



EDA DAY LONDON 2024

Serverless compute at the heart of your EDA

Julian Wood

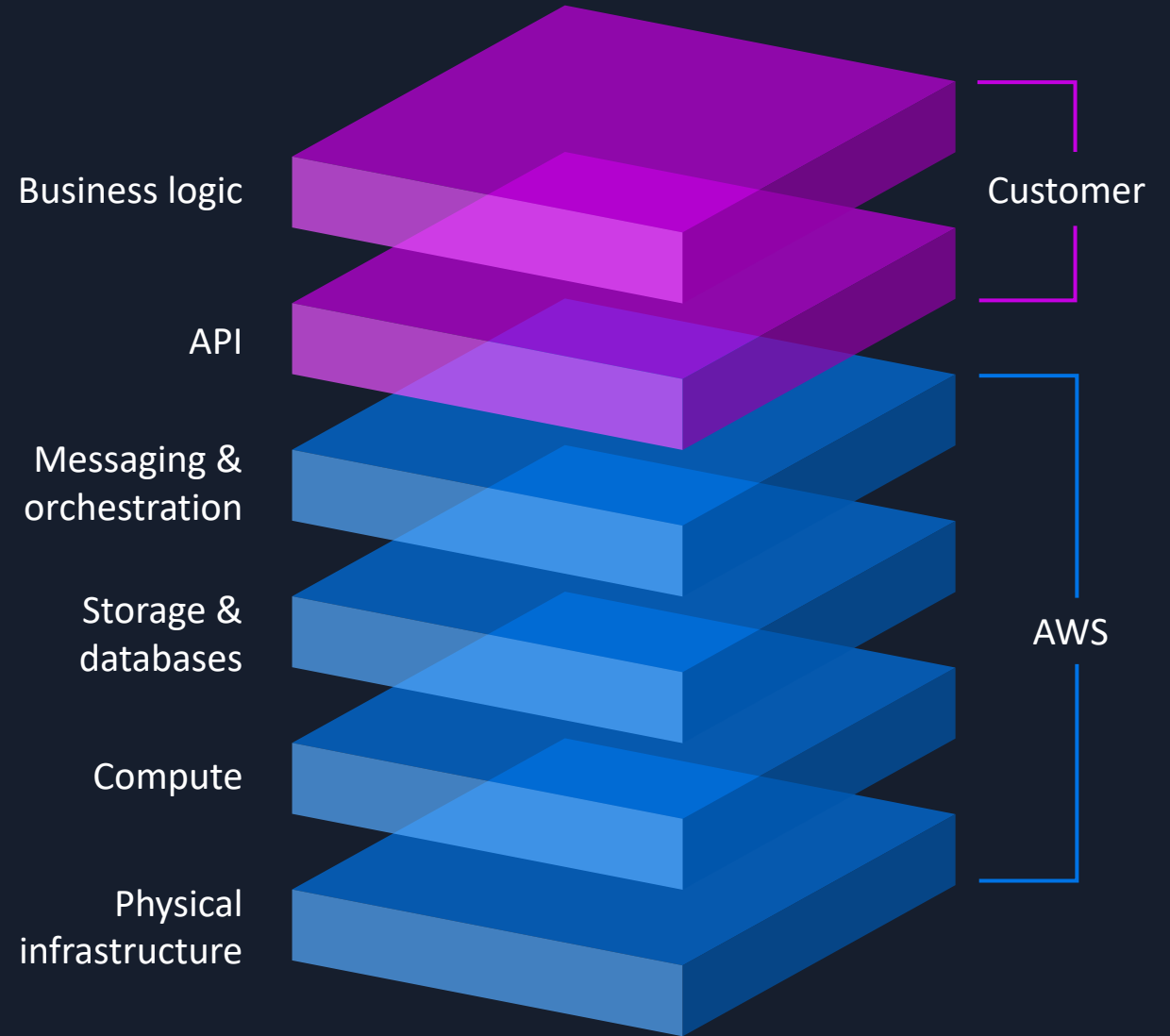
Principal Developer Advocate
AWS Serverless

Serverless EDA



What is Serverless?

Allow development teams
to focus on writing code
that solve
business problems



AWS Serverless spectrum

INCLUDES

AWS OFFERS A WIDE PORTFOLIO OF SERVERLESS SERVICES

Compute



AWS
Lambda



AWS
Fargate



Amazon
ECS



AWS App
Runner

Storage



Amazon
S3



Amazon
EFS

Workflows and Integrations



Amazon
EventBridge



AWS Step
Functions



Amazon
API Gateway



AWS
AppSync



Amazon
SQS



Amazon
SNS

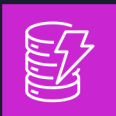


Amazon
Kinesis

Databases and Analytics



Amazon Aurora
Serverless



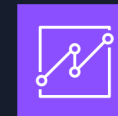
Amazon
DynamoDB



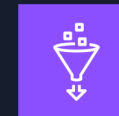
Amazon
OpenSearch Service



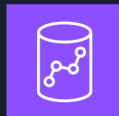
Amazon
Bedrock



Amazon
QuickSight



AWS
Glue



Amazon
Redshift

Serverless at scale



AWS Lambda processes **tens of trillions of** requests every month



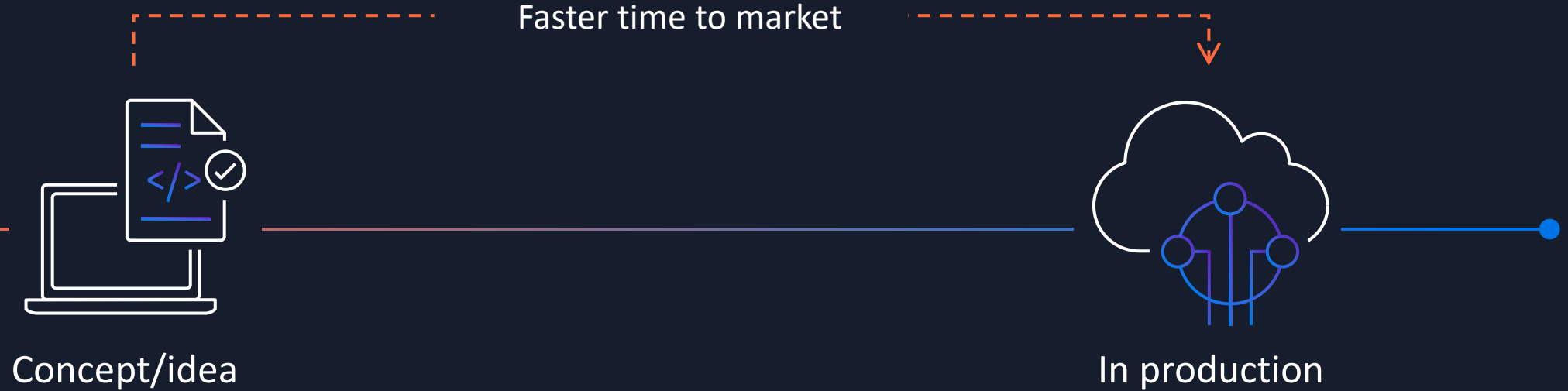
Average number of invokes on Amazon EventBridge is over **2 trillion per month**



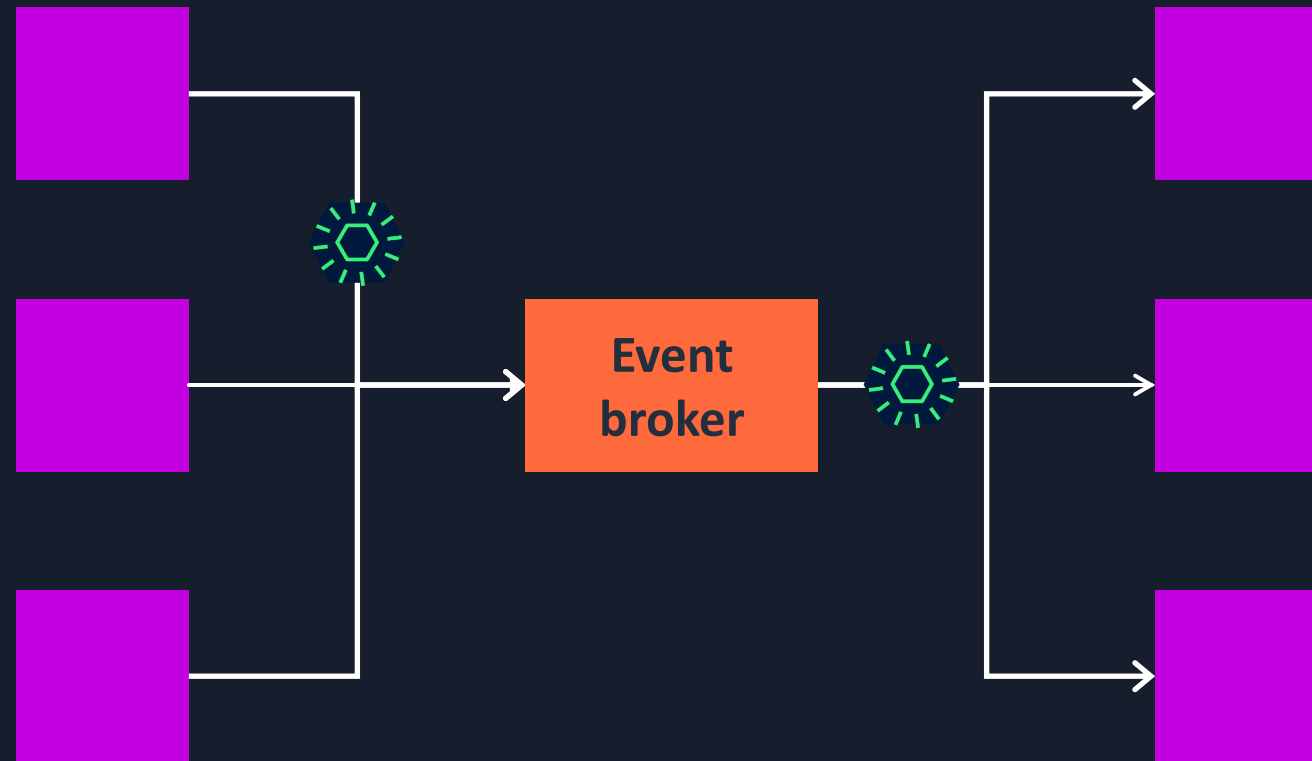
Over 2.25 billion Amazon ECS tasks are launched each week

Why go serverless?

THE PHILOSOPHY BEHIND SERVERLESS AND SERVERLESS CONTAINERS



Communicating via events





Amazon EventBridge

Serverless event bus service:

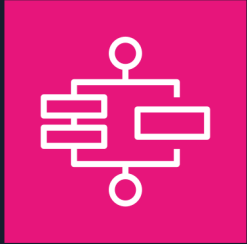
AWS services

Your own applications

SaaS providers

Choreographs events between application domains

- Pay only for the events you process
- Simplified scaling
- No upfront investments, ongoing licensing, or maintenance costs

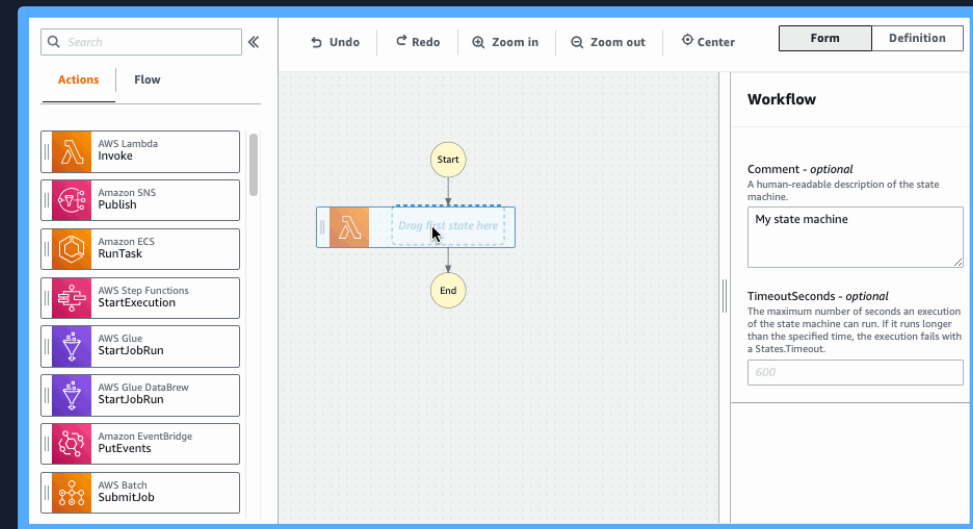


AWS Step Functions

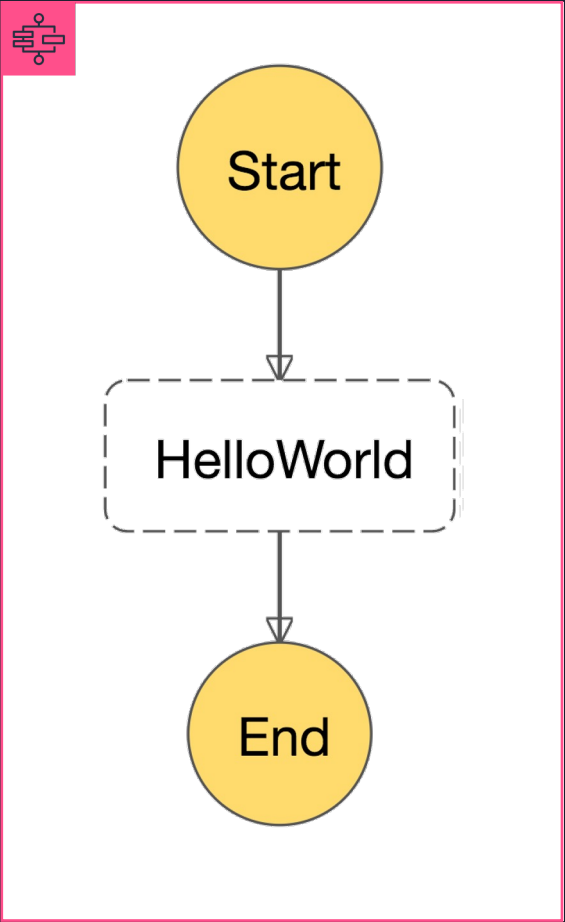
Serverless workflow service.

Orchestrate multiple services

- State Machines
- Composed of steps called states
- Written using Amazon States Language



Service integrations – Request Response



Request



Step Functions
*invokes the task and
waits until the task
responds*



AWS Lambda



Amazon SQS



AWS Step Functions



Amazon Athena



Amazon ECS



Amazon SNS



Amazon SageMaker



Amazon API Gateway



AWS Fargate



Amazon DynamoDB



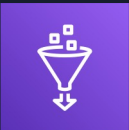
Amazon EMR/EKS



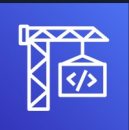
AWS Glue DataBrew



AWS Batch



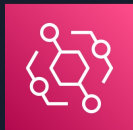
AWS Glue



AWS CodeBuild

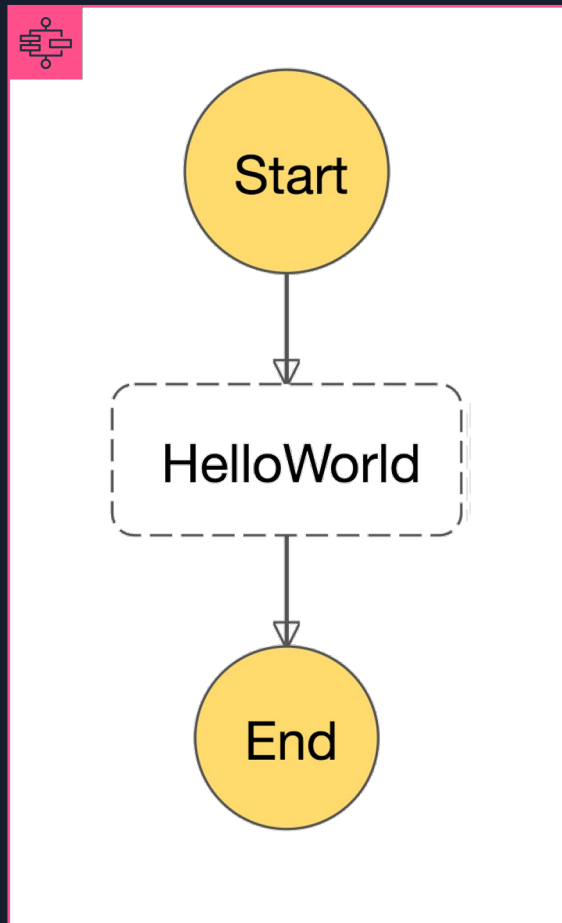


Amazon EKS



EventBridge

Service integrations – Run a job (Synchronous)



Sync



Step Functions
invokes the task and
pauses until the task
completes



Amazon EKS



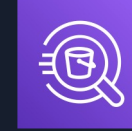
Amazon ECS



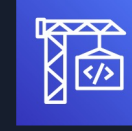
AWS Fargate



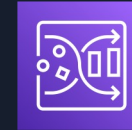
AWS Batch



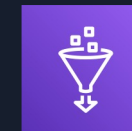
Amazon Athena



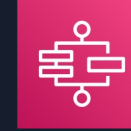
AWS CodeBuild



AWS Glue DataBrew



AWS Glue



AWS Step Functions

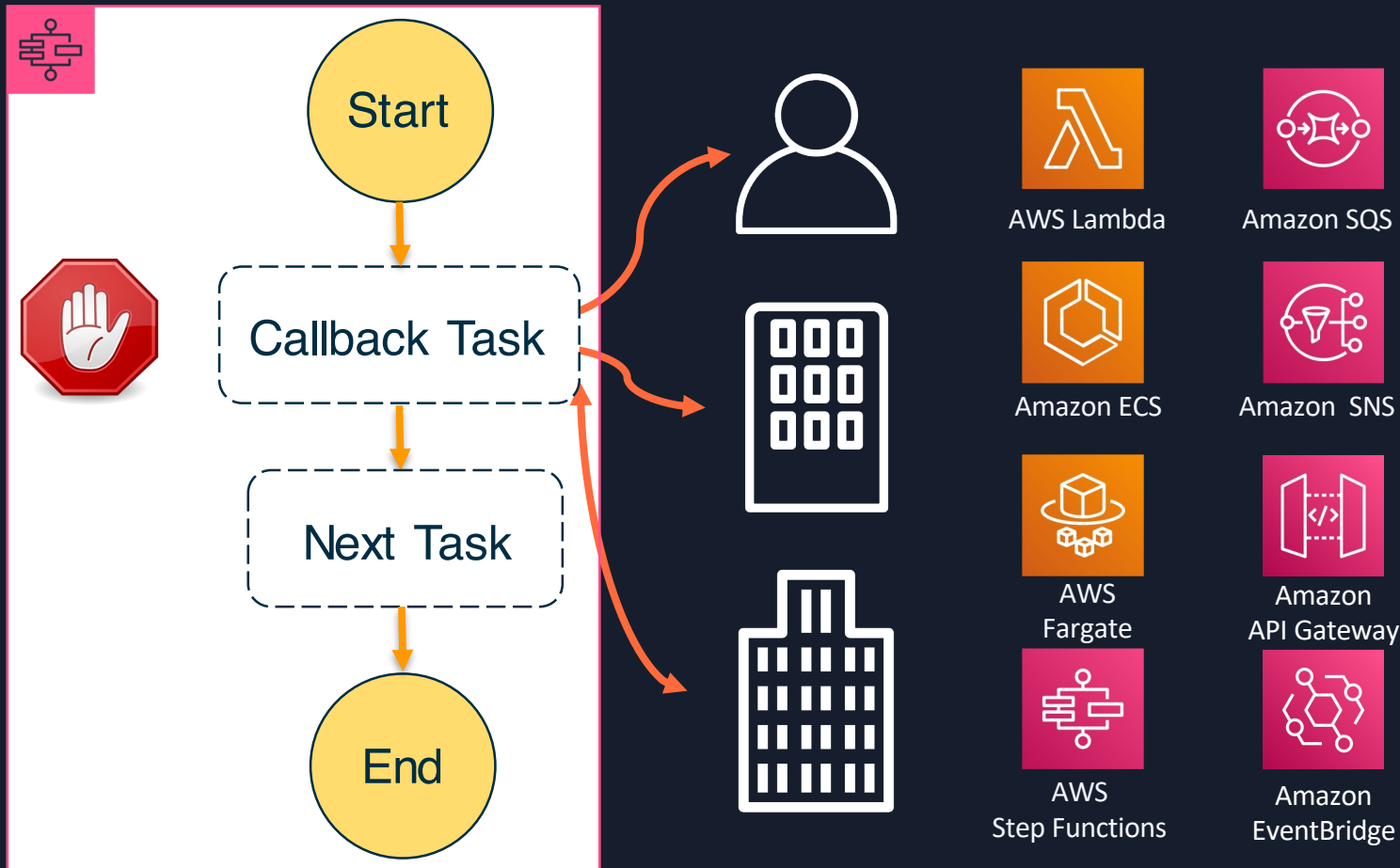


Amazon SageMaker



Amazon EMR/EKS

Service integrations – Wait for Callback



Callback Pattern

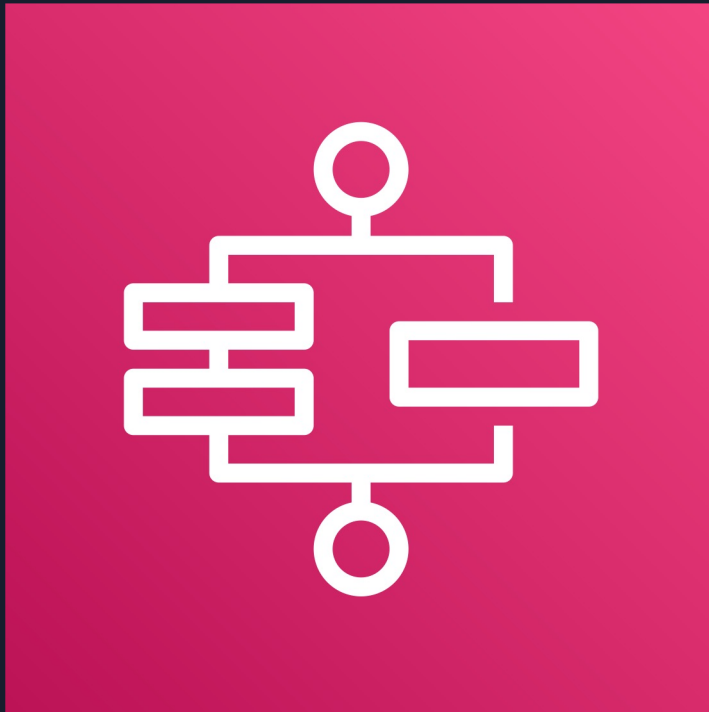
Call an external resource with a token. Pause workflow for a callback event with the token.

Wait for as long as you need – minutes, days, weeks, or months:

- Human activity
- Third party API
- Legacy application

Orchestration/choreography as configuration

Orchestration



AWS Step Functions

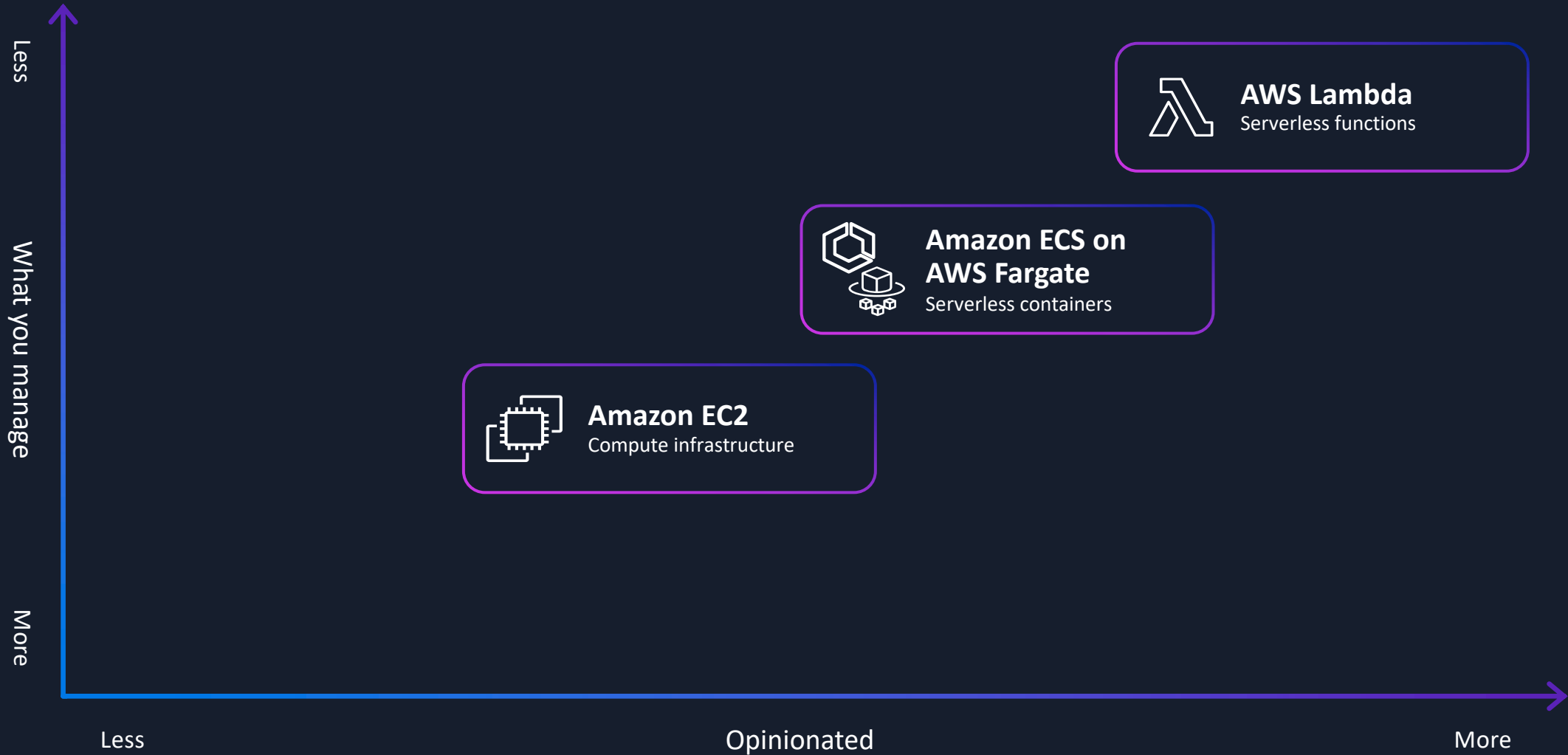
Choreography



Amazon EventBridge

Serverless compute

The AWS serverless landscape





AWS Lambda

Serverless functions



AWS Lambda



Event Source



Changes in
data state



Requests to
endpoints



Changes in
resource state

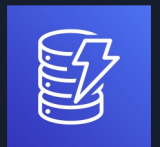
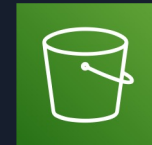


Function



Node.js
Python
Java
C#
Go
Ruby
Bring Your Own

Services / Other



The AWS Lambda invocation model

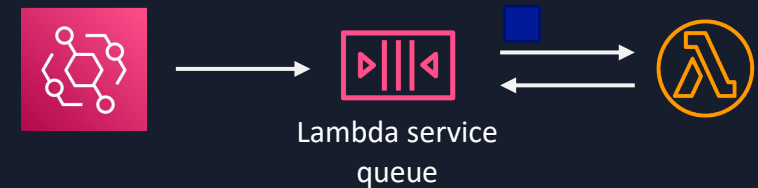
Three types of Lambda integrations:

- **Synchronous**
 - Including request streaming with Node.js
- **Asynchronous**
- **Poll-based**
(event source mapping)

Synchronous



Asynchronous



Poll-based



AWS Lambda function scaling

LAMBDA FUNCTIONS CAN NOW SCALE UP TO 12X FASTER PER MINUTE

Scale up to **1,000 concurrent executions** every **10 seconds** for each function

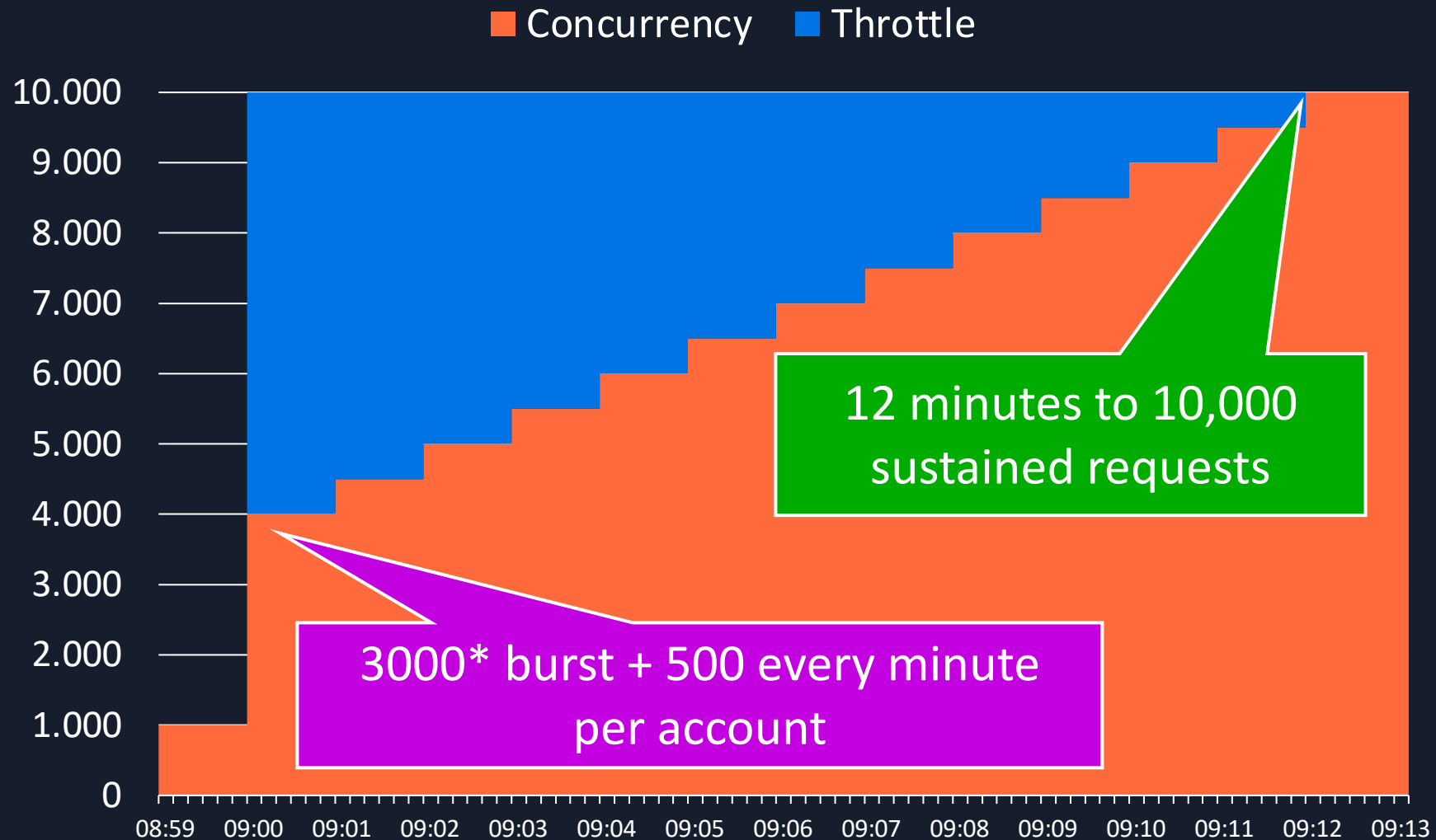
Suited for applications **with varying traffic patterns**

Scale automatically without managing the complexity of infrastructure

Within account concurrency quota

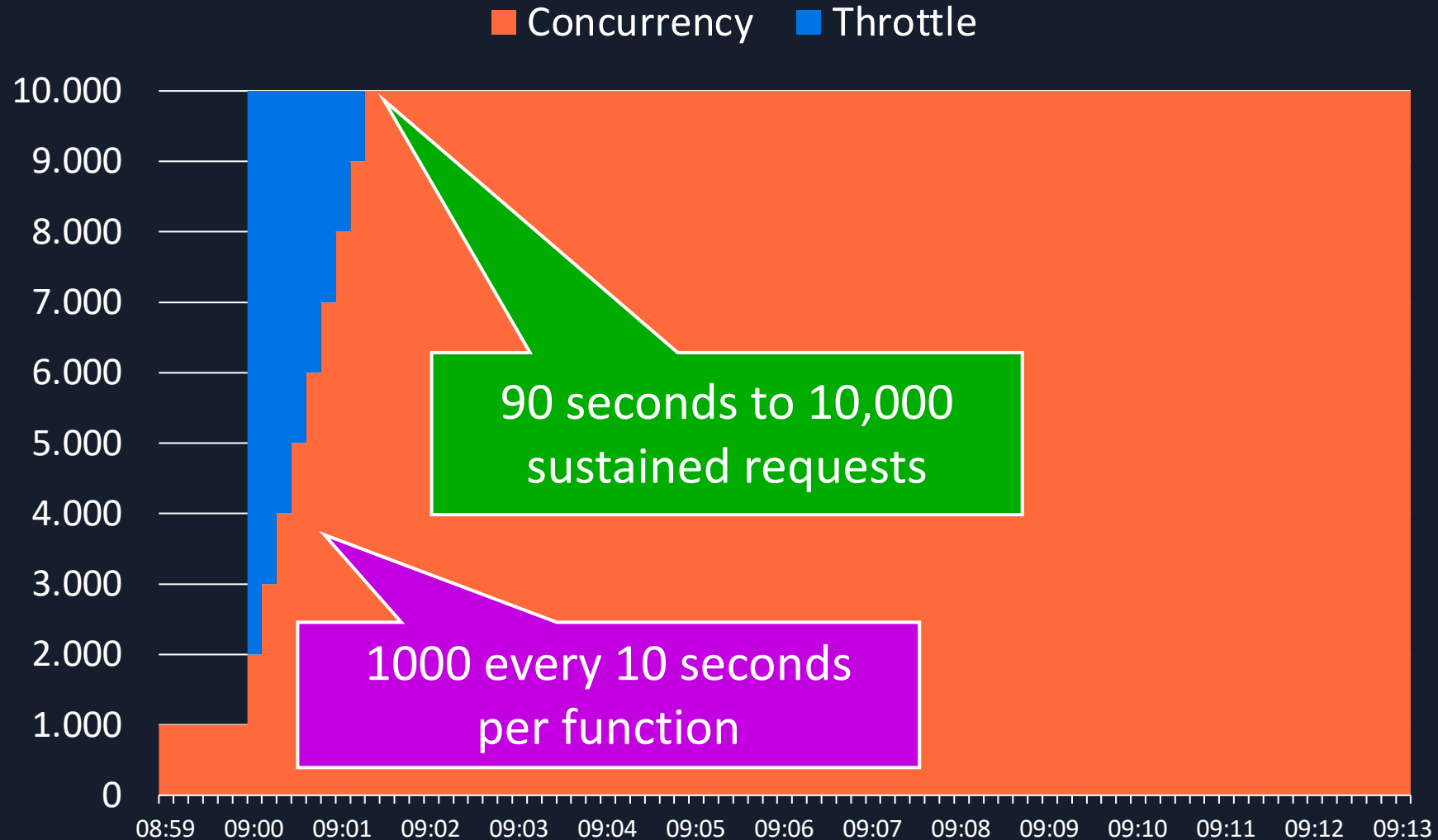


Previous concurrency scaling



New concurrency scaling

NEW



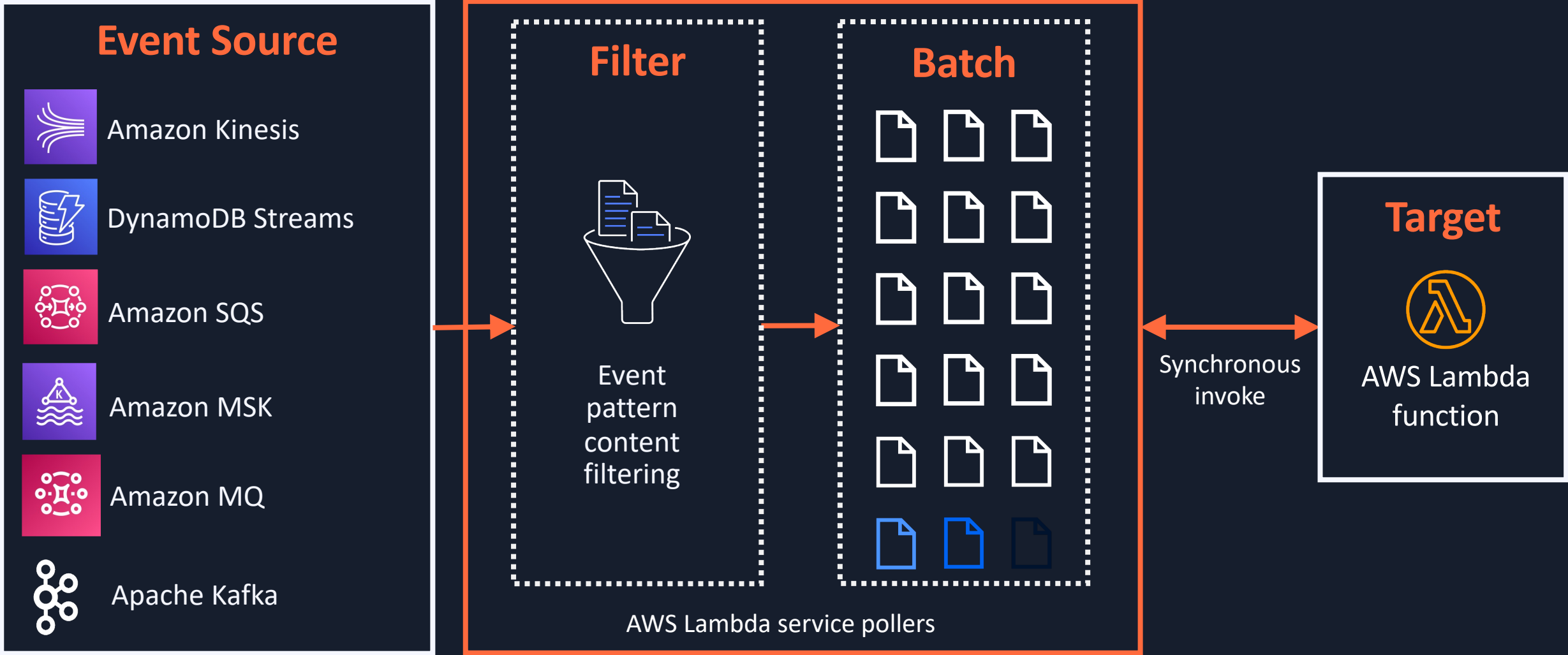
Demo:

Lambda function scaling

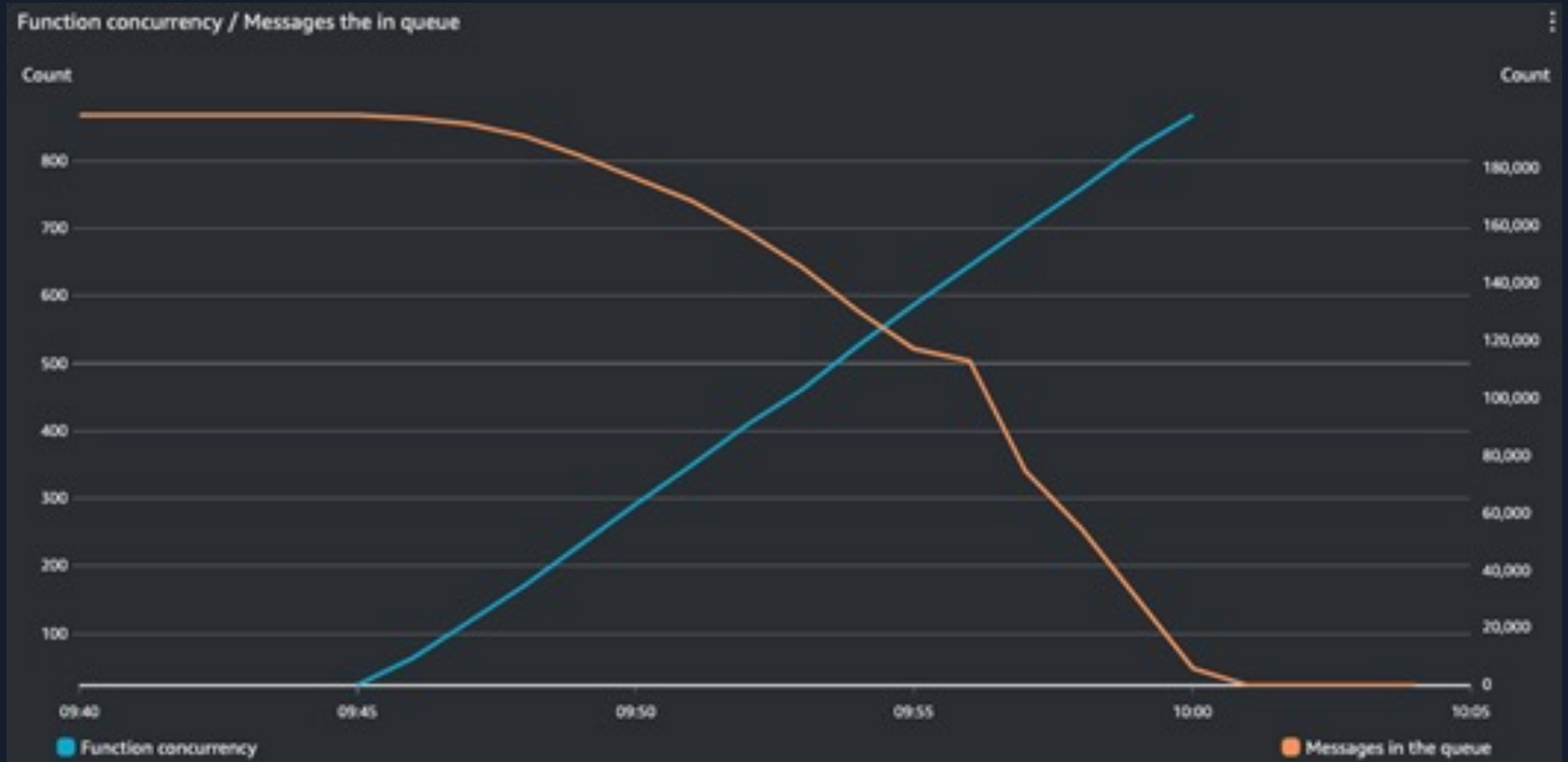


Lambda event source mapping

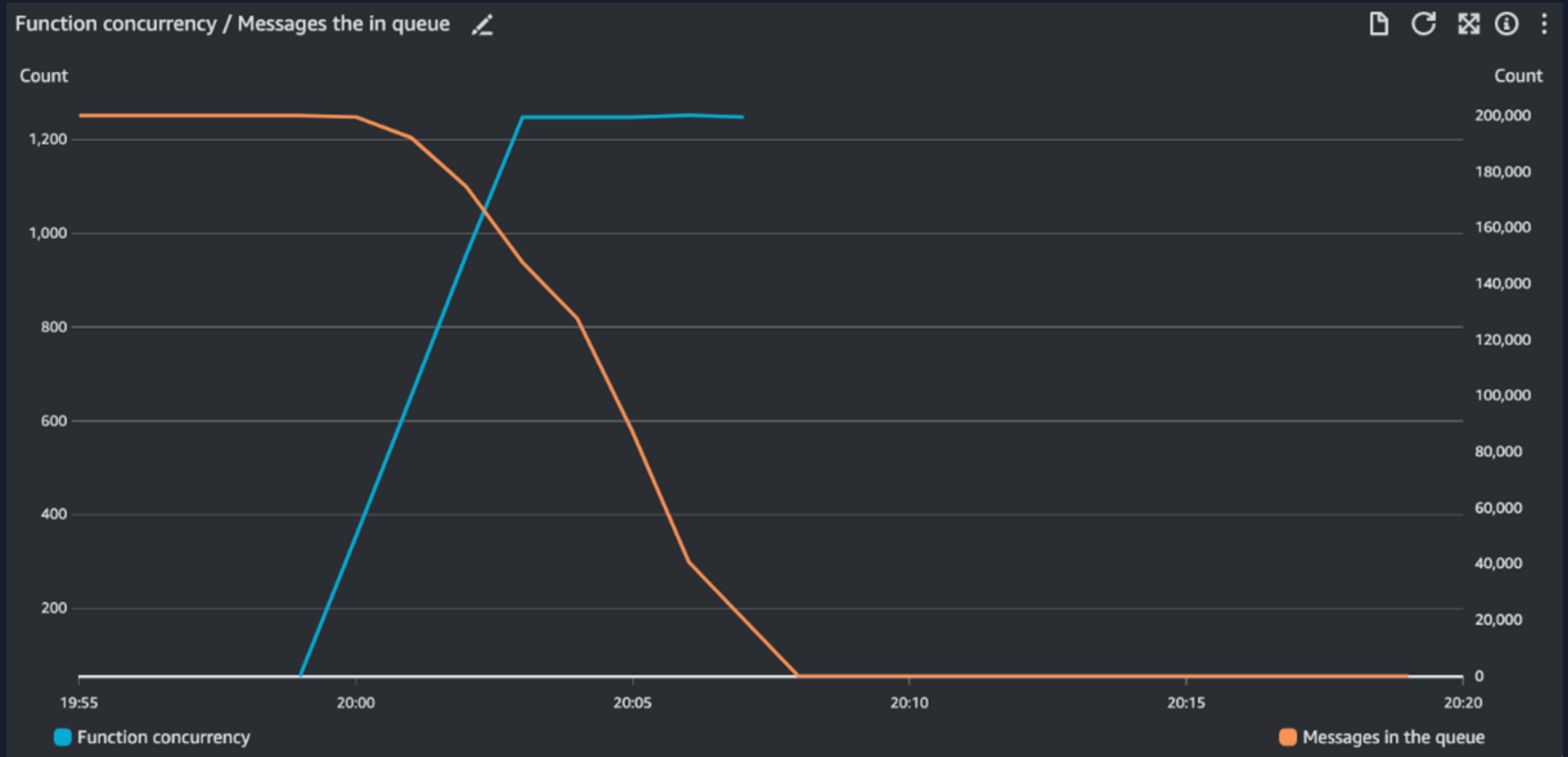
LAMBDA RESOURCE THAT READS FROM AN EVENT SOURCE AND INVOKES A LAMBDA FUNCTION



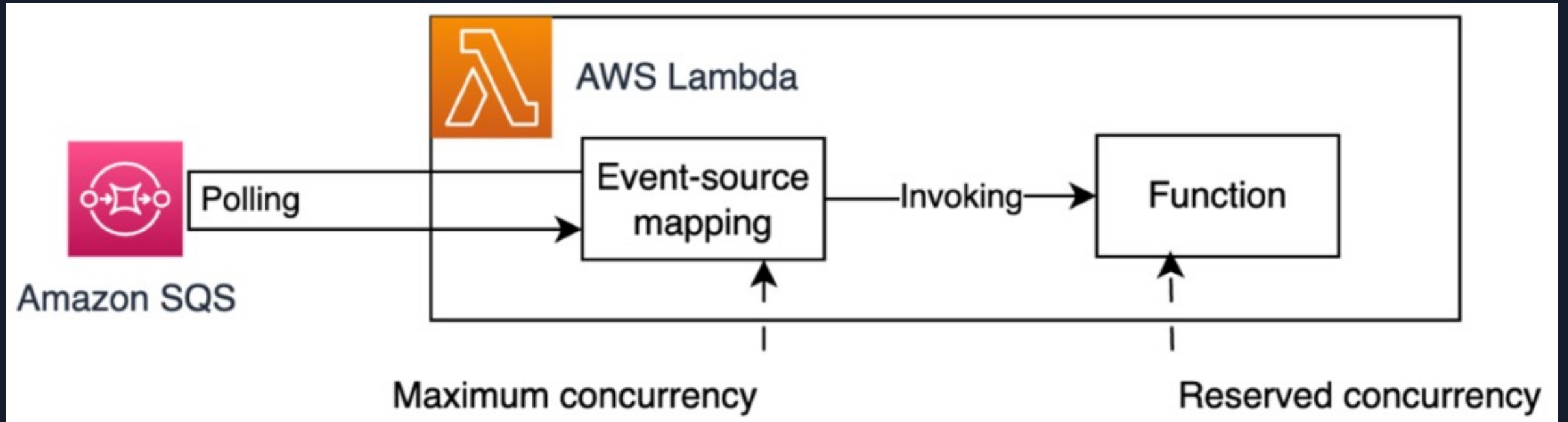
SQS previous scaling



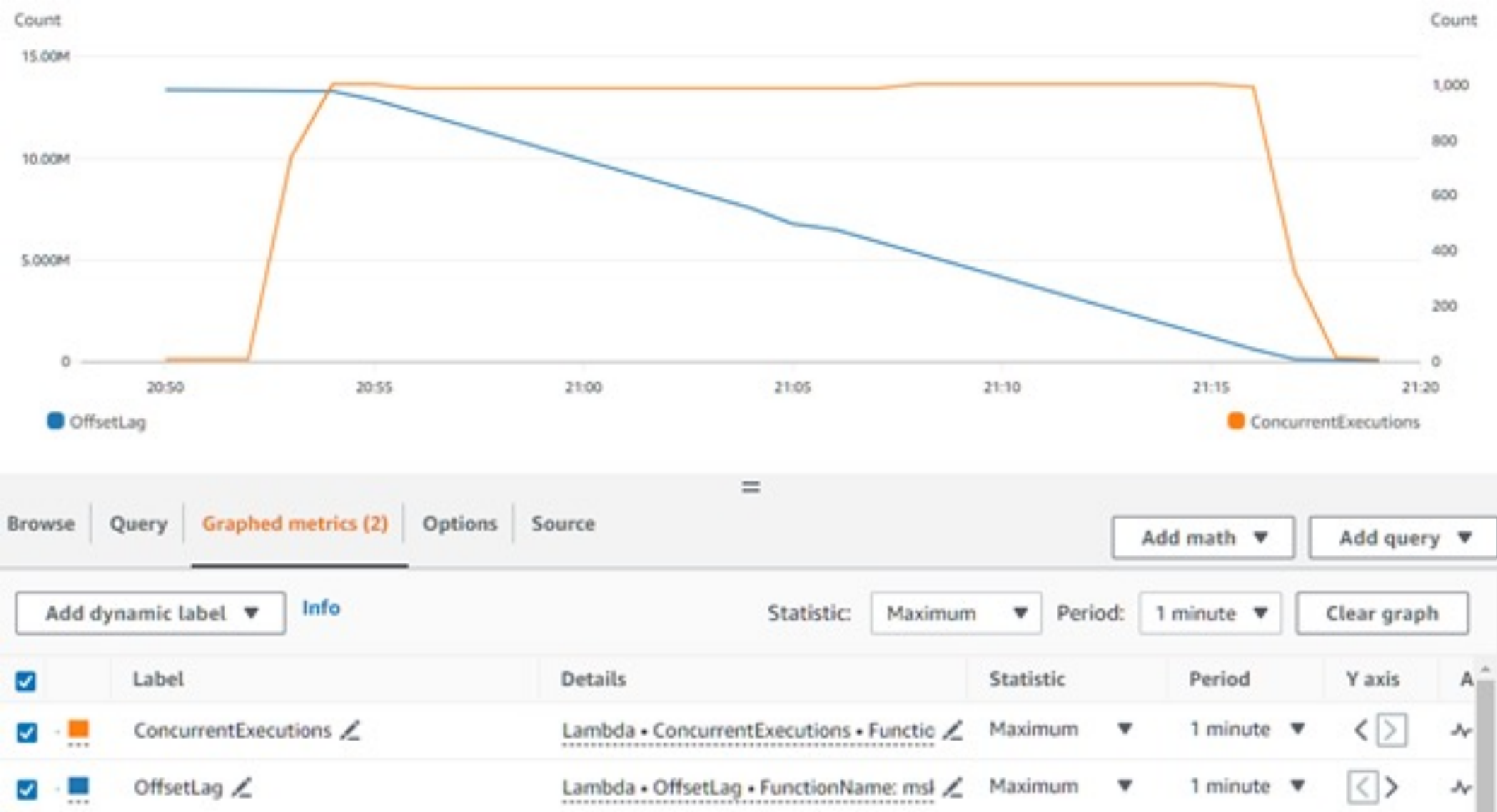
Current SQS scaling



Lambda event source mapping



Kafka scaling



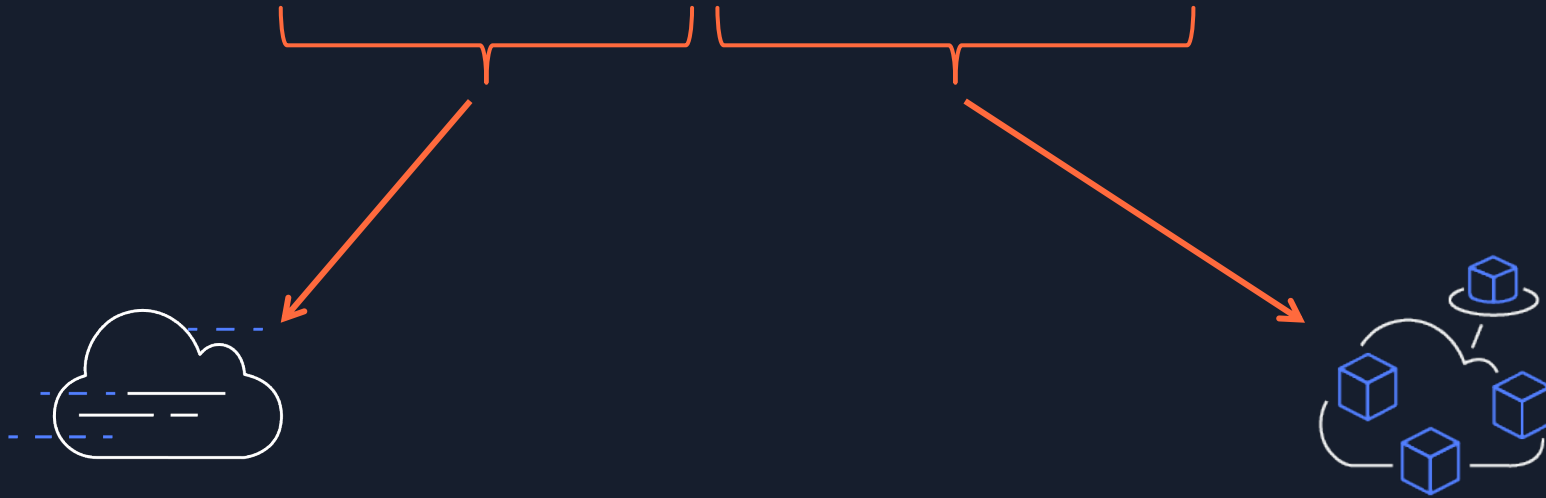


Amazon ECS on AWS Fargate

Serverless containers



Amazon ECS: serverless container orchestrator

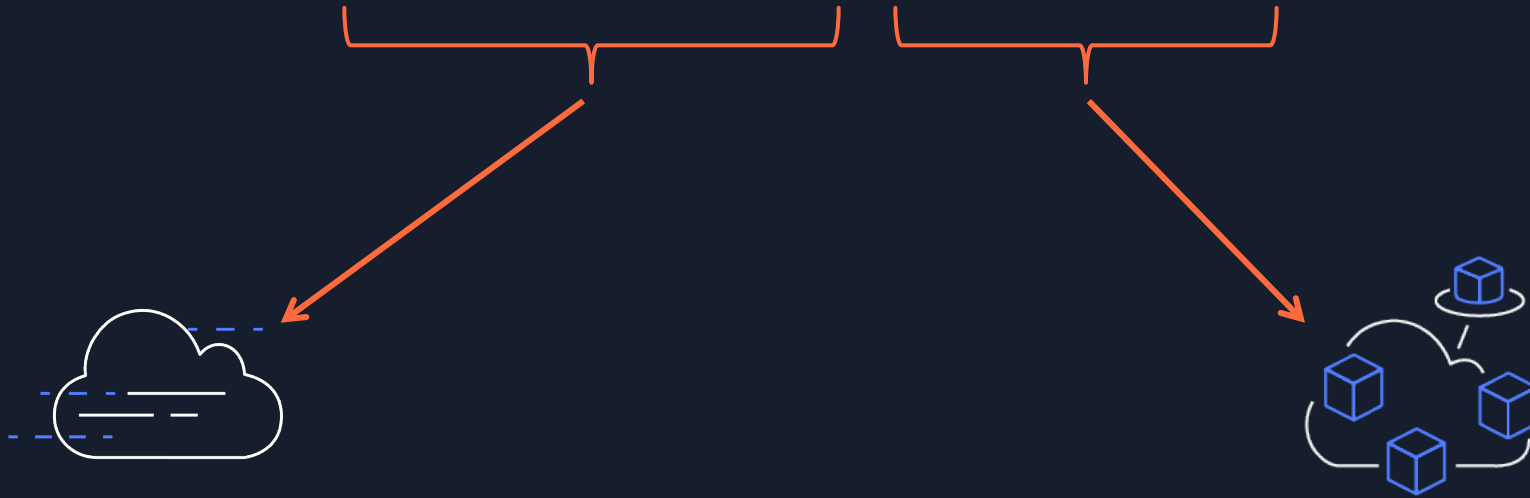


Managed by AWS:

- no cluster versions
- no data store to maintain or scale
- free to use with EC2 or Fargate

Works with different
“compute” types, including
AWS Fargate

AWS Fargate: serverless compute engine for containers



Managed by AWS

- no AMIs to maintain
- no Amazon instances to provision, scale or manage
- pay for what you need

Fargate is where your containers run;
you don't interact with it directly

Amazon ECS and AWS Fargate:

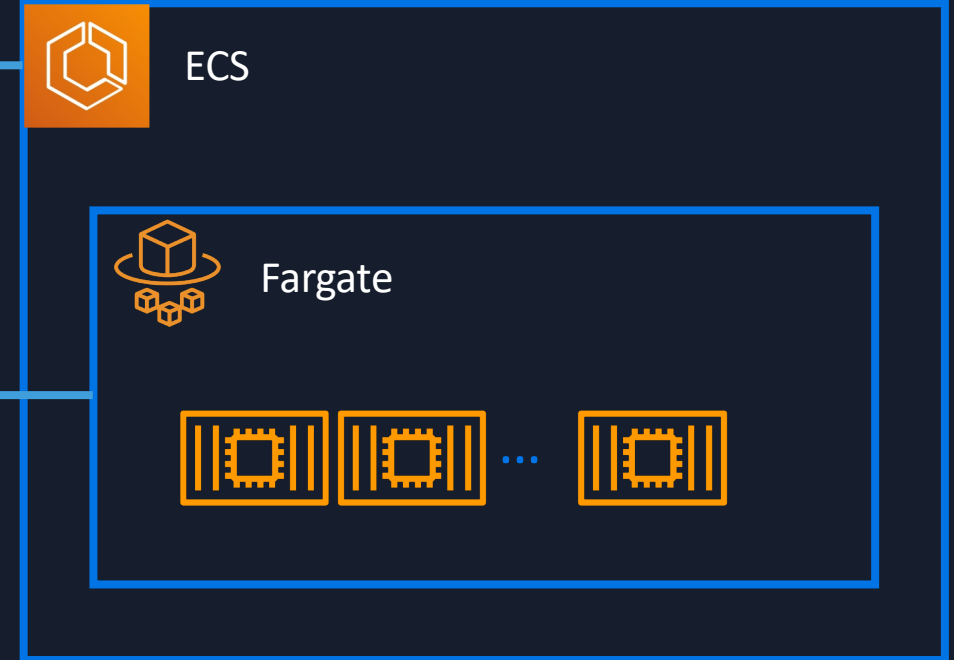
FULLY MANAGED CONTROL & DATA PLANE

Amazon ECS

- Fully managed control plane

AWS Fargate

- Fully managed data plane





Amazon ECS

Fully managed container
orchestration

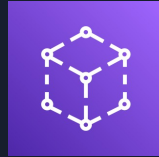
ECS built-in integrations with AWS services



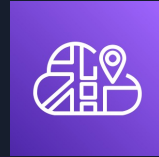
Amazon Elastic
Container Registry



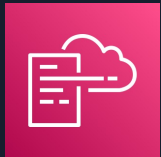
AWS CodeDeploy



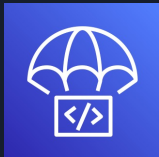
AWS
App Mesh



AWS
Cloud Map



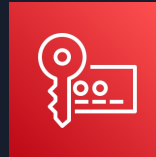
AWS CloudFormation



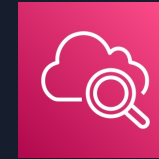
AWS CodeBuild



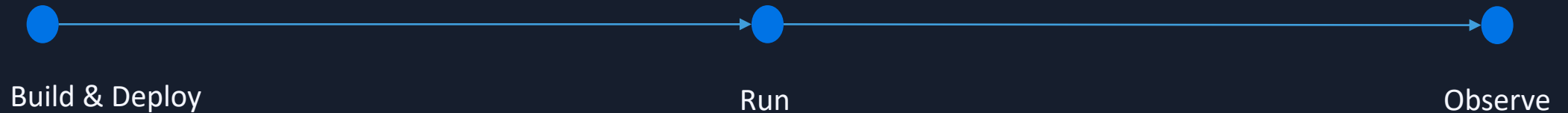
AWS
Secrets Manager



AWS Key
Management
Service



Amazon
CloudWatch



ECS core constructs

Task

- One or more containers
- Specify compute, networking, IAM, config

Service

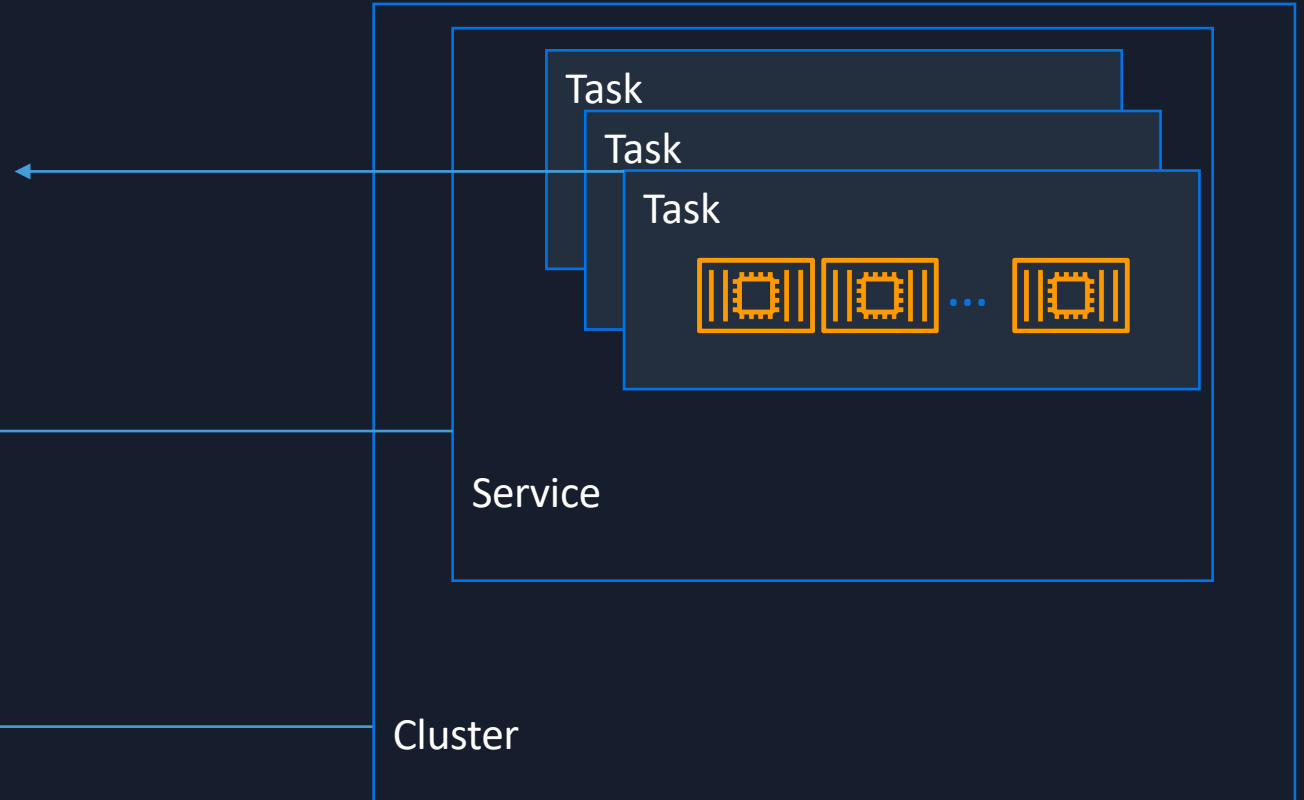
- One or more identical tasks
- Check and replace unhealthy tasks

Cluster

Logical grouping of services, tasks, capacity providers in a region

Compute

Use EC2, EC2 Spot, Fargate, or Fargate Spot



Deploying on ECS: Tasks vs services

On-demand workloads

ECS task scheduler

Run once or at intervals

Batch jobs

RunTask API

Long-running apps

ECS service scheduler

Health management

Scale-up and scale-down

AZ aware

Grouped containers

Task definition

```
{
  "containerDefinitions": [
    {
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "name": "nginx-container",
      "image": "nginx",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "ecs-log-streaming",
          "awslogs-region": "us-west-2",
          "awslogs-stream-prefix": "fargate-task-1"
        }
      }
    },
  ],
```

continued...

```
        "cpu": 0
      }
    ],
    "networkMode": "awsvpc",
    "executionRoleArn":
"arn:aws:iam::123456789012:role/ecsTask
ExecutionRole",
    "memory": "2048",
    "cpu": "1024",
    "requiresCompatibilities": [
      "FARGATE"
    ],
    "family": "example_task_1"
  }
}
```

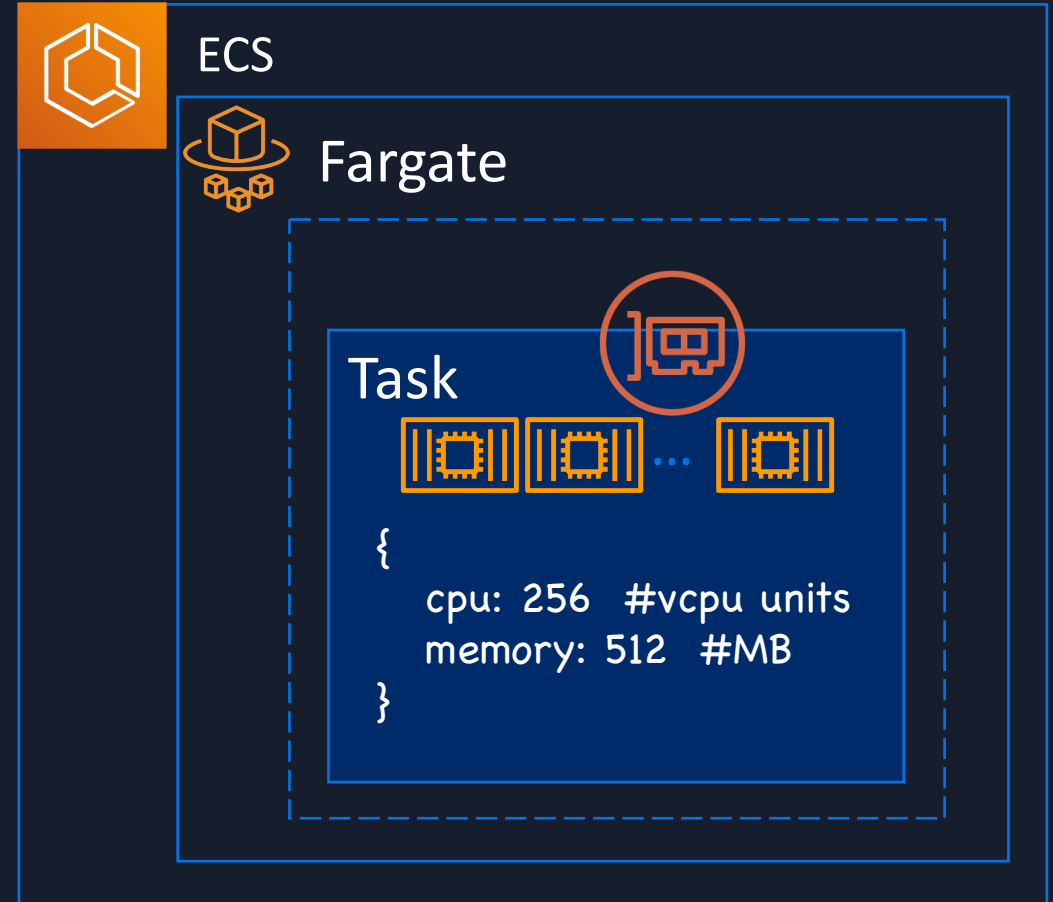


AWS Fargate

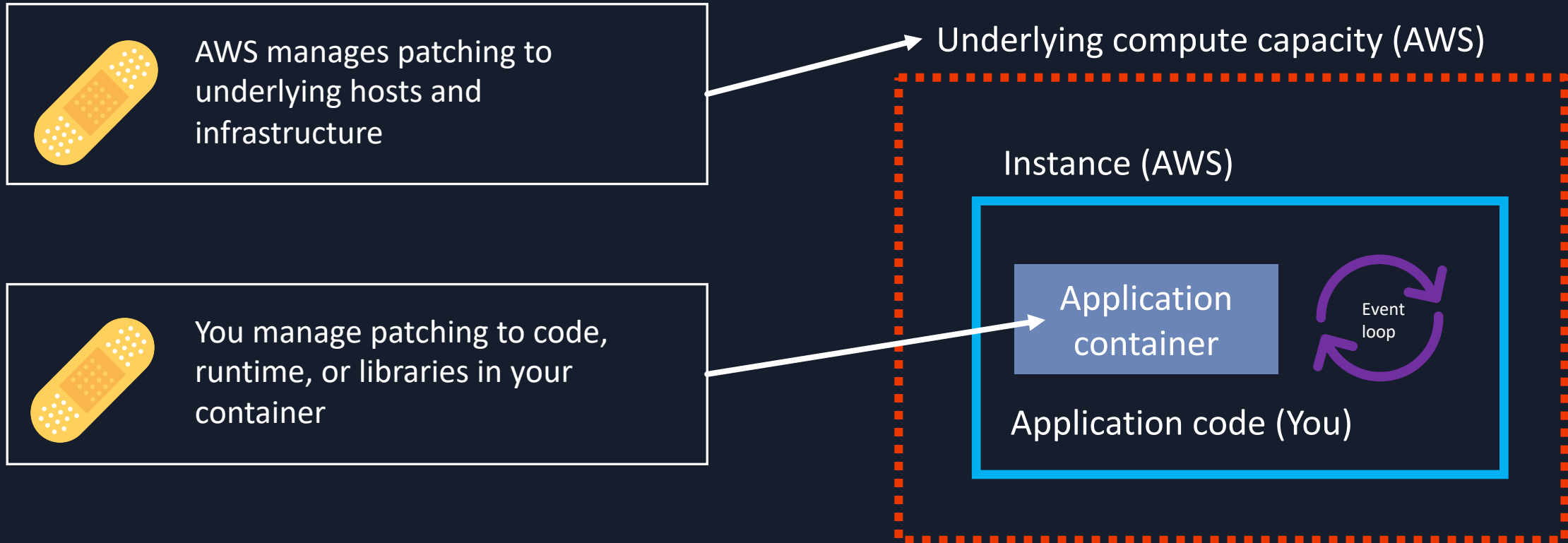
Serverless compute
engine for containers

How AWS Fargate serverless containers work

- Fargate runs each Task in **new, dynamically provisioned instance**
- Resources based on **Task vCPU and RAM**
- Task vCPU and RAM is for **all containers in the task**

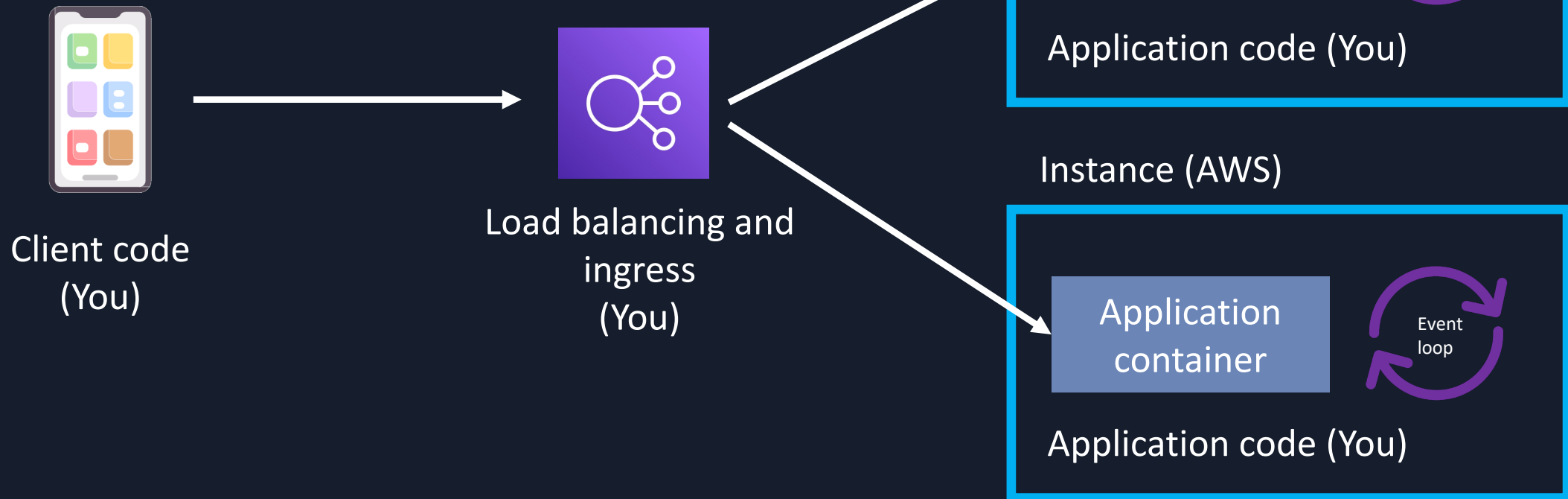


AWS Fargate patching



AWS Fargate load balancing and ingress

AWS = Fully managed by AWS
You = Your responsibility to manage



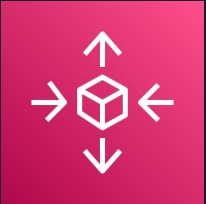
AWS Fargate scaling



AWS provides underlying compute capacity to make sure you can launch containers on demand



You decide how many containers you need and what triggers scaling for those containers

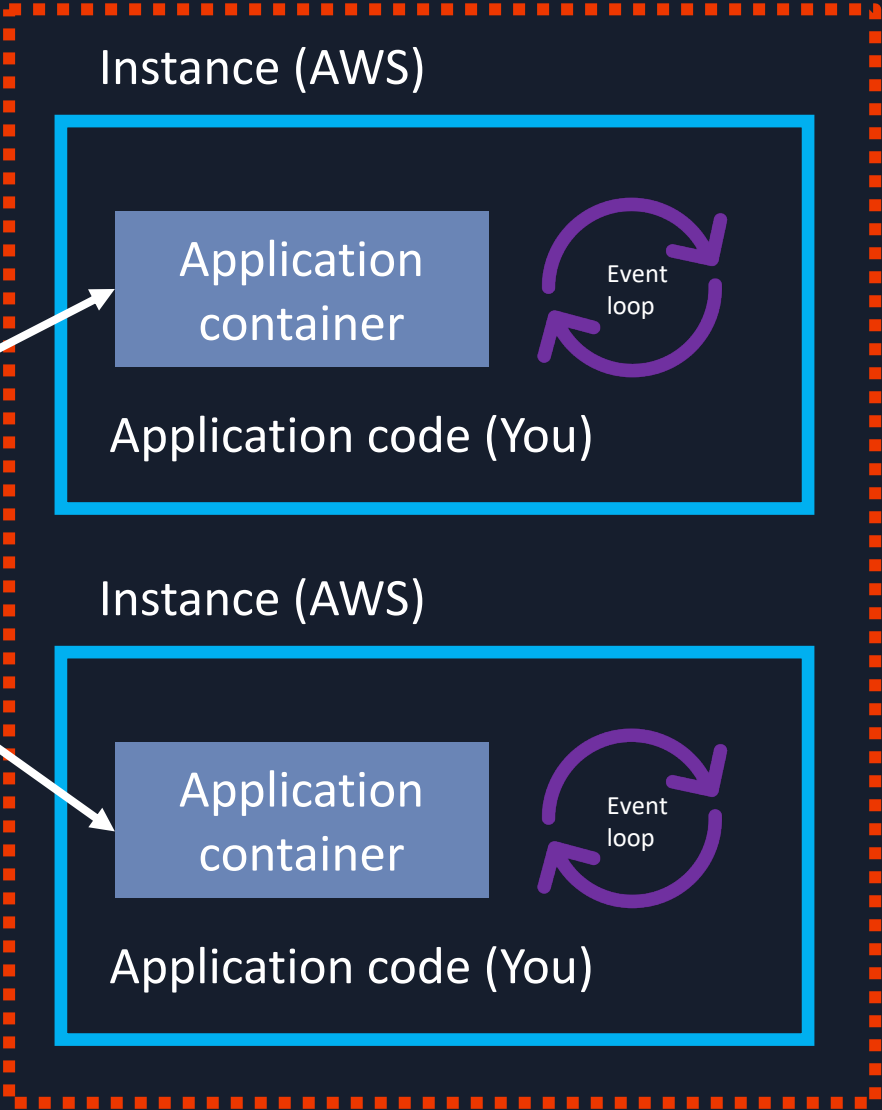


AWS Auto Scaling



Amazon CloudWatch

Underlying compute capacity (AWS)

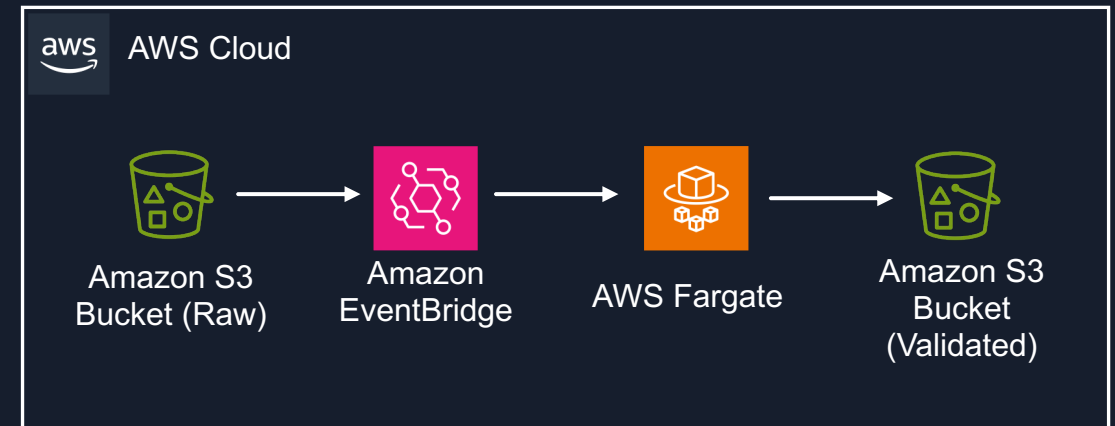


EDA patterns using containers

ECS event-driven architectures

Task per event

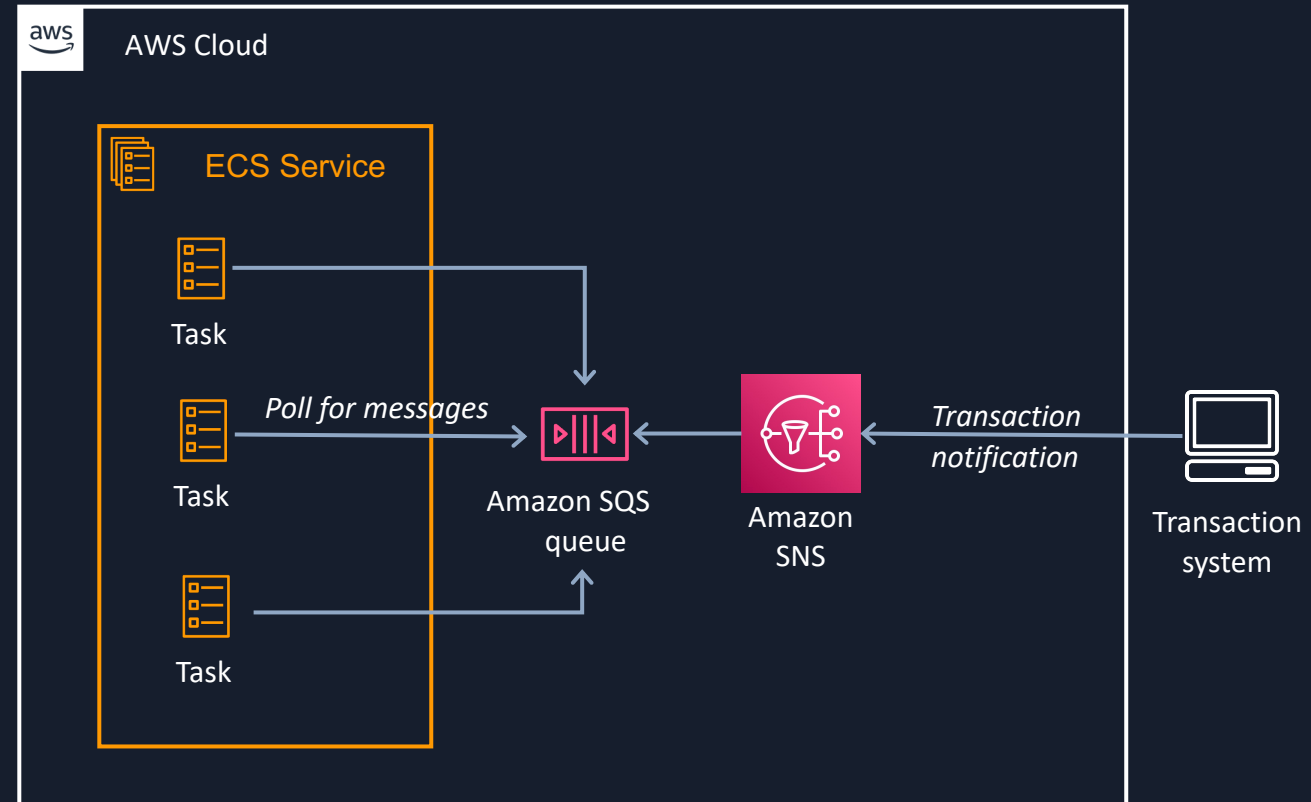
- Event triggers ephemeral task to processes a single event
- Requires EventBridge or Step Functions to invoke the task
- Less frequent events and/or longer execution



ECS event-driven architectures

Event processing service

- Long-running, resilient Service that polls and processes events
- ECS-managed deployments and configurable autoscaling
- Common event sources: SQS, Kinesis, MSK, etc
- High-frequency events



Monitoring transactions with ECS & SQS

Event-driven scaling using AWS CDK ECS Patterns



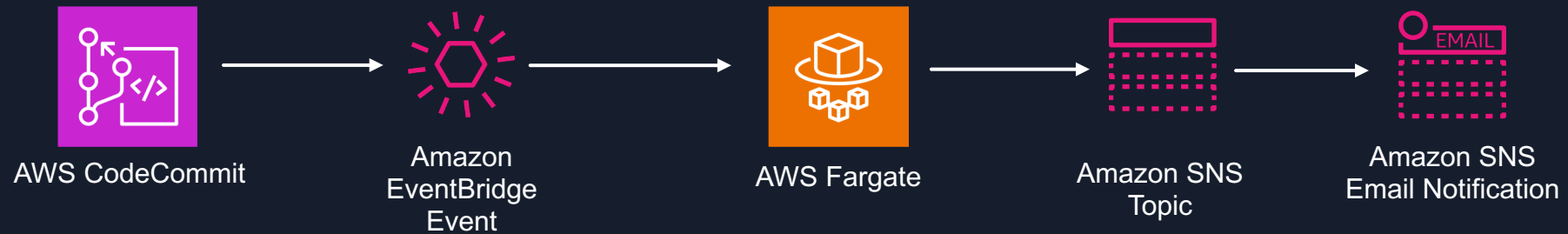
CDK L3 construct

Scaling steps based on
ApproximateNumberOfMessagesVisible
metric from SQS Queue

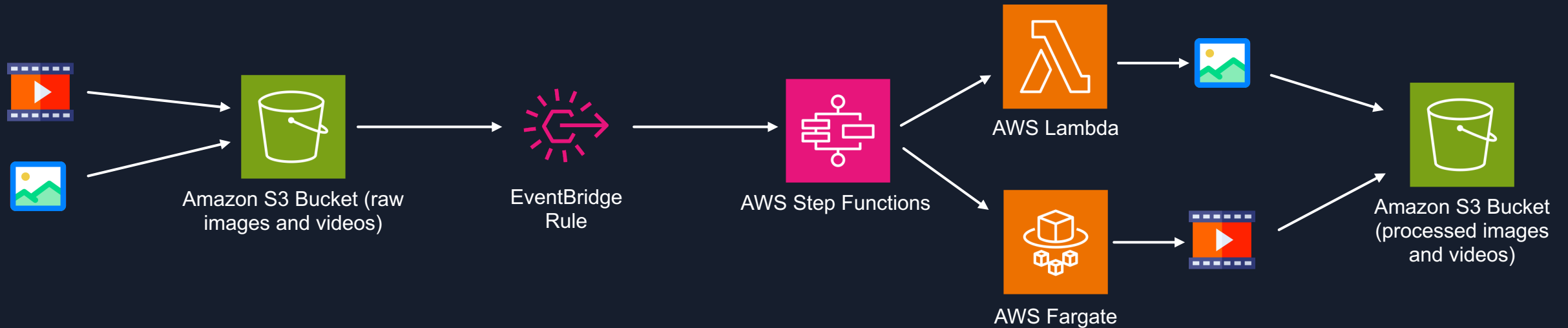
Distribute between on-demand or spot

```
1 new ecs_patterns.QueueProcessingFargateService(this, 'Service', {
2     cluster,
3     memoryLimitMiB: 1024,
4     cpu: 512,
5     queue: queue,
6     image: ecs.ContainerImage.fromDockerImageAsset(asset),
7     minScalingCapacity: 1,
8     maxScalingCapacity: 5,
9     scalingSteps: [
10        { upper: 10, change: 0 },
11        { lower: 5, change: 1 }
12    ],
13    environment: {
14        "SQS_ENDPOINT_URL": queue.queueUrl,
15        "EVENTBUS_NAME": props?.bus.eventBusName,
16        "DYNAMODB_TABLE_NAME": this.table.tableName
17    },
18    capacityProviderStrategies: [
19        {
20            capacityProvider: 'FARGATE_SPOT',
21            weight: 2,
22        },
23        {
24            capacityProvider: 'FARGATE',
25            weight: 1,
26        },
27    ],
28 });
```

Use case: : Code scanning



Use case: File processing



Demo: File processing - Lambda and Fargate



Which serverless compute option is right for my application?

Architectural considerations

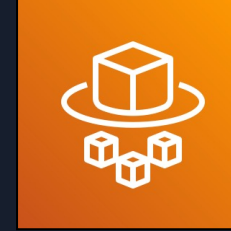
Functions



AWS Lambda

- Managed triggers and runtimes
- Security and scaling managed by AWS

Containers



AWS Fargate

- Custom code and services
- Requires management and orchestration

Resource considerations

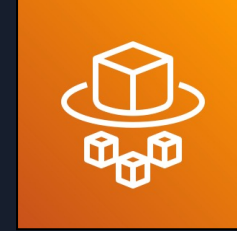
Functions



AWS Lambda

- 128 MB to 10 GB memory
- Proportional network and CPU
- Max 15 minutes
- Single concurrency

Containers

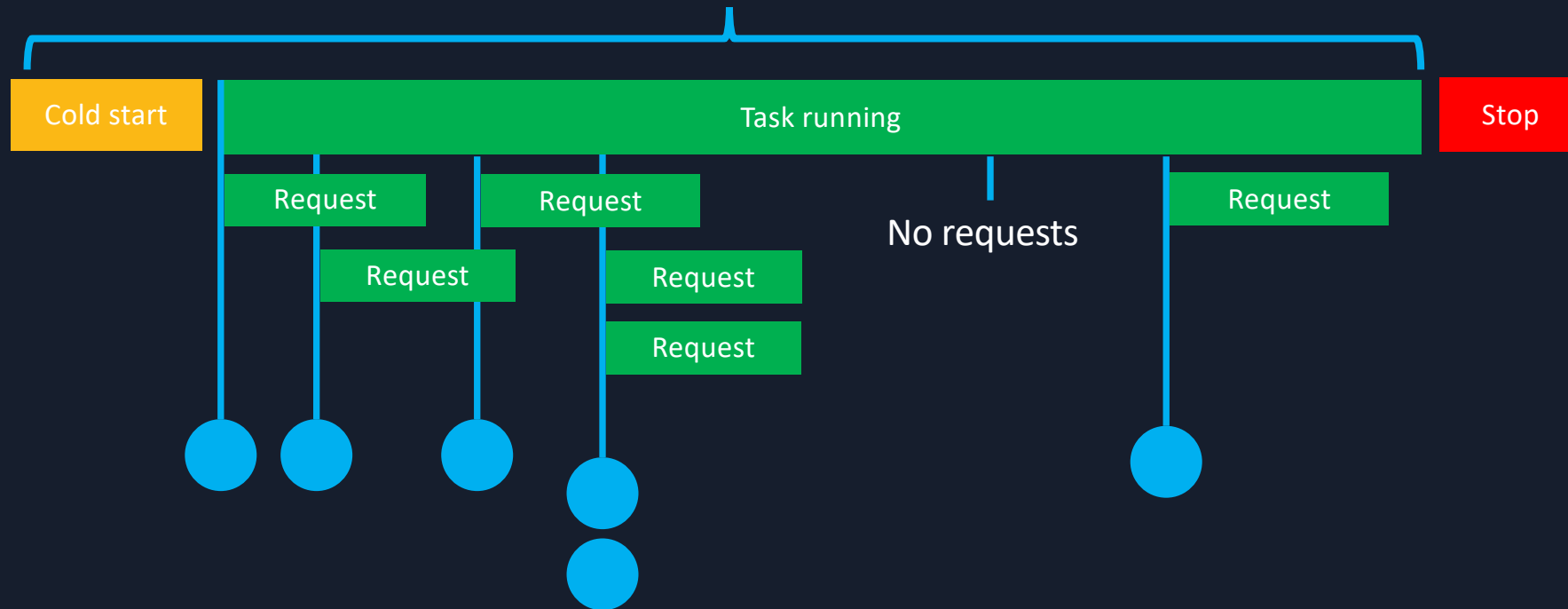


AWS Fargate

- .25–16 vCPU, 512 MiB–120 GB
- Multiple networking modes
- Unlimited running time
- Multi concurrency

AWS Fargate: Pay for what you provision

Pay a constant rate per second for the task CPU and memory

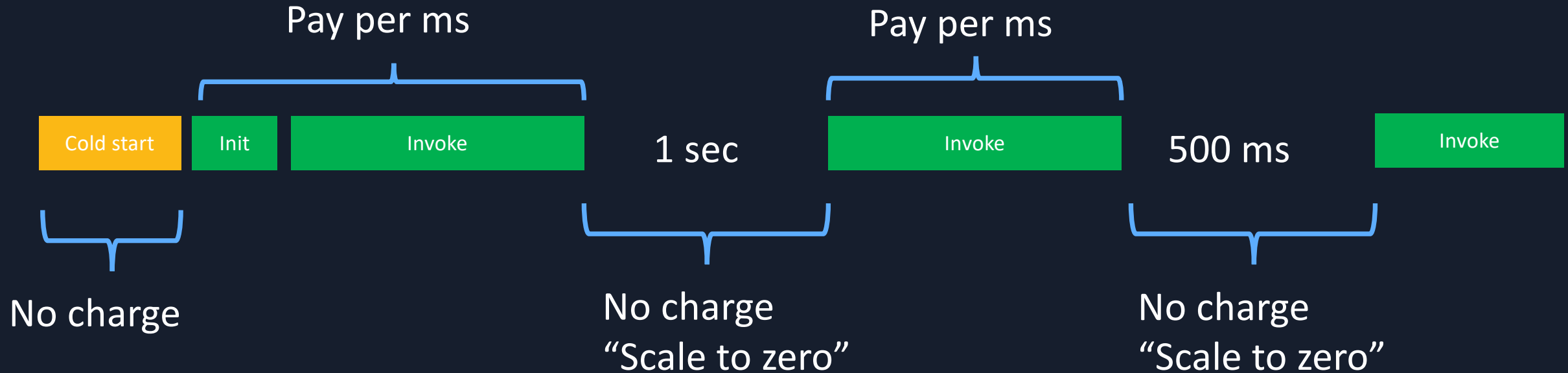


Per container instance per second, based on CPU and memory

Constant price, can use Fargate Spot

Lowest per-request cost when concurrent traffic is high and code is efficient

AWS Lambda: Pay for what you use



Per ms, per invoke, based on memory size of function

Each invoke charged separately

Cost effective when there are gaps between work to do

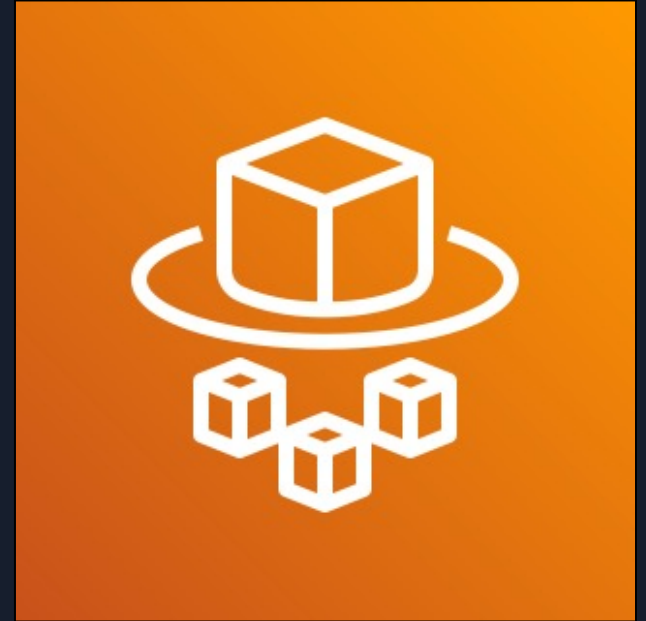
Ideas on selecting . . . serverless containers

When you need . . .

- Low startup latency
- Long-running compute jobs (> 15 minutes)
- Predictable, high-traffic usage
- Data persistence

When you want . . .

- Control of compute environment
- Tooling consistency



Ideas on selecting . . . serverless functions

When you need . . .

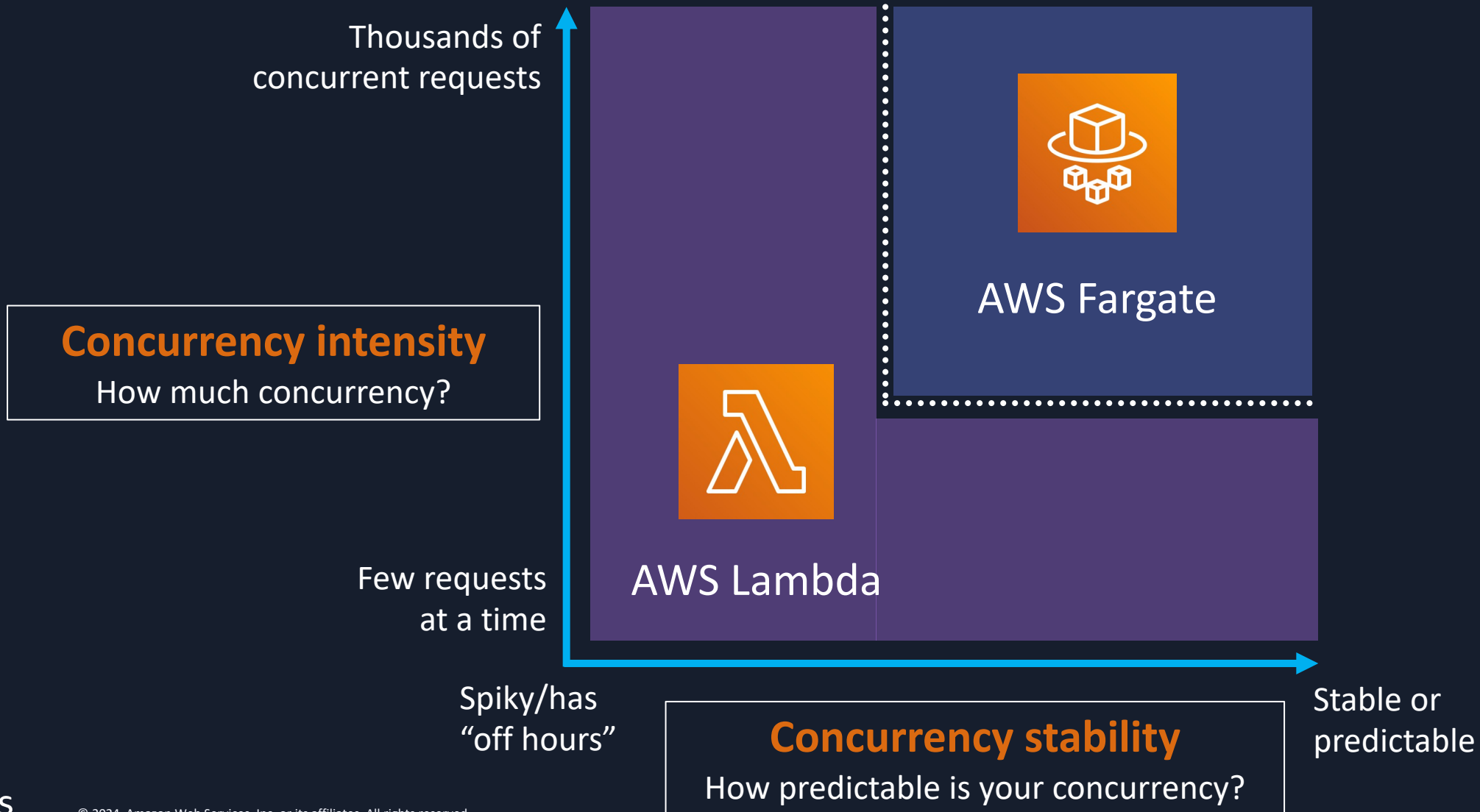
- Immediate startup time
- To trigger action on an event
- Support for varying utilisation
- Ability to handle unknown demand

When you want to . . .

- Make fewer decisions
- Faster prototype to production
- Hand operational complexity (for example, patching, scaling) to AWS



Serverless compute sweet spots



OK, simpler, please!

ECS/Fargate:

Long running, stable, high-volume, high
resource event sources

Lambda:

For most other event sources

The road ahead

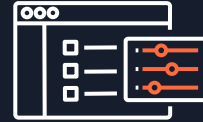
CONTINUED INVESTMENTS ACROSS THE SPECTRUM TO HELP CUSTOMERS BECOME MORE PRODUCTIVE



AWS Well-
Architected
fundamentals



Developer
experience



Improved
controls
over services



Simpler
and broader
integrations

Thank you!

Julian Wood

 @julian_wood

 [linkedin.com/in/julianrwood](https://www.linkedin.com/in/julianrwood)

