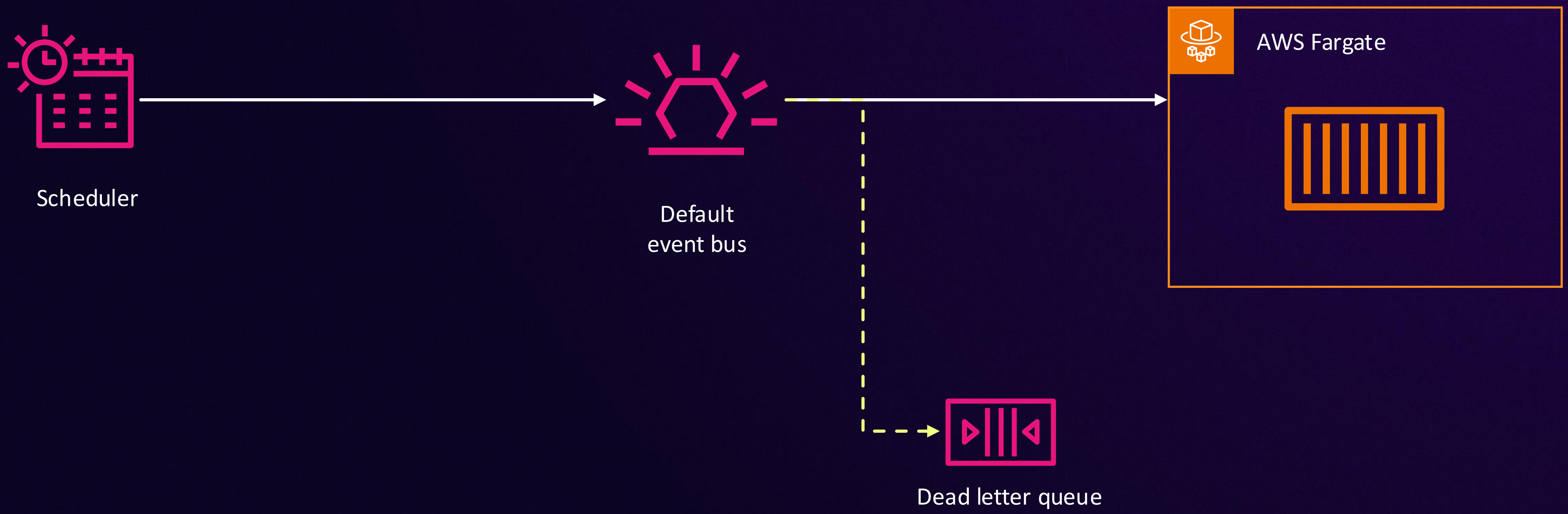


Patterns with standalone tasks

Large object processing



Scheduled processing



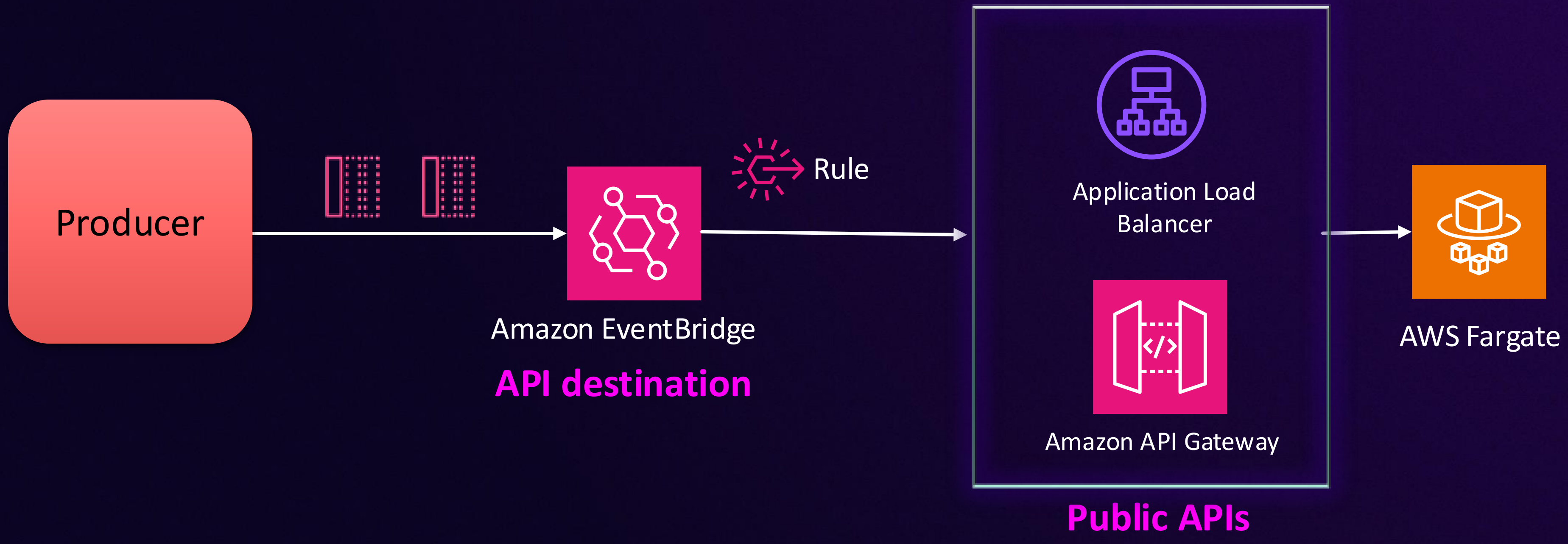
Considerations for stand alone tasks

- Use when AWS Lambda is not a good fit
  - When you need more than 15 minutes
  - When you need more CPU options
- Task failures need to be managed separately
- Use a DLQ to prevent data loss
- Burst can cause scaling issues with task

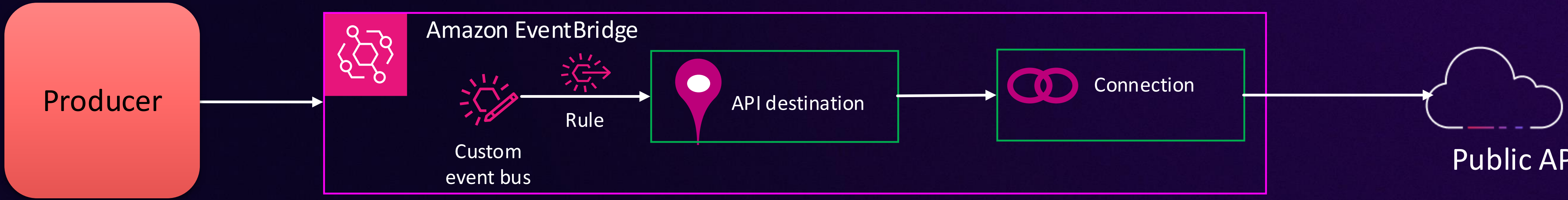
Patterns with long-running services

API pattern – Public API

Removed shipping text



API pattern – Public API with API destination



EventBridge content-based routing rules

EventBridge event

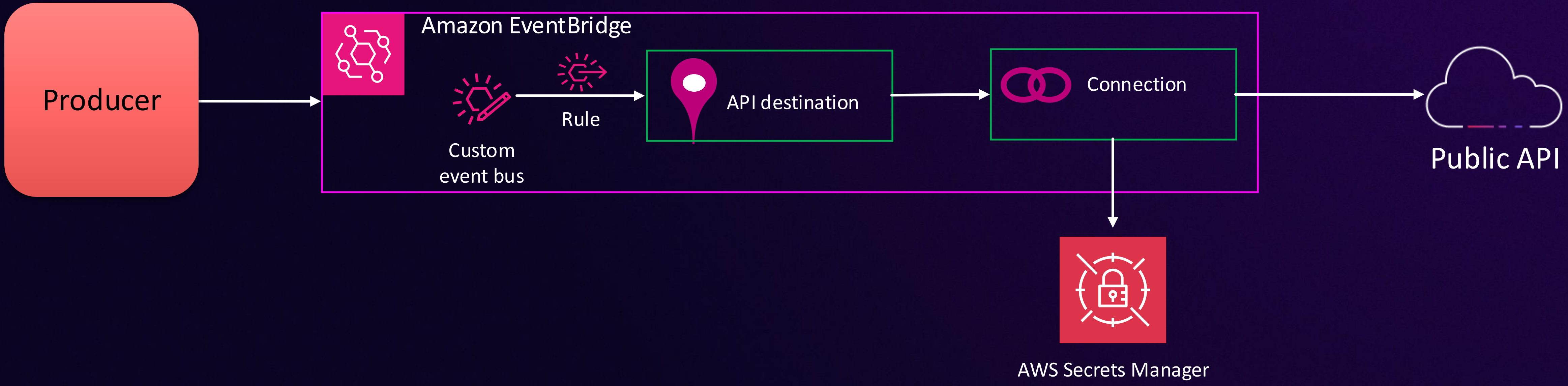
```
{
  "source": "com.flix",
  "detail-type": ”ChannelCreated",
  "detail": {
    "metadata": {
      ”idempotency-key”: ”837uy4erje”
    },
    "data": {
      ”channel-id": ”983u4ejrhewio9039o[illegible]j",
      "created-at": "2021-11-26T16:0[illegible]-04[illegible],
      ”name": “noPineappleOnPizza”
      “description”: “All about rea[illegible]zza”,
      “Tags”: [“Food”,”Proper [illegible]a”],
      ”city”: “Melbourne”
      ”region": ”Asia[illegible]
    }
  }
}
```

EventBridge rule

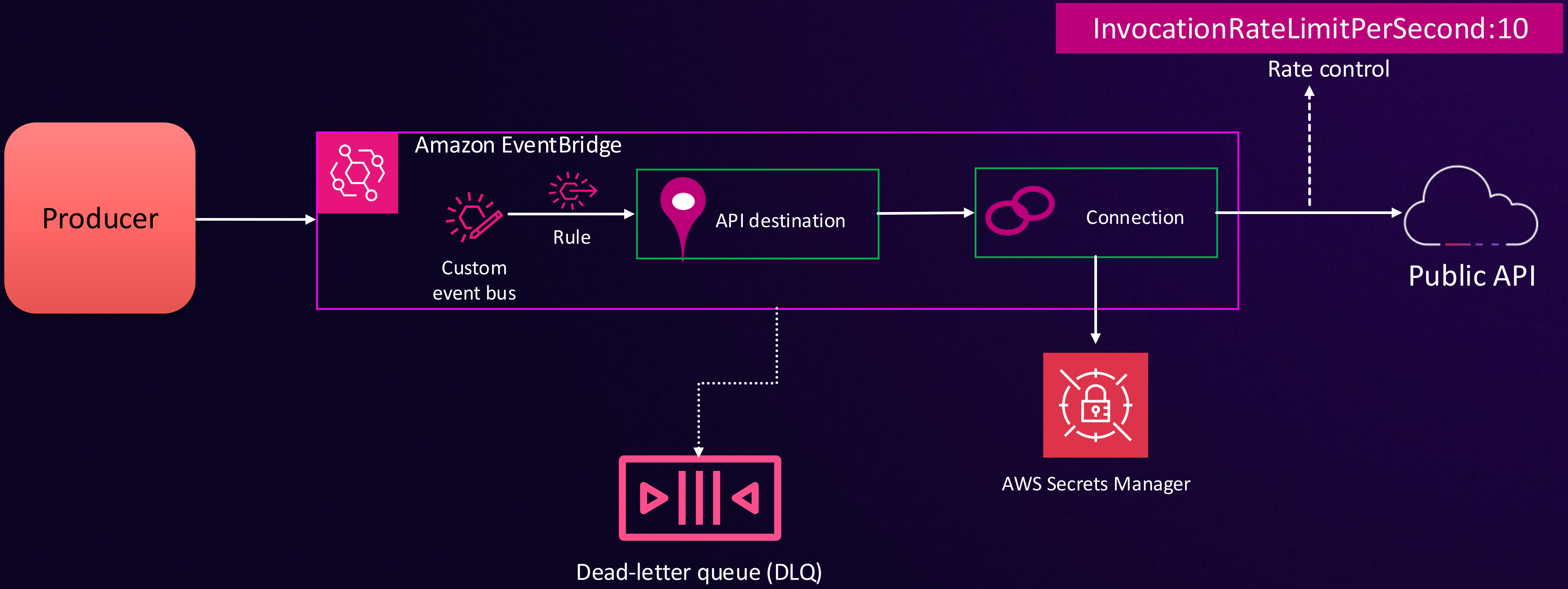
```
{
  “source”: [“com.flix”]
  "detail": {
    "data":  {
      ”name": ["noPineappleOnPizza"]
       “region”:[“Asia”]
    [illegible]
  [illegible]
}
```

MATCH

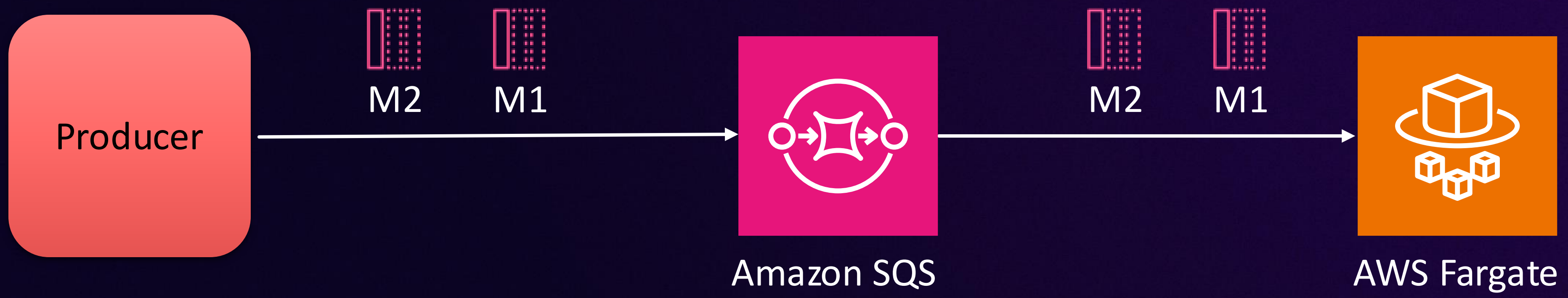
API pattern – Public API with API destination



API pattern – Public API with API destination

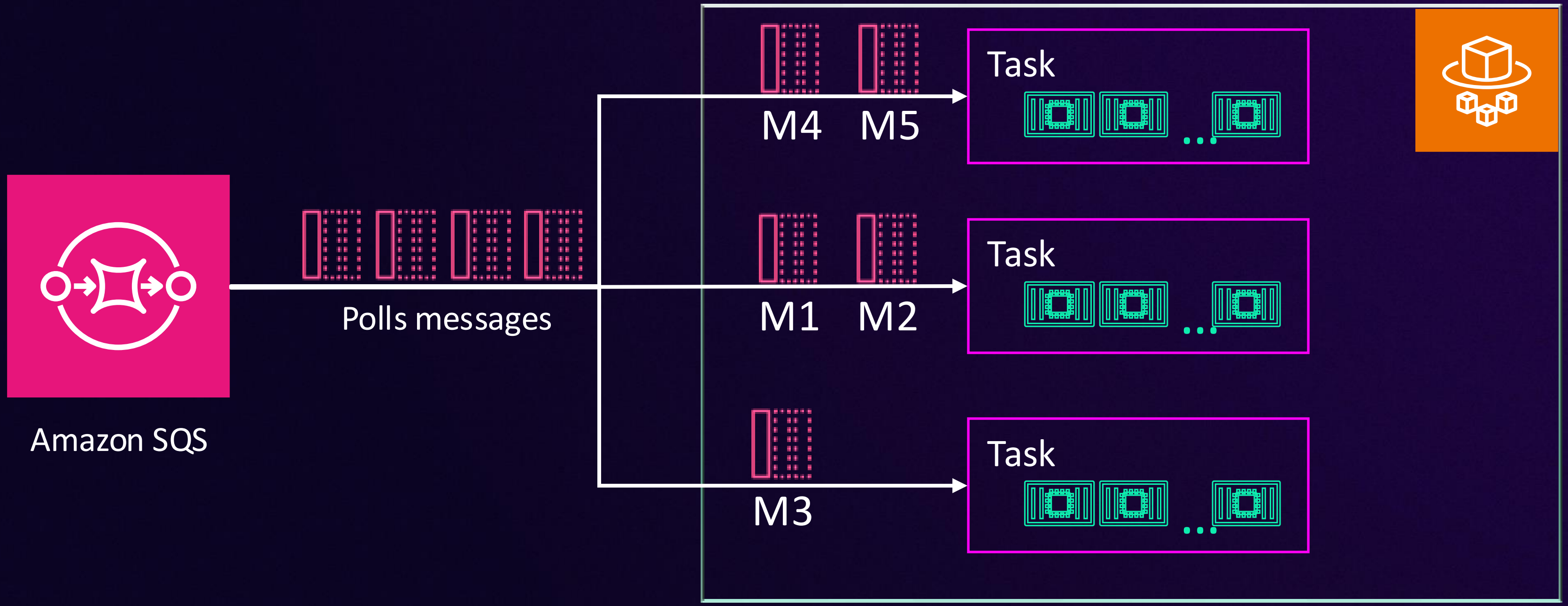


Point-to-point integration with Amazon SQS



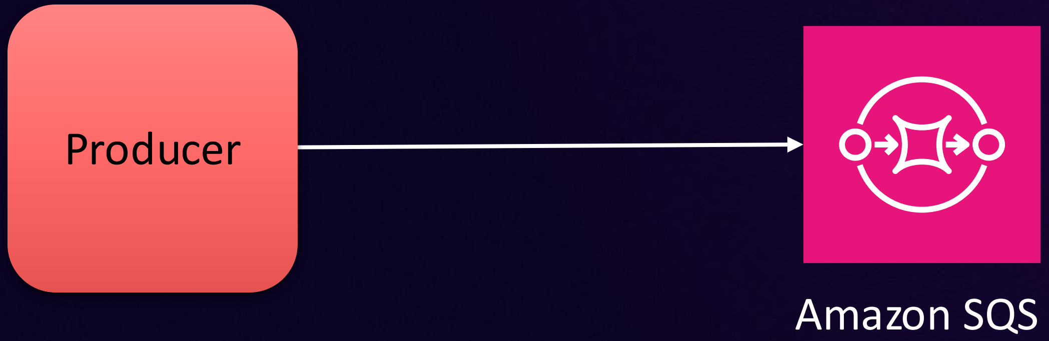
Consuming messages from Amazon SQS

Animation



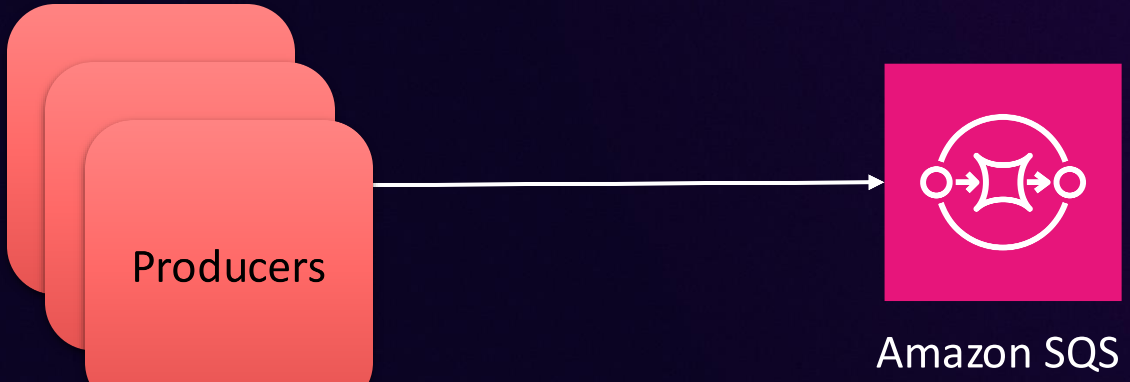
```
List<Message> messages =
sqs.receiveMessage(queueUrl).getMessages();
//delete
for (Message m : messages) {
     sqs.deleteMessage(queueUrl, m.getReceiptHandle());
}
```

Traffic pattern – Steady



Traffic pattern – Unpredictable

Changed from bursty to unpredictable



Potential issues without auto scaling

1. Underutilized resources
2. Message loss
3. Increased latency

Auto scaling by metrics

Predefined metrics
e.g., queue depth

Custom metrics
e.g., Amazon SQS/
task backlog

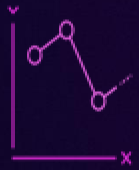
Auto scaling by predefined metrics

ApproximateNumberOfMessagesVisible – the number of messages to be processed

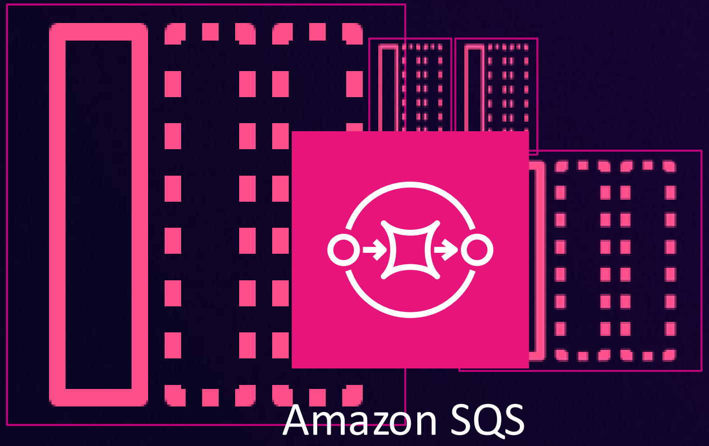
```
auto_scale_task_.scale_on_metric(
        "SqsFargateCdkPythonScaleOnMetric",
        metric=queue.metric_approximate_number_of_messages_visible(),
        adjustment_type=autoscaling.AdjustmentType.CHANGE_IN_CAPACITY,
        cooldown=Duration.seconds(300),
        scaling_steps=[{"upper": 0, "change": -1},{"lower": 1, "change": +1}]
)
```

Auto scaling by predefined metrics

Animation



ApproximateNumberOfMessagesVisible – the number of messages to be processed

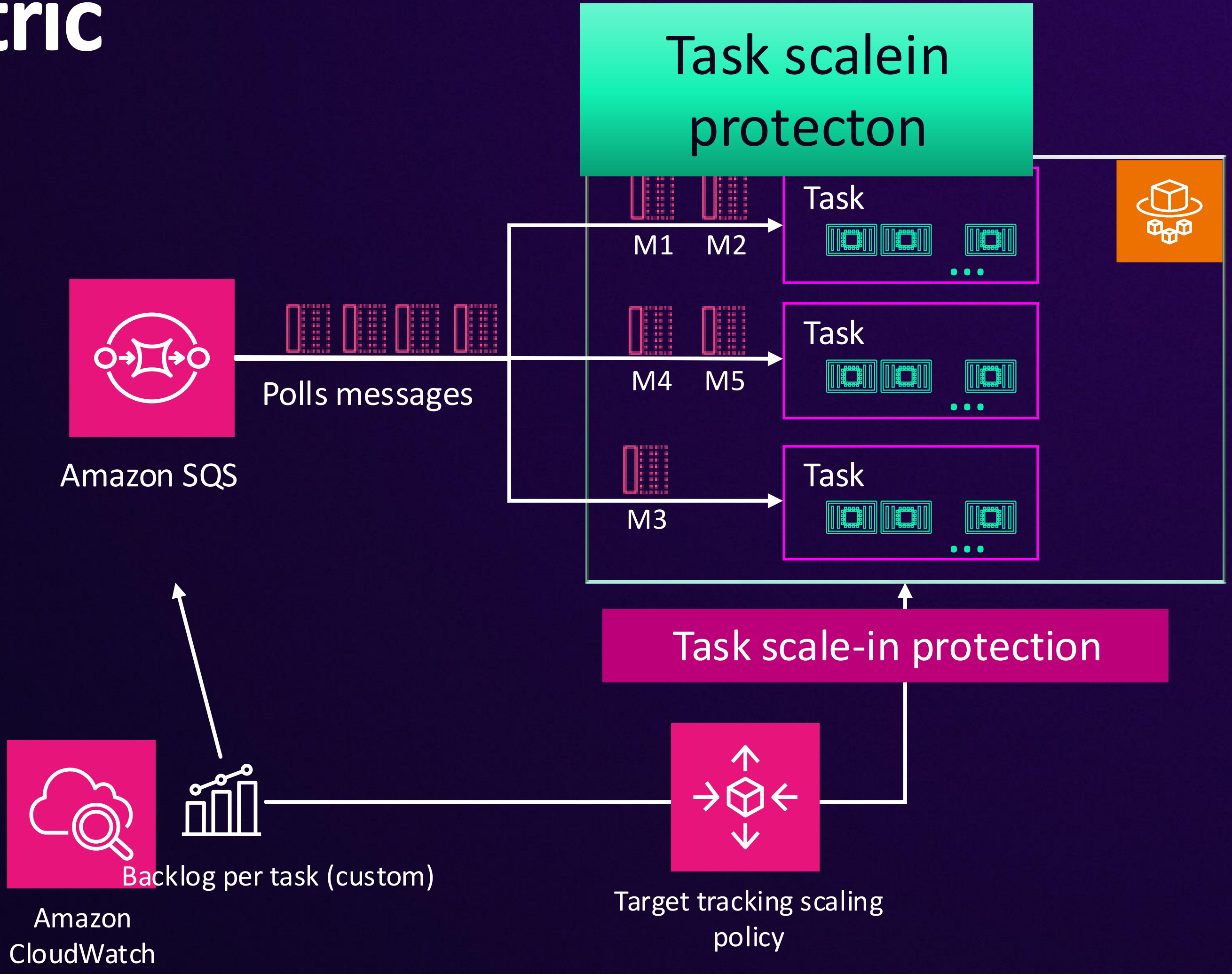


This does not consider how long to process message nor the acceptable latency

Auto scaling by custom metric

Backlog per task – the number of messages to be processed/no of running task

Acceptable backlog per task – acceptable latency/average time to process a message



Orchestration patterns

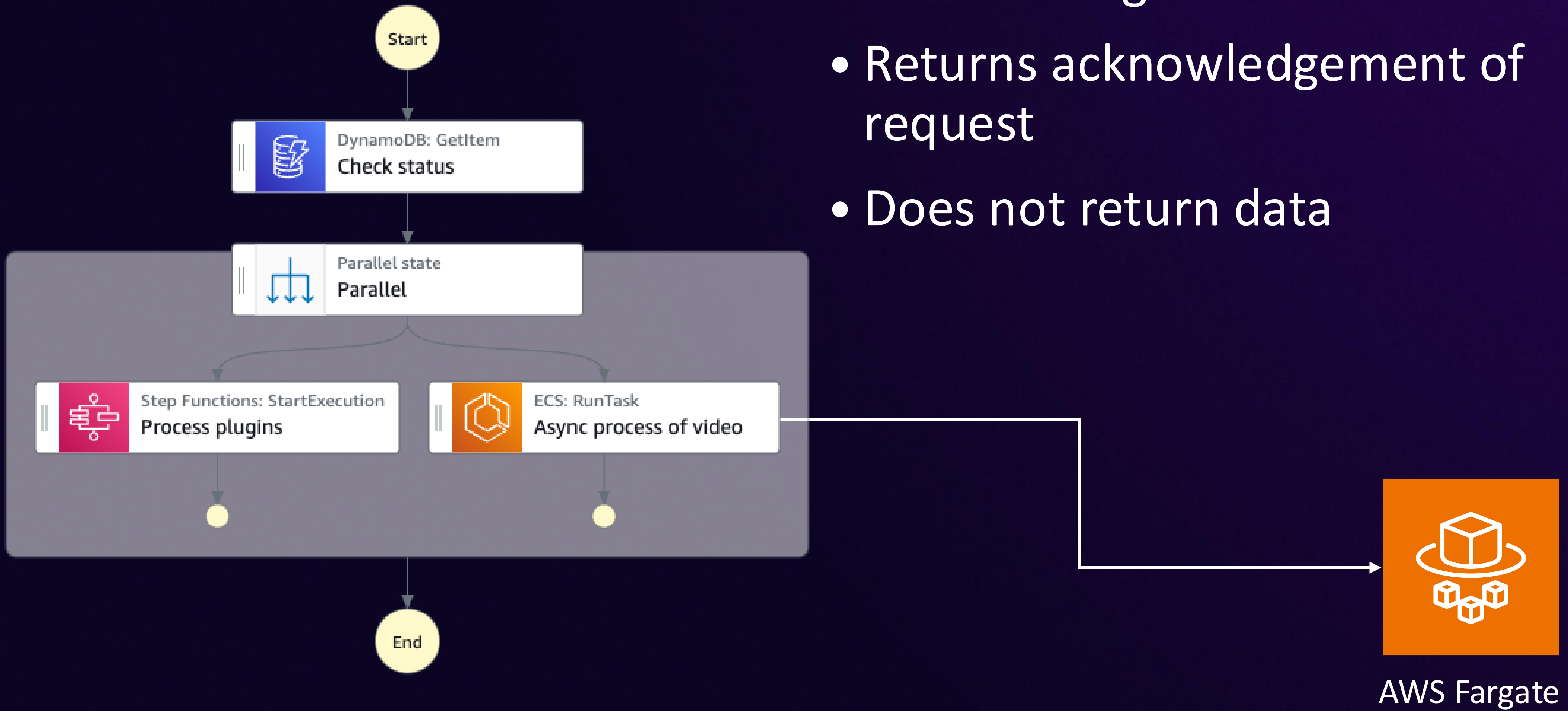
AWS Step Functions

- Pay per use
- Scales automatically
- Fully managed
- Drag and drop or ASL
- Built-in error handling
- Integrates with over 200 AWS services

Amazon Managed Workflows for Apache Airflow

- Easy Apache Airflow deployment
- Automatic scaling
- Built-in security
- Plug-in integration
- Great for data orchestration

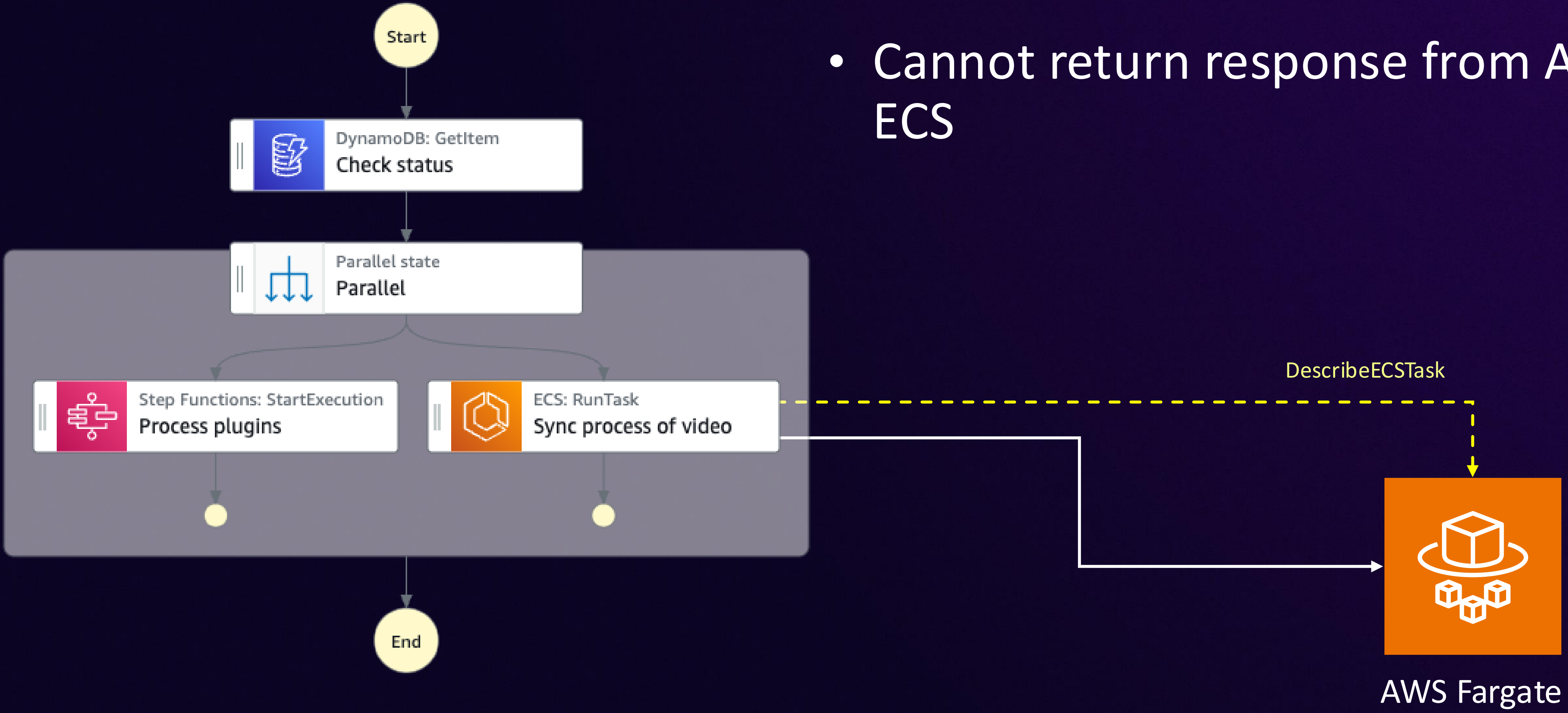
The async pattern



- Async invocation
- Fire and forget
- Returns acknowledgement of request
- Does not return data

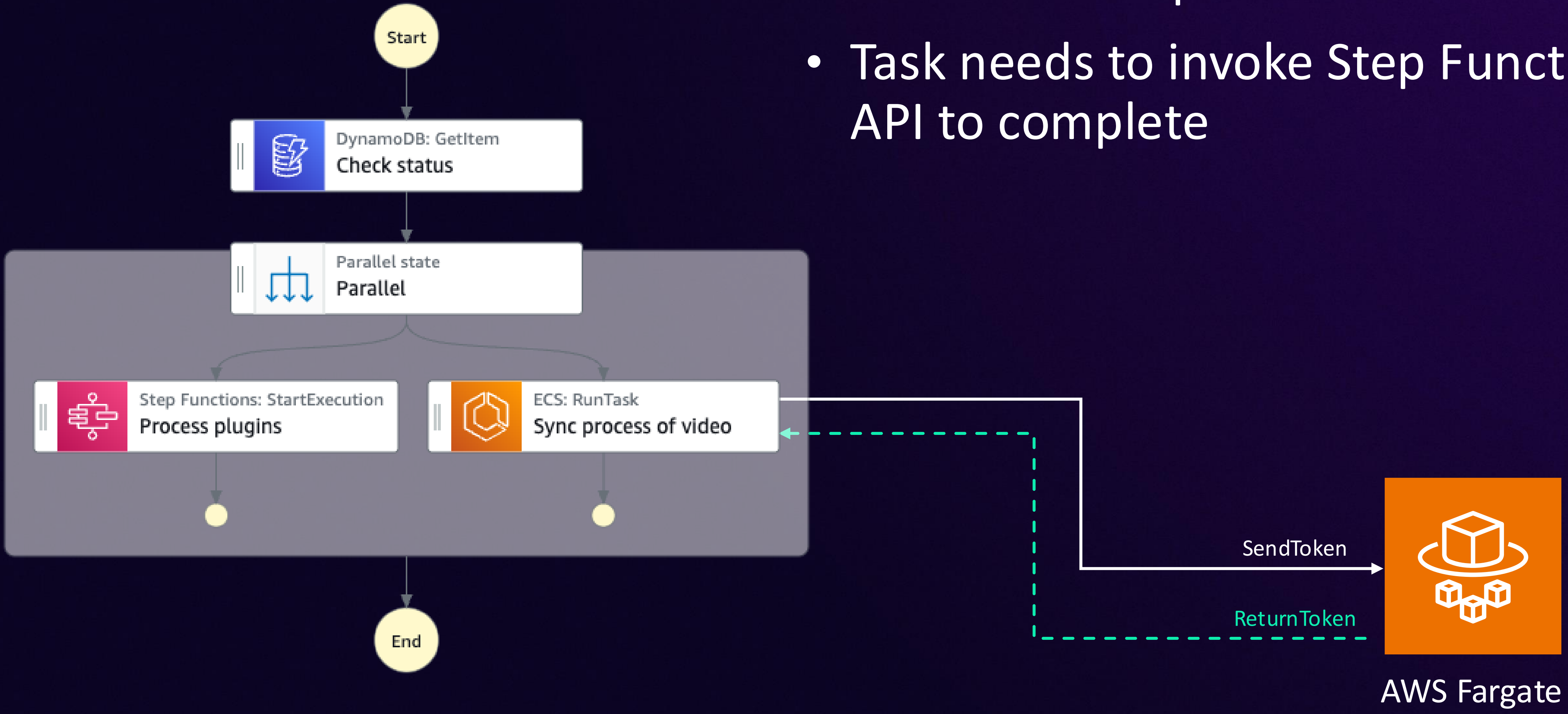
The .sync pattern

- Waits to complete tasks
- Polls DescribeECSTask for status
- Cannot return response from Amazon ECS

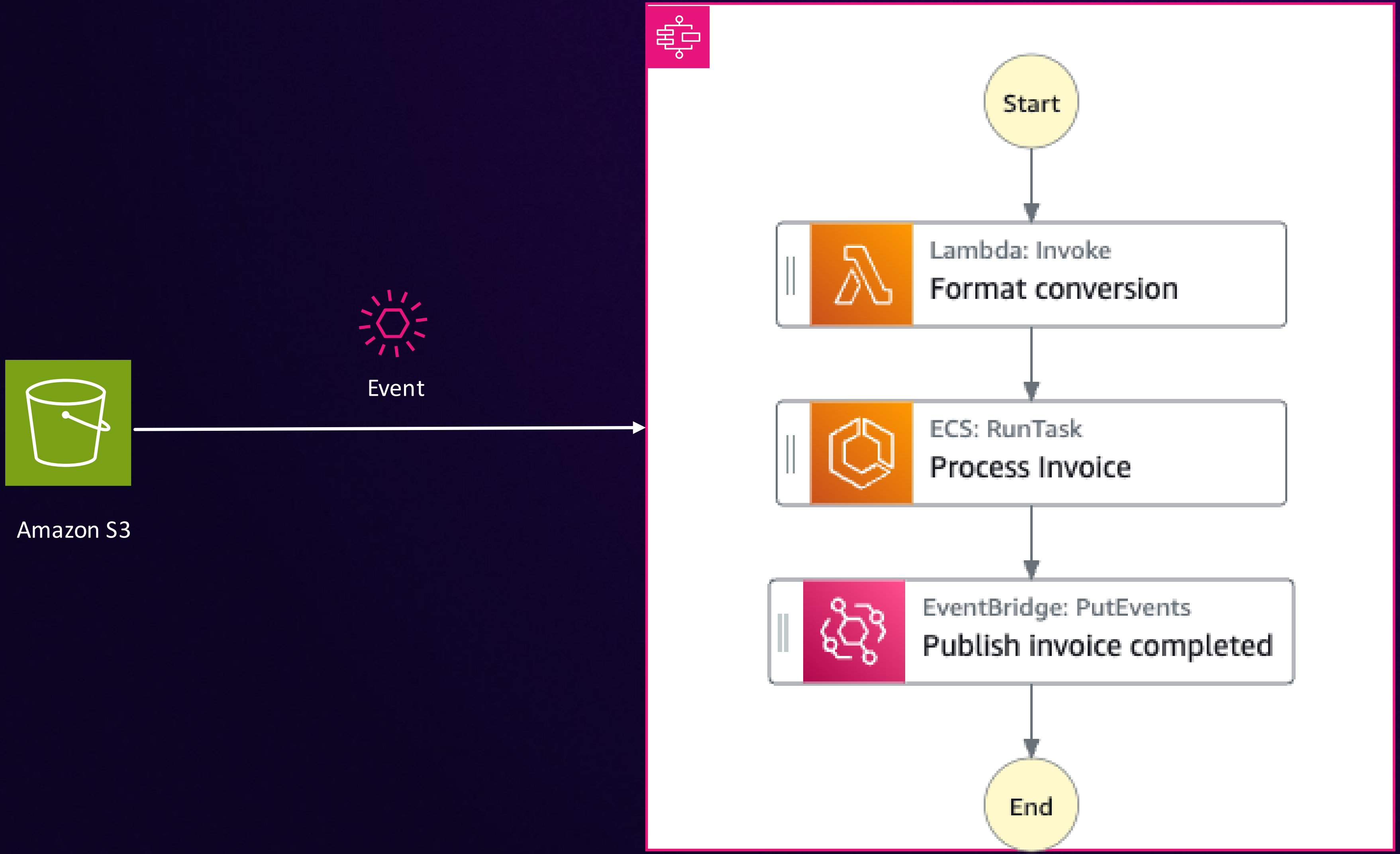


The callback pattern

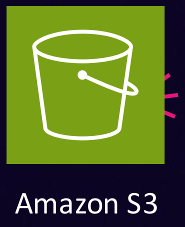
- Uses token to send task completion
- Can return response from Amazon ECS
- Task needs to invoke Step Functions API to complete

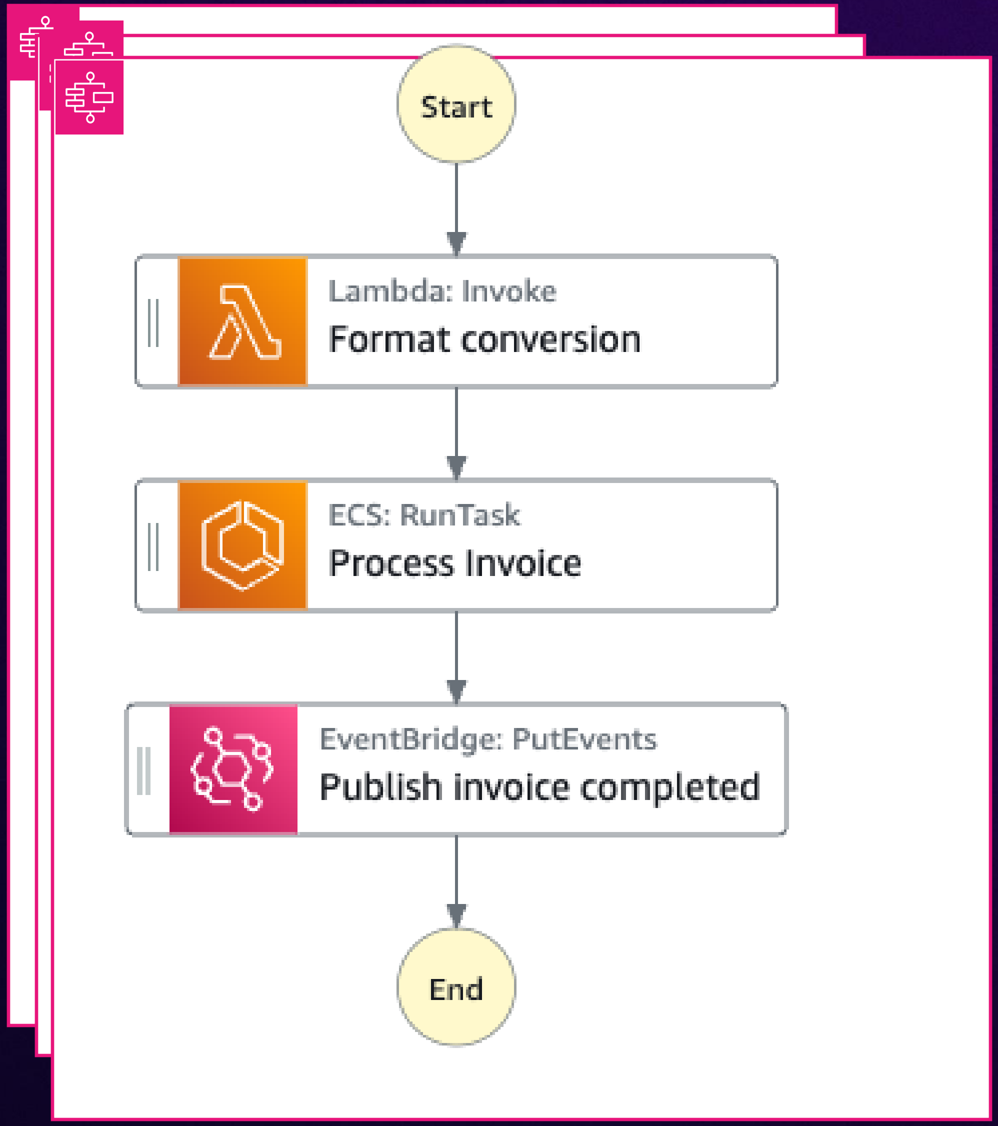


Large-scale processing with RunTask

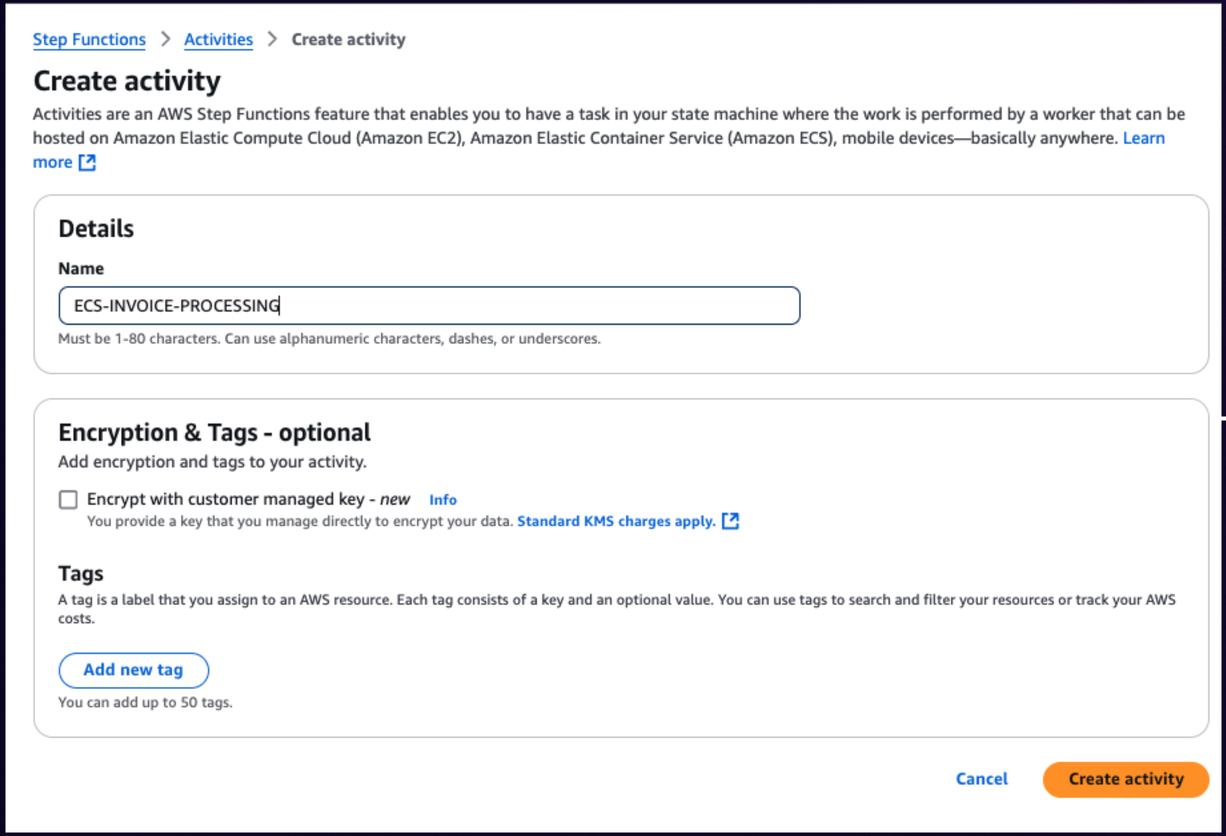


Large-scale processing with RunTask – Challenge





Activity



Step Functions > Activities > Create activity

Create activity

Activities are an AWS Step Functions feature that enables you to have a task in your state machine where the work is performed by a worker that can be hosted on Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), mobile devices—basically anywhere. Learn more

Details

Name

ECS-INVOICE-PROCESSING

Must be 1-80 characters. Can use alphanumeric characters, dashes, or underscores.

Encryption & Tags - optional

Add encryption and tags to your activity.

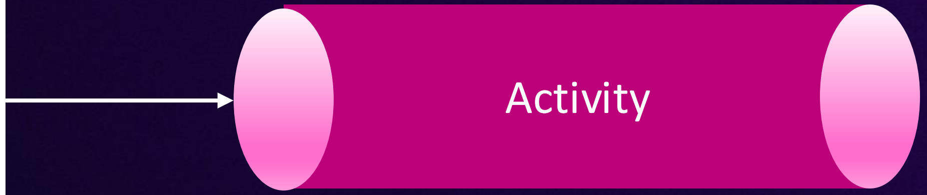
☐ Encrypt with customer managed key - *new* Info

You provide a key that you manage directly to encrypt your data. Standard KMS charges apply.

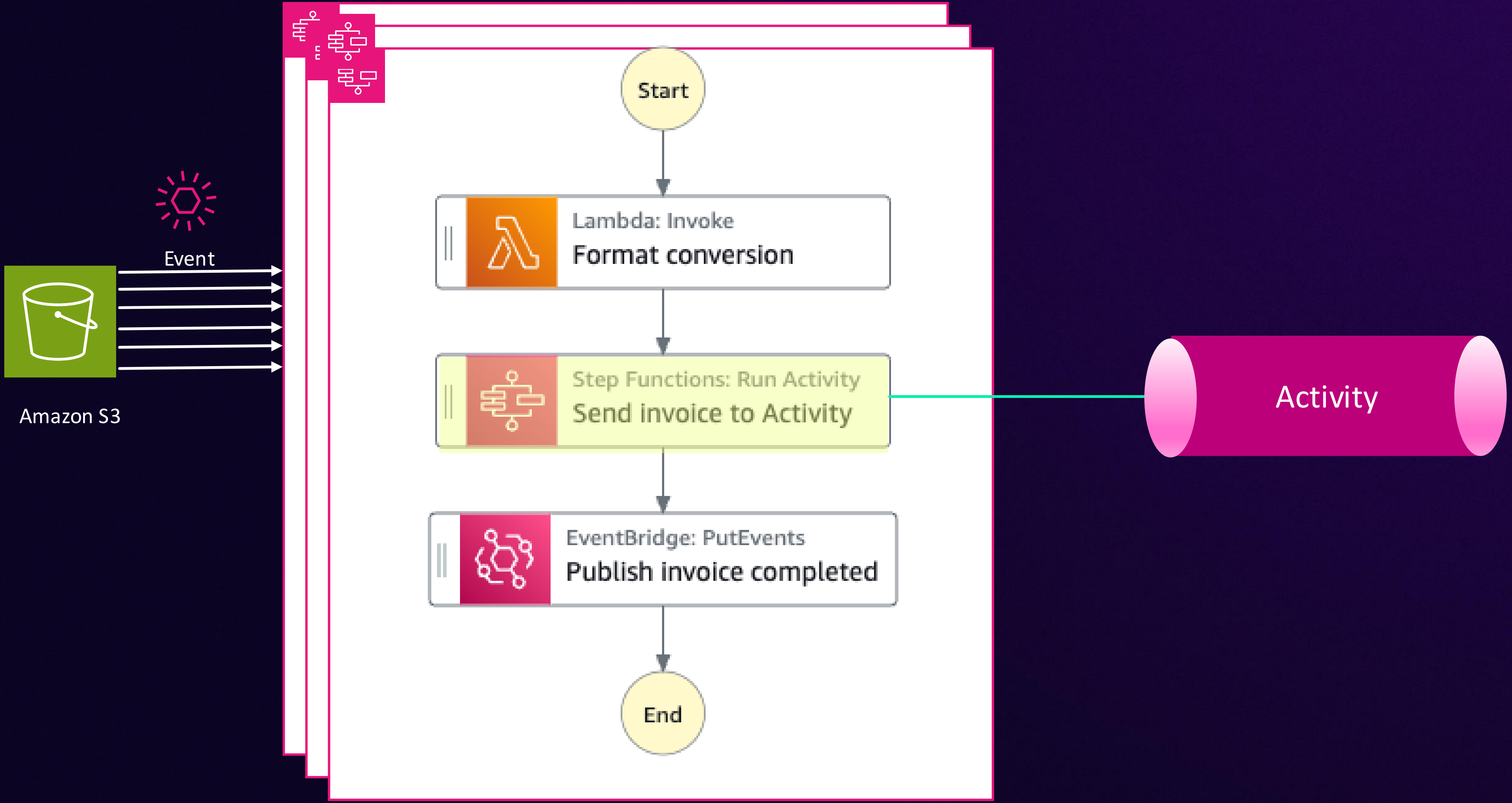
Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

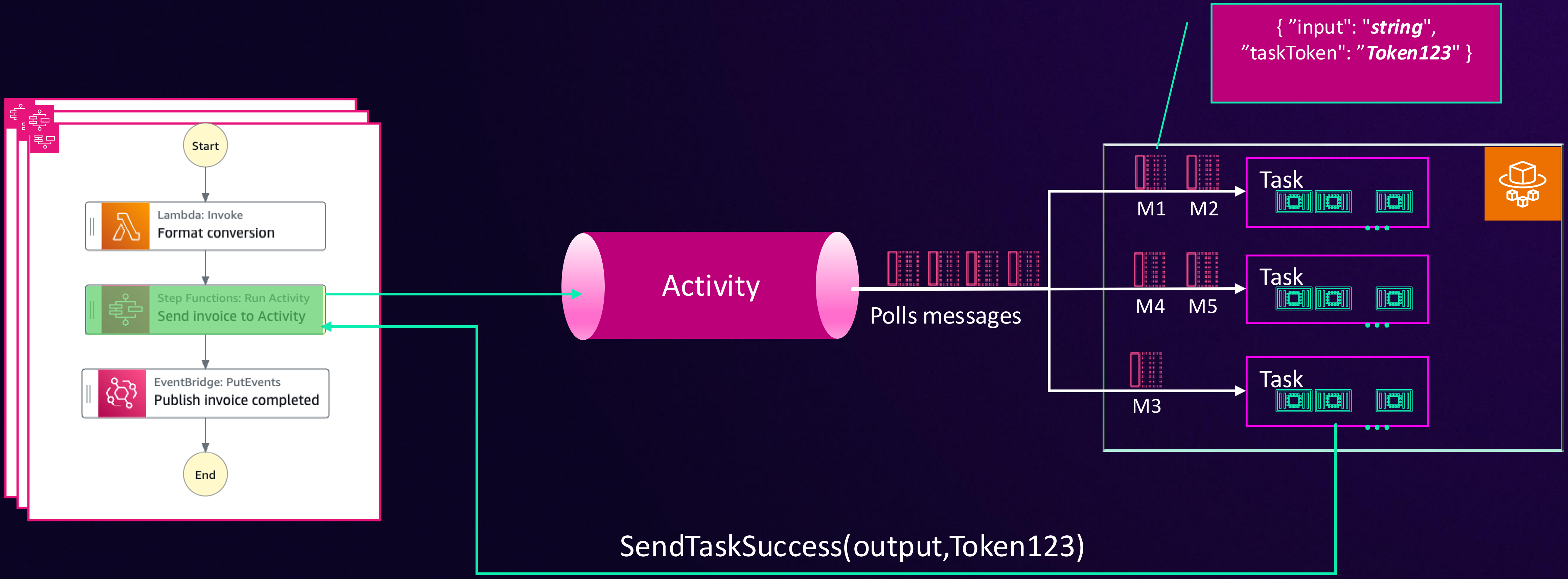
You can add up to 50 tags.



Sending messages to activities



Processing messages from activity


Start
Lambda: Invoke
Format conversion
Step Functions: Run Activity
Send invoice to Activity
EventBridge: PutEvents
Publish invoice completed
End
Activity
Polls messages
{ "input": "*string*", "taskToken": "*Token123*" }
M1
M2
M4
M5
M3
Task
Task
Task
SendTaskSuccess(output,Token123)

Benefits of activities

Managed queue with 1-year retention

Connects on-premises/ containerized services

Loosely coupled architecture

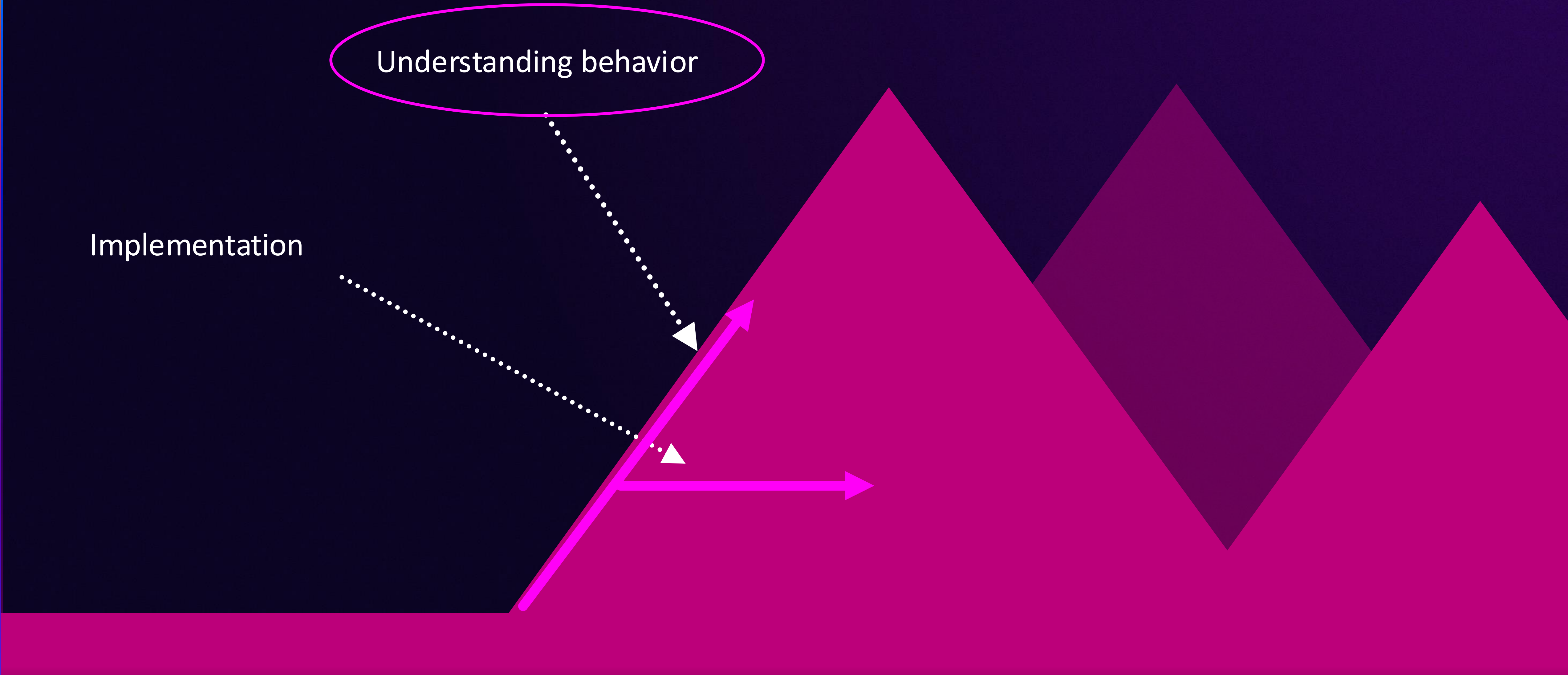
Flexibility to use Spot for cost savings

Wrap-up

“Systems that don’t evolve will die.”

Dr. Werner Vogels
VP and CTO at Amazon.com

Where to start your EDA journey?



Resources

https://s12d.com/svs339-24

Check out these other sessions

API301-R1: Accelerating event-driven architecture with event storming
Wednesday, December 4, 1:00 PM, Caesars Forum, Level 1, Academy 416

SVS332-R1: Build secure & performant apps easily with Amazon ECS & AWS Fargate
Wednesday, December 4, 9:00 AM, MGM Grand, Level 3, Premier 320

SVS336: Serverless data ingestion and processing using containers on Amazon ECS
Tuesday, December 3, 5:30 PM, Mandalay Bay, Level 2 South, Reef C

Continue your AWS serverless learning

Learn at your own pace

Expand your serverless skills with our learning plans on **AWS Skill Builder**

Increase your knowledge

Use our **Ramp-Up Guides** to build your serverless knowledge

Earn the AWS Serverless badge

Demonstrate your knowledge by achieving **digital badges**

https://s12d.com/serverless-learning

Thank you!

Please complete the session survey in the mobile app

Uma Ramadoss

@uma_ramadoss

Eric Johnson

@edjgeek