



GOTO AMSTERDAM 2024

Serverless compute at the heart of your EDA

Julian Wood

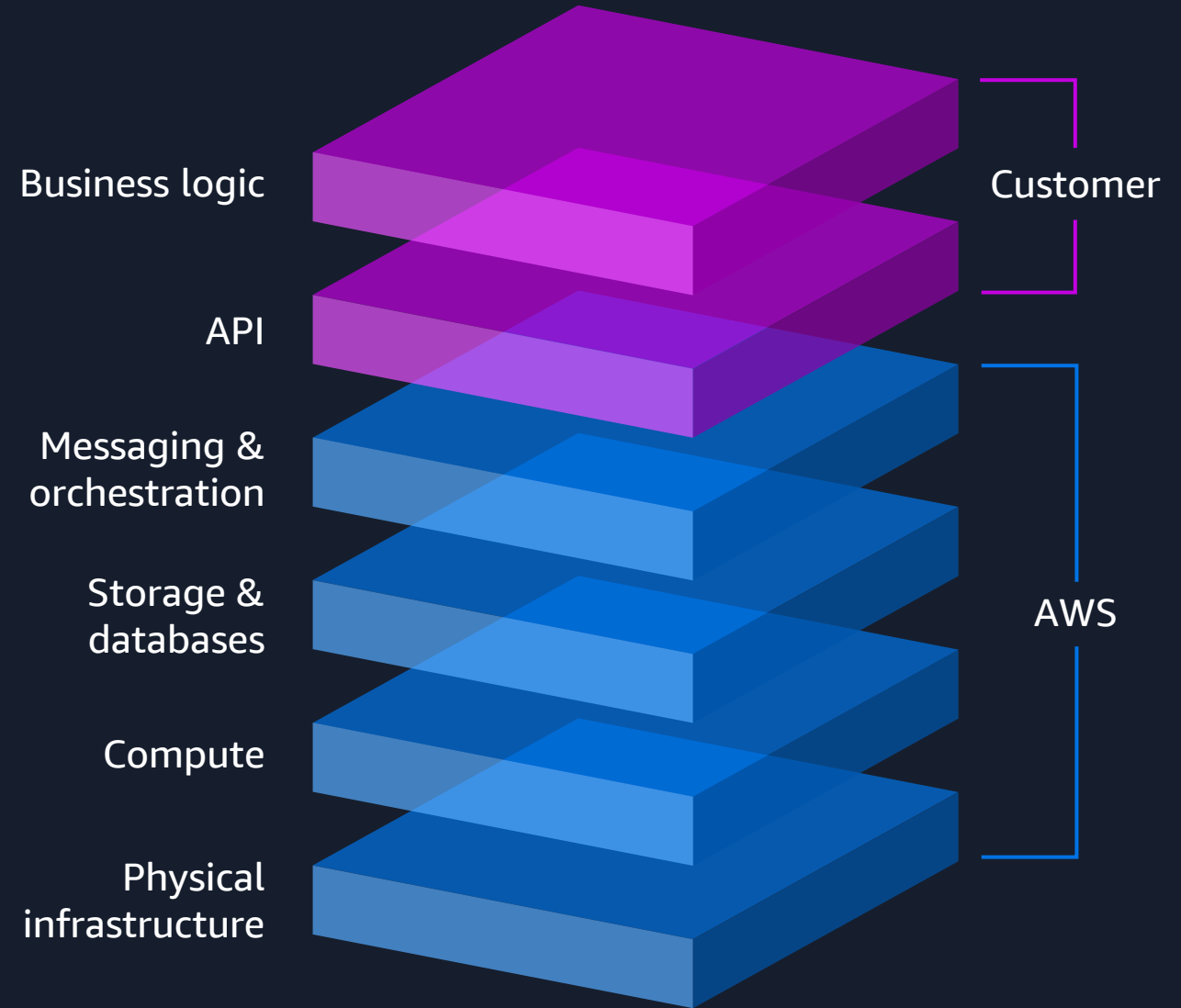
Principal Developer Advocate
AWS Serverless

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

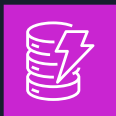


Amazon
Kinesis

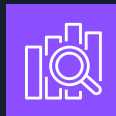
Databases and Analytics



Amazon Aurora
Serverless



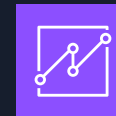
Amazon
DynamoDB



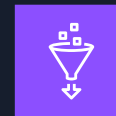
Amazon
OpenSearch Service



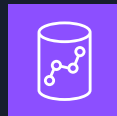
Amazon
Bedrock



Amazon
QuickSight



AWS
Glue

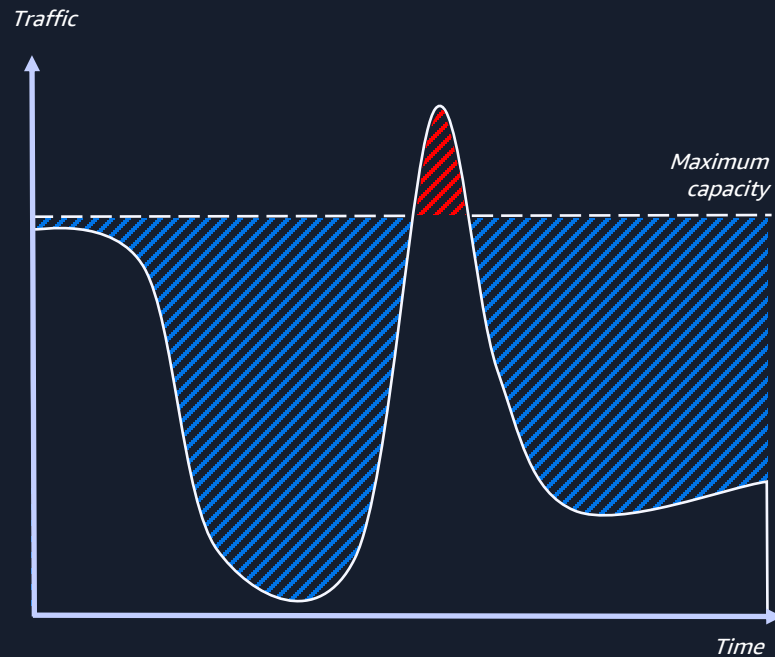


Amazon
Redshift

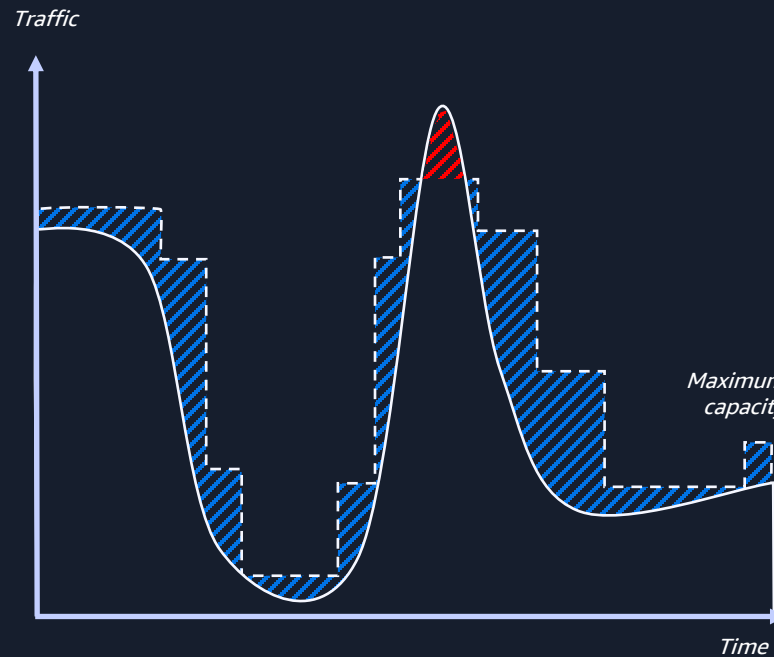


© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

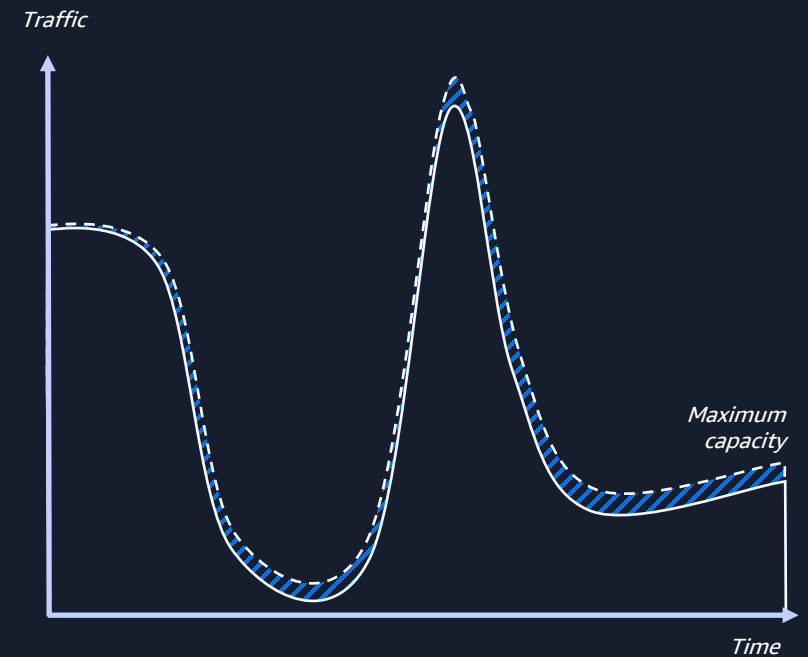
Utilization



Traditional/on-premises



Instance-based (EC2)



AWS Serverless

 Over capacity  Under capacity

Serverless at scale



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



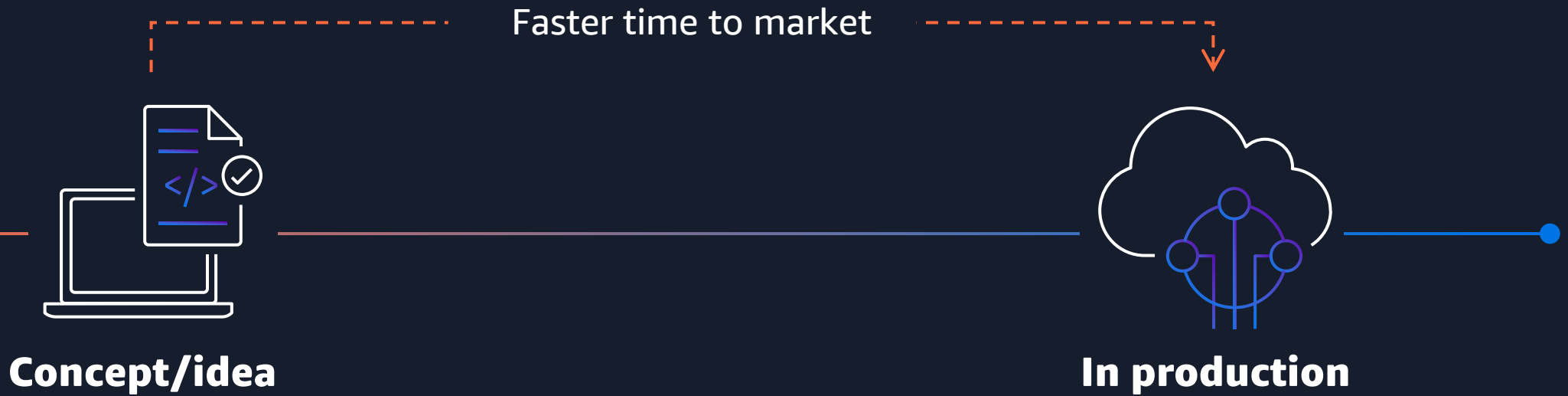
Average number of invokes on Amazon EventBridge is over **2 trillion per month**



Over 2.25 billion Amazon ECS tasks are launched each week

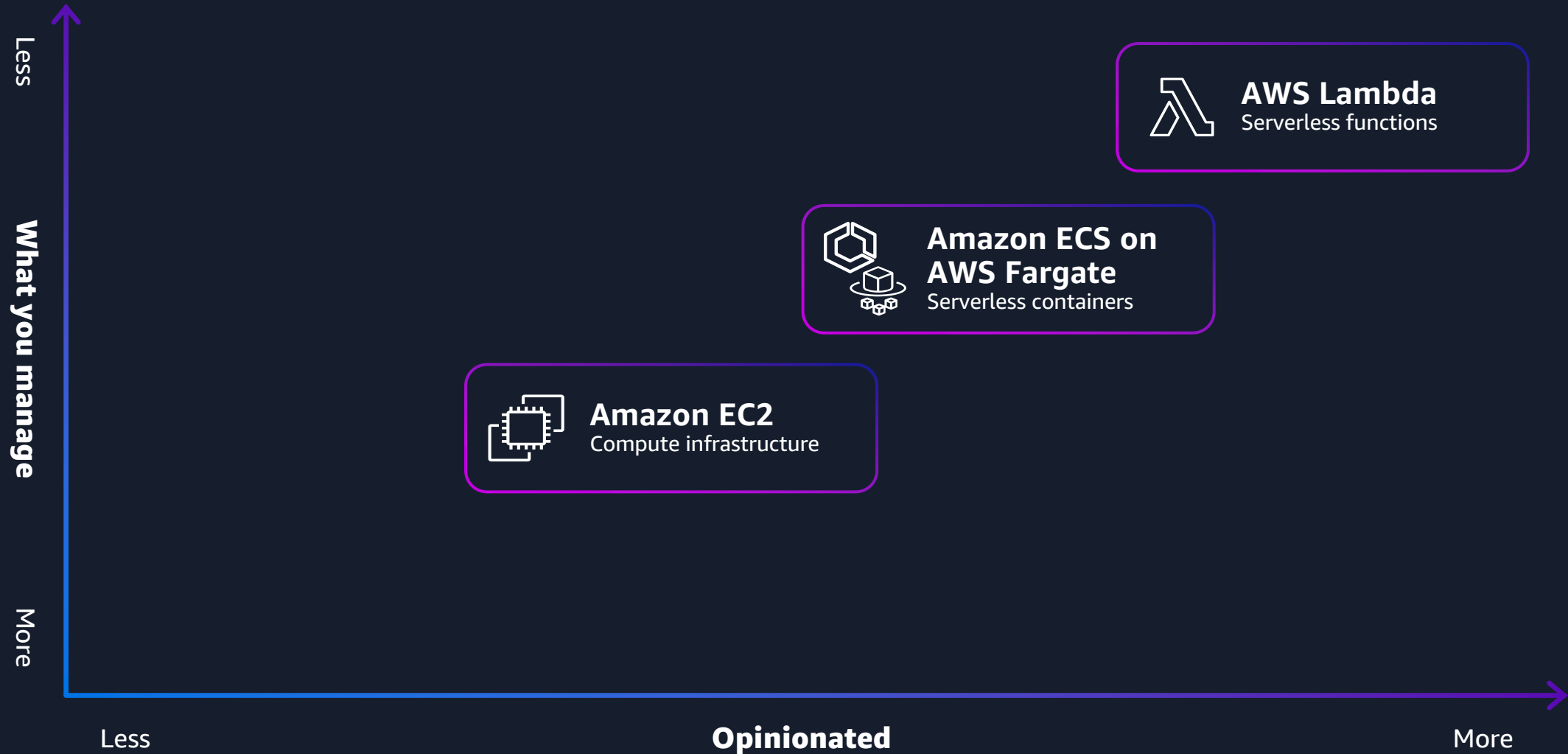
Why go serverless?

THE PHILOSOPHY BEHIND SERVERLESS AND SERVERLESS CONTAINERS



Serverless compute

The AWS compute landscape





Serverless functions **AWS Lambda**



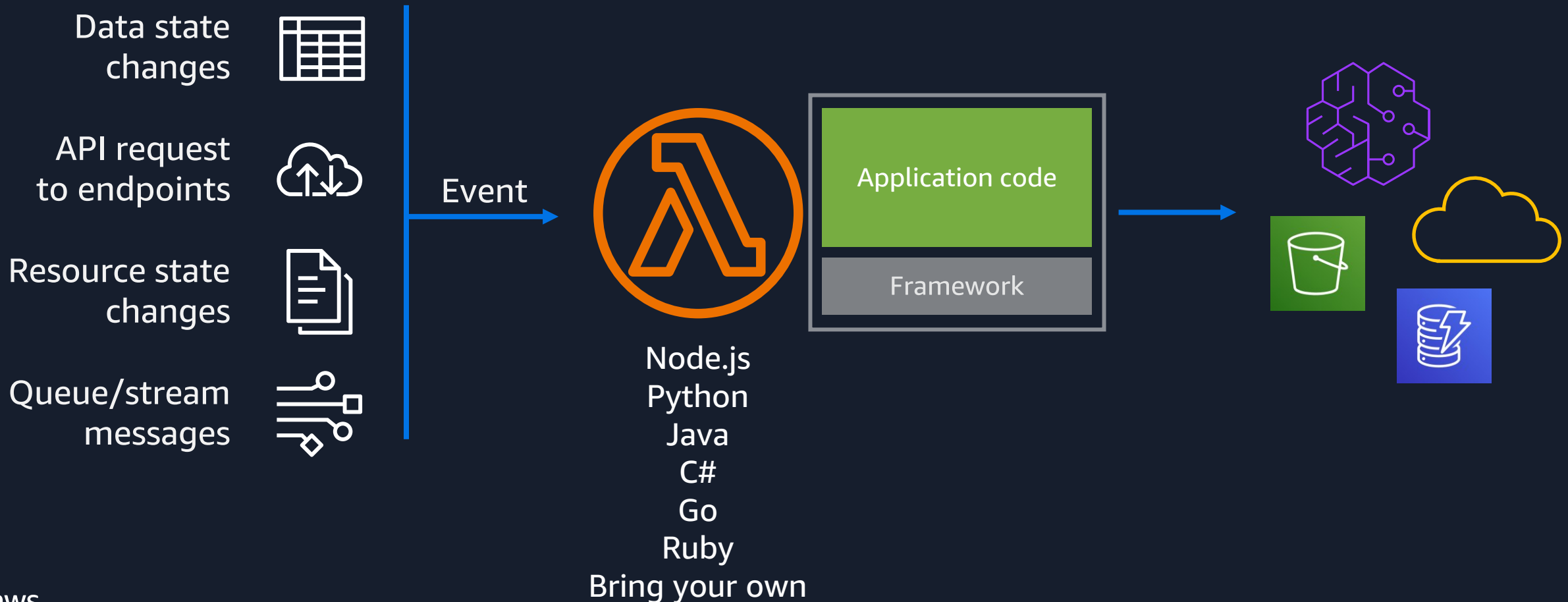
Serverless functions: AWS Lambda



Event Source

Function

Services / Other



The AWS Lambda invocation model

Three types of Lambda integrations:

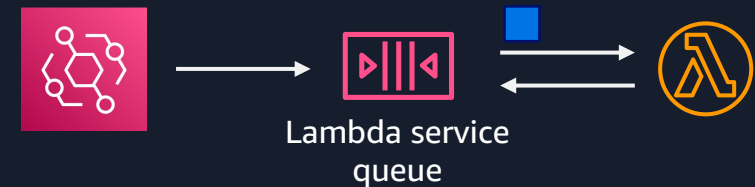
- **Synchronous**
 - Including request streaming with Node.js
- **Asynchronous**
- **Poll-based**
(event source mapping)

Functions run in secure, isolated execution environment

Synchronous



Asynchronous



Poll-based



AWS Lambda packaging



.zip archive



container image

AWS Lambda container image

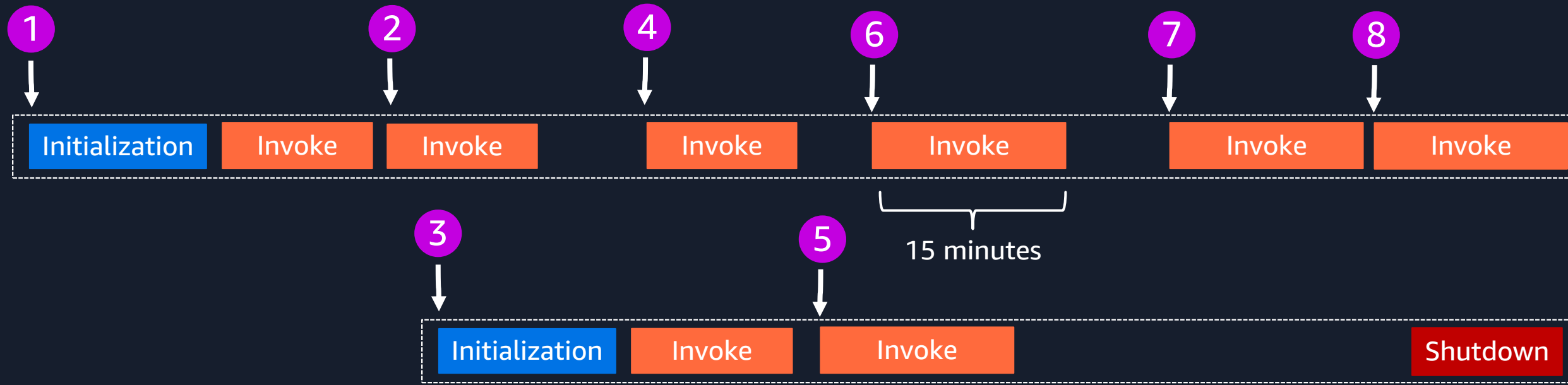


```
$ cat Dockerfile
FROM public.ecr.aws/lambda/nodejs:20
COPY app.js package*.json ./
RUN npm install
CMD [ "app.lambdaHandler" ]
```

AWS Lambda function scaling



- Scaling per request: from 0 to x
- Short lived (up to 15 min) and ephemeral
- Execution environments are shut down after inactivity



Serverless functions in action



AWS Lambda function scaling

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

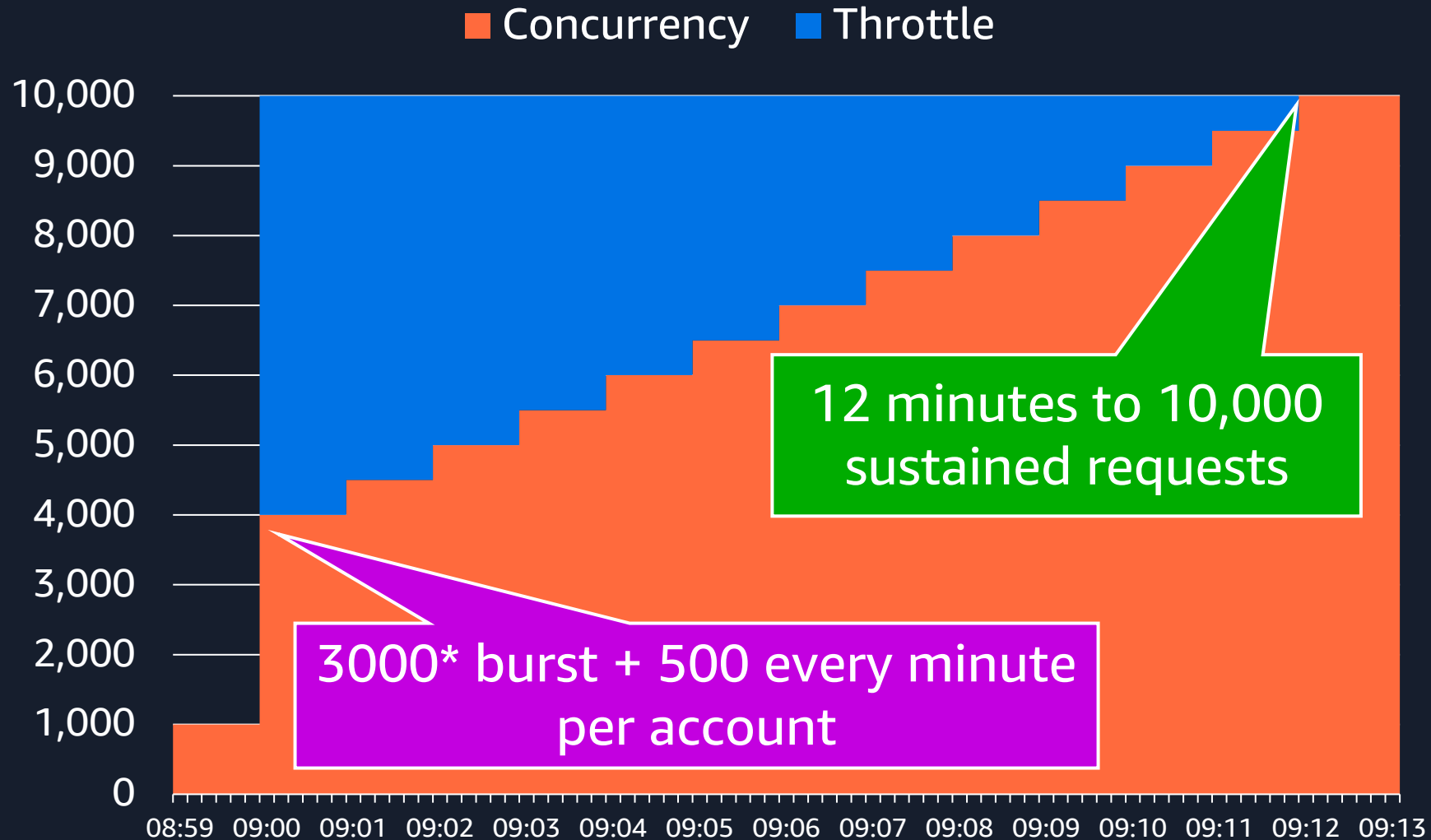
Suited for applications **with varying traffic patterns**

Scale automatically without managing the complexity of infrastructure

Within account concurrency quota

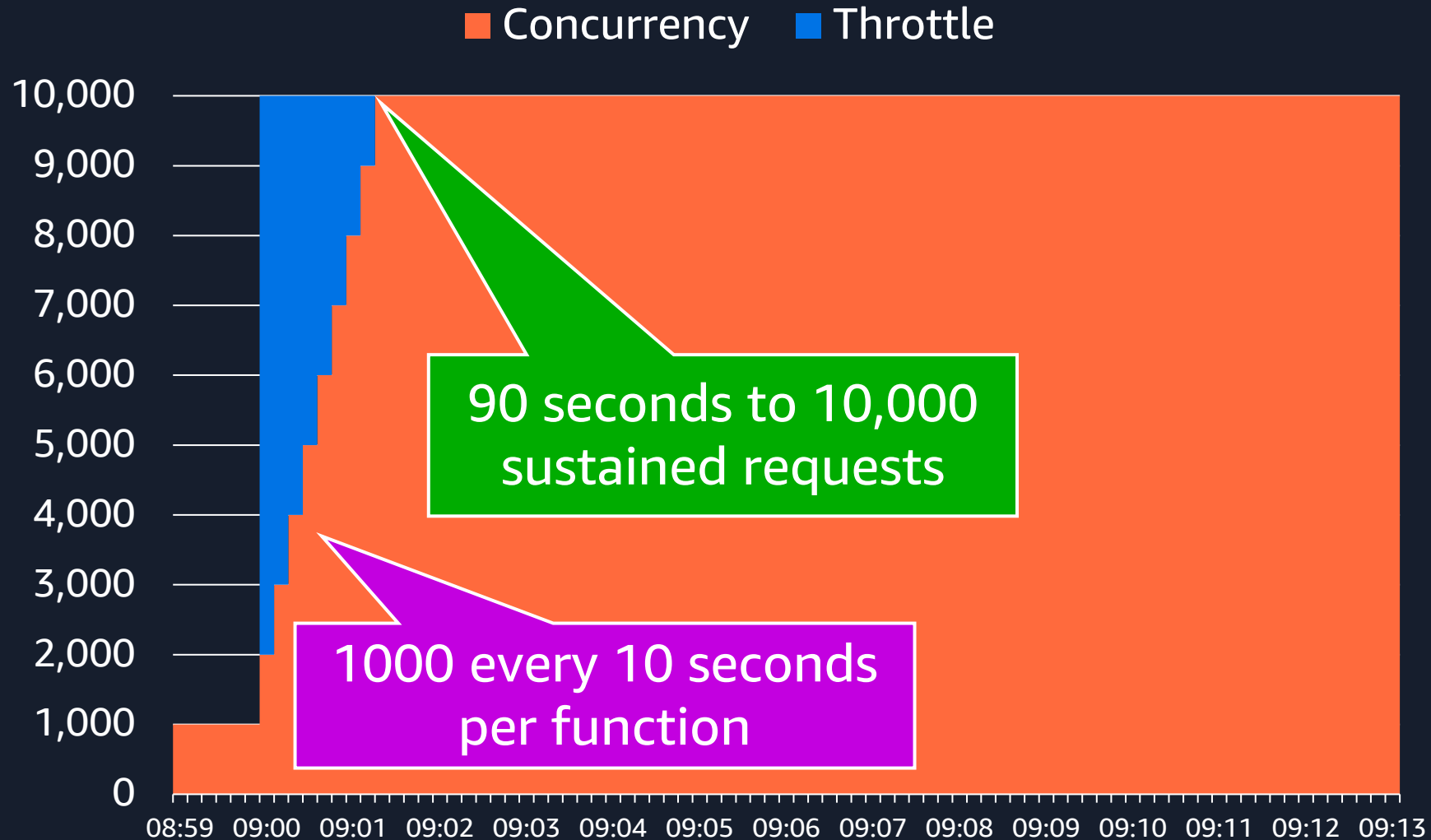


Previous concurrency scaling



New concurrency scaling

NEW



Serverless functions in action



Serverless containers

Amazon ECS

Fully managed container orchestration



AWS Fargate

Serverless container compute engine



Amazon ECS and AWS Fargate:

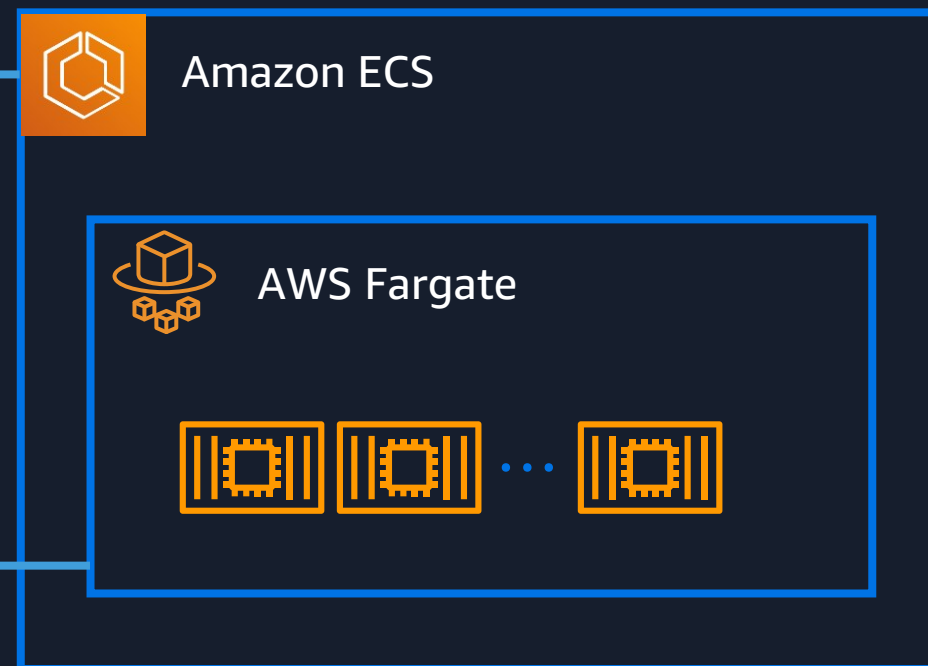
FULLY MANAGED CONTROL & DATA PLANE

Amazon ECS

- Fully managed control plane
- Free to use (pay for underlying compute resources)
- no cluster versions
- no data store to maintain or scale

AWS Fargate

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





Amazon ECS

Fully managed
container orchestration

ECS built-in integrations with AWS services



Amazon Elastic
Container Registry



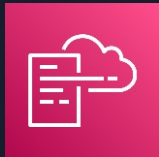
AWS
CodeDeploy



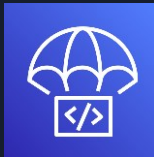
AWS
App Mesh



AWS
Cloud Map



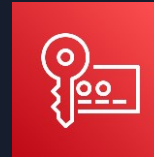
AWS
CloudFormation



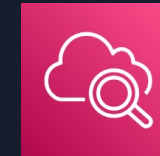
AWS
CodeBuild



AWS
Secrets Manager



AWS Key
Management
Service



Amazon
CloudWatch



Build & Deploy



Run



Observe

ECS core constructs

Task

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

Service

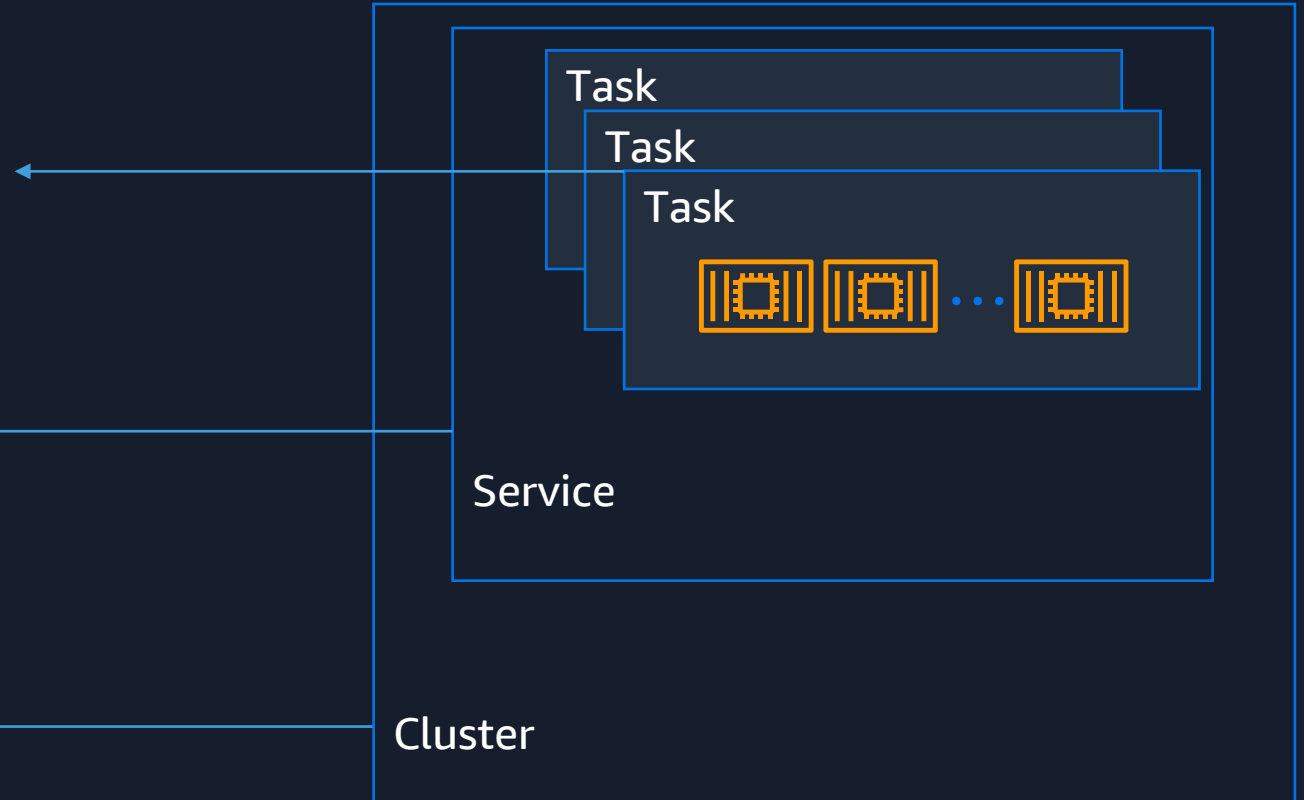
- One or more identical tasks
- Check and replace unhealthy tasks

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

Tasks:

On-demand workloads

ECS task scheduler

Run once or at intervals

Batch jobs

RunTask API

Services:

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"
  }
}
```

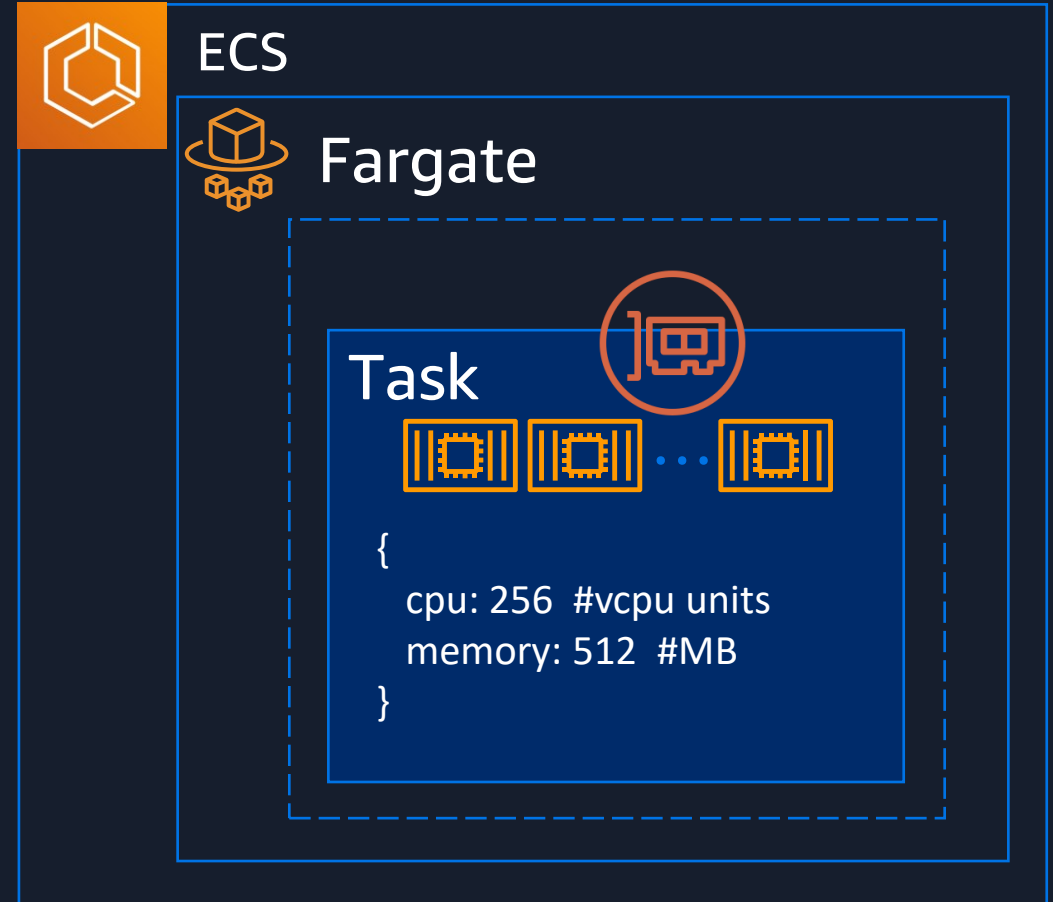


AWS Fargate

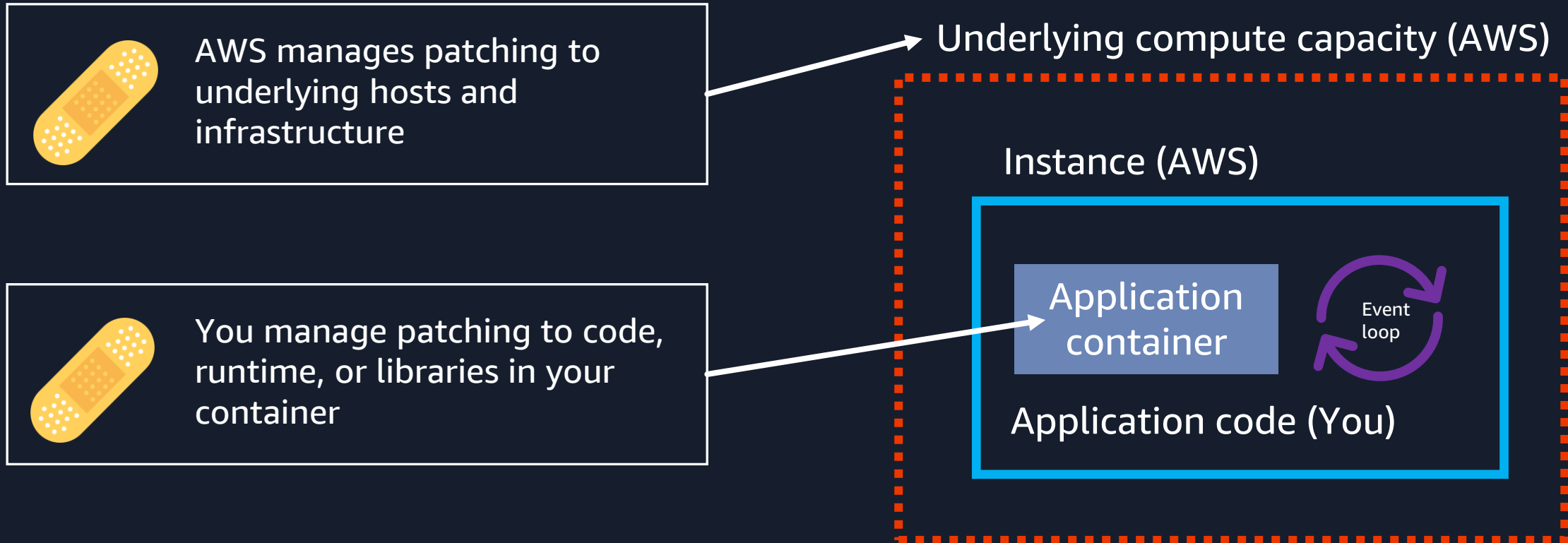
Serverless compute
engine for containers

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**

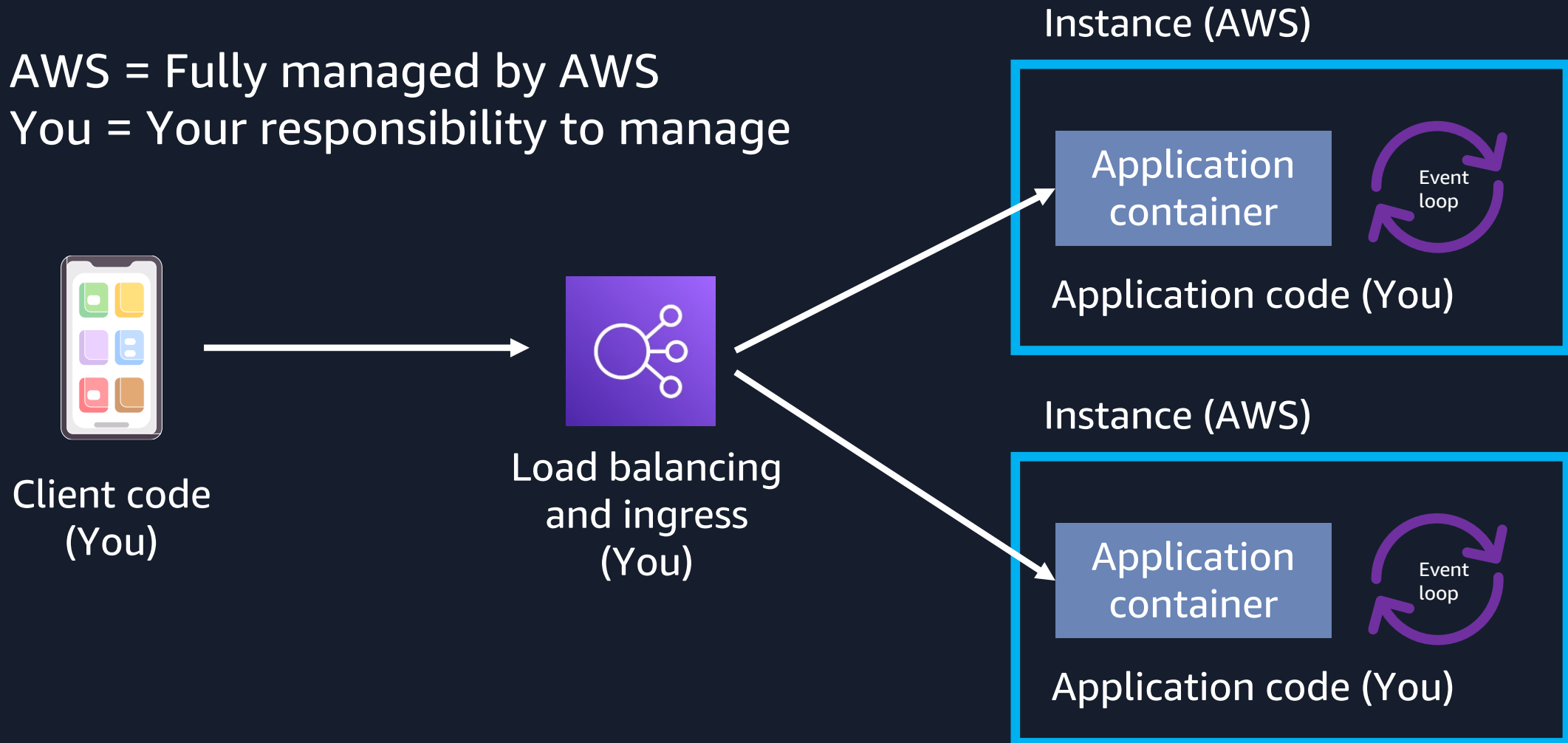


AWS Fargate patching



AWS Fargate load balancing and ingress

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



AWS Fargate scaling



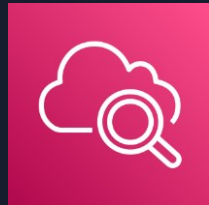
AWS provides underlying compute capacity to make sure you can launch containers on demand



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



AWS Auto Scaling



Amazon CloudWatch

Underlying compute capacity (AWS)

Instance (AWS)

Application container



Application code (You)

Instance (AWS)

Application container



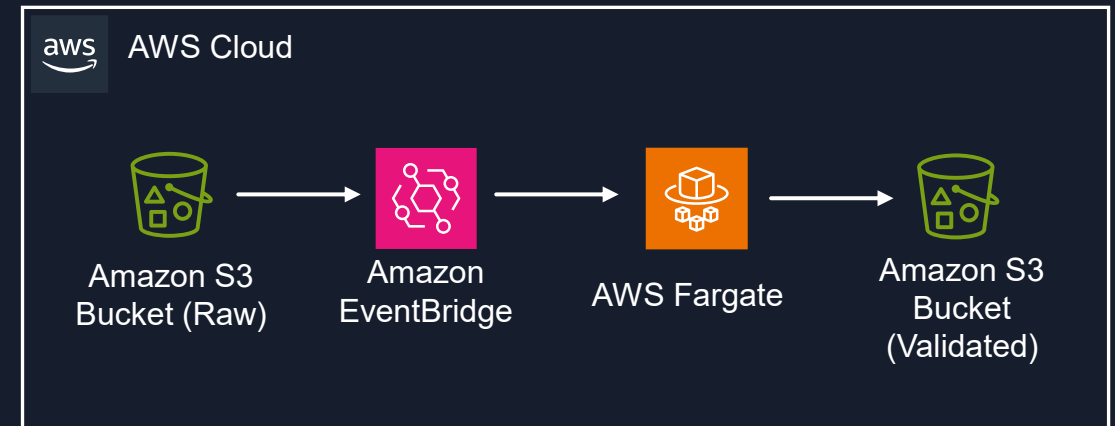
Application code (You)

EDA patterns using containers

ECS event-driven architectures

Task per event

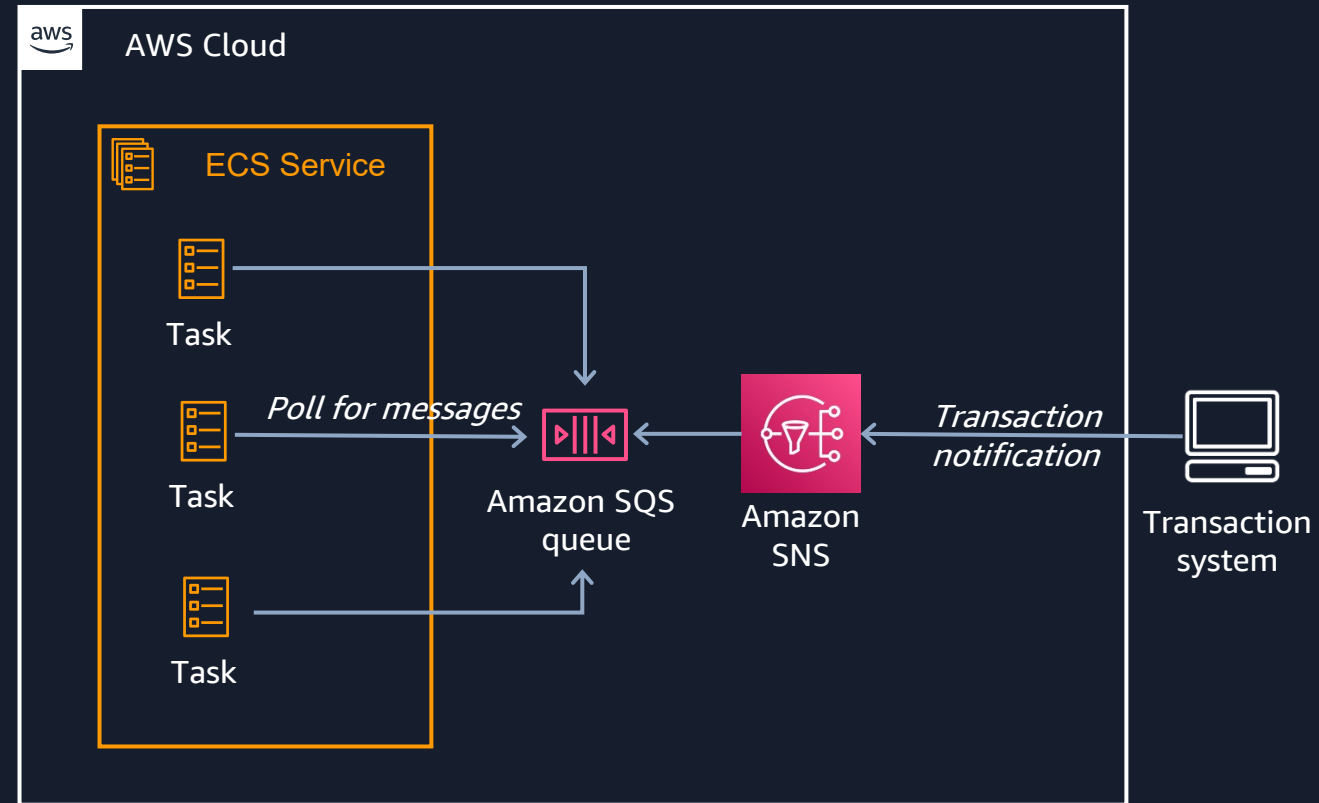
- Event triggers ephemeral task to processes a single event
- Requires EventBridge or Step Functions to invoke the task
- Less frequent events and/or longer execution



ECS event-driven architectures

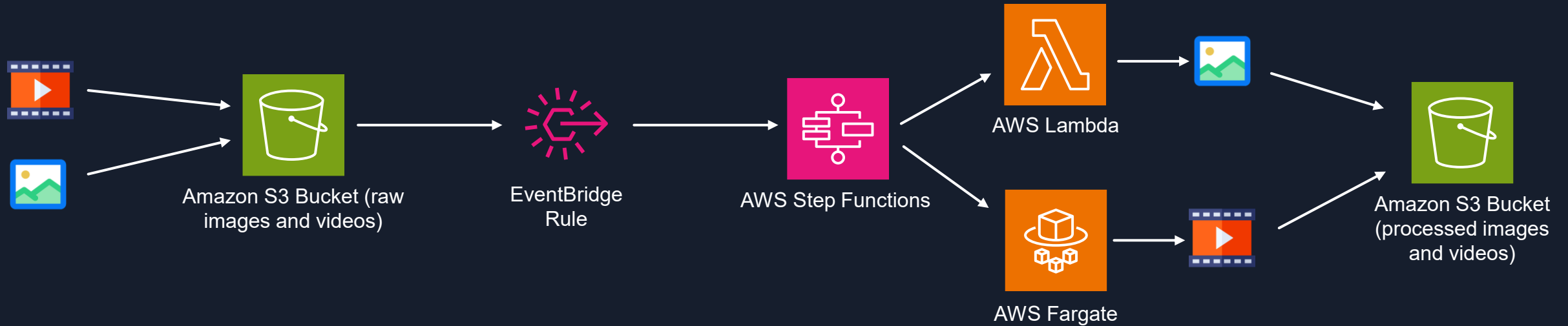
Event processing service

- 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

Use case: File processing

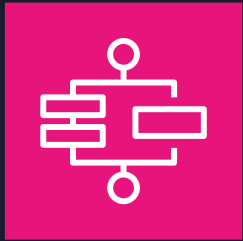




Serverless event bus service:
AWS services
Your own applications
SaaS providers

**Choreographs events between
application domains**

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

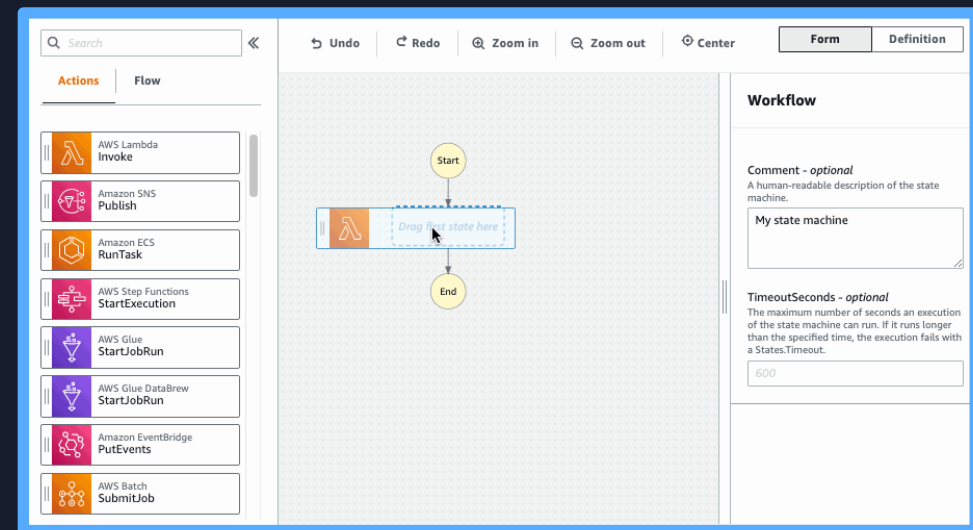


AWS Step Functions

Serverless workflow service.

Orchestrate multiple services

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



Orchestration/choreography as configuration

Orchestration



AWS Step Functions

Choreography



Amazon EventBridge

Serverless containers in action



Which serverless compute option is right for my application?

Architectural considerations

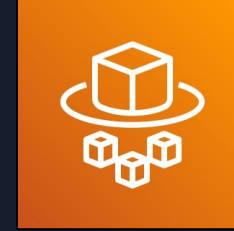
Functions



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

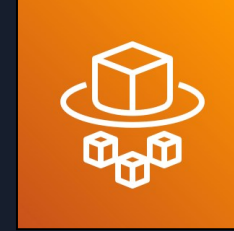
Functions



AWS Lambda

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

Containers

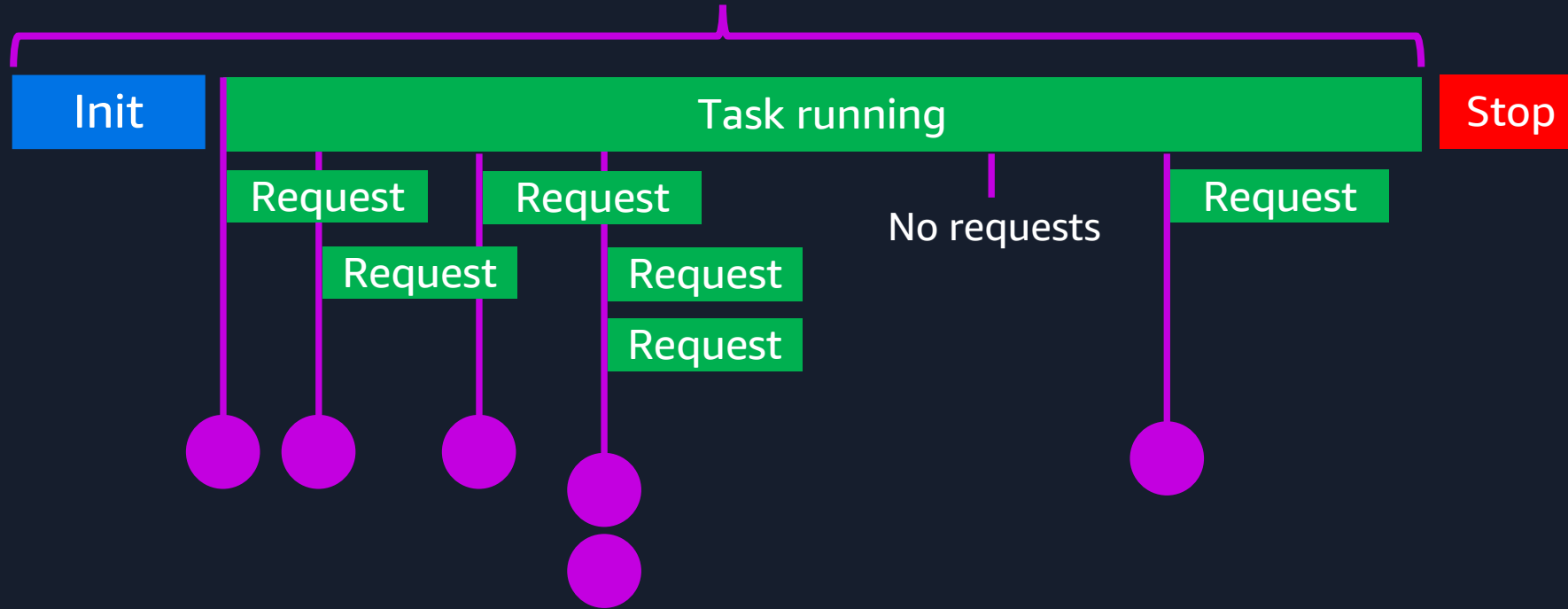


AWS Fargate

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

AWS Fargate: Pay for what you provision

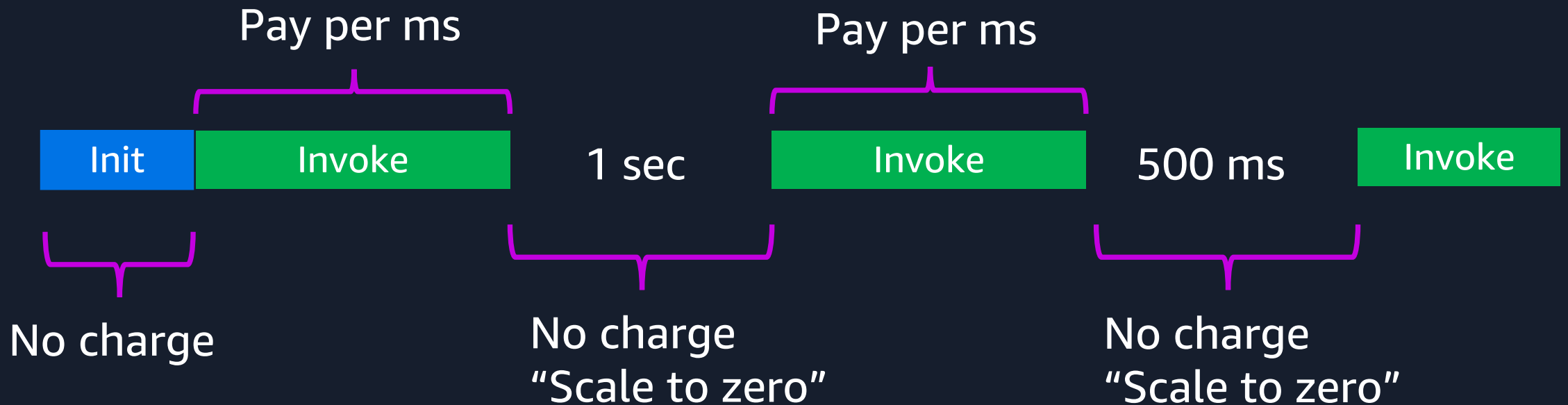
Pay a constant rate per second for the task CPU and memory



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

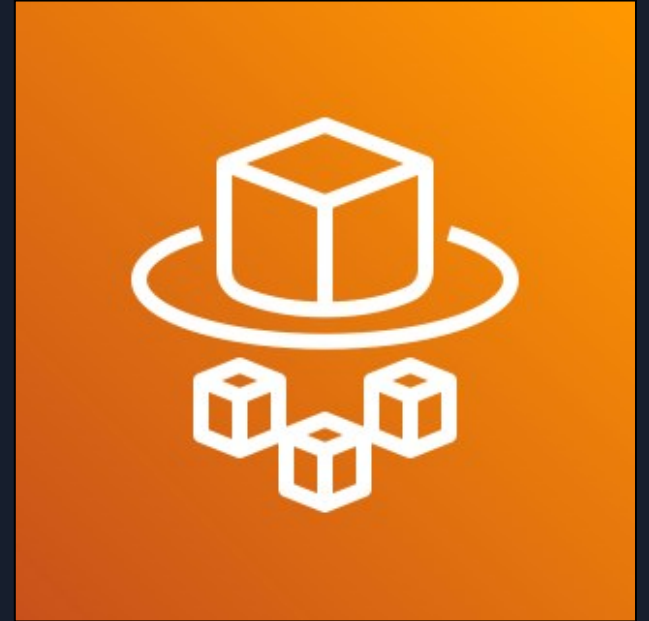
Ideas on selecting . . . serverless containers

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 . . .

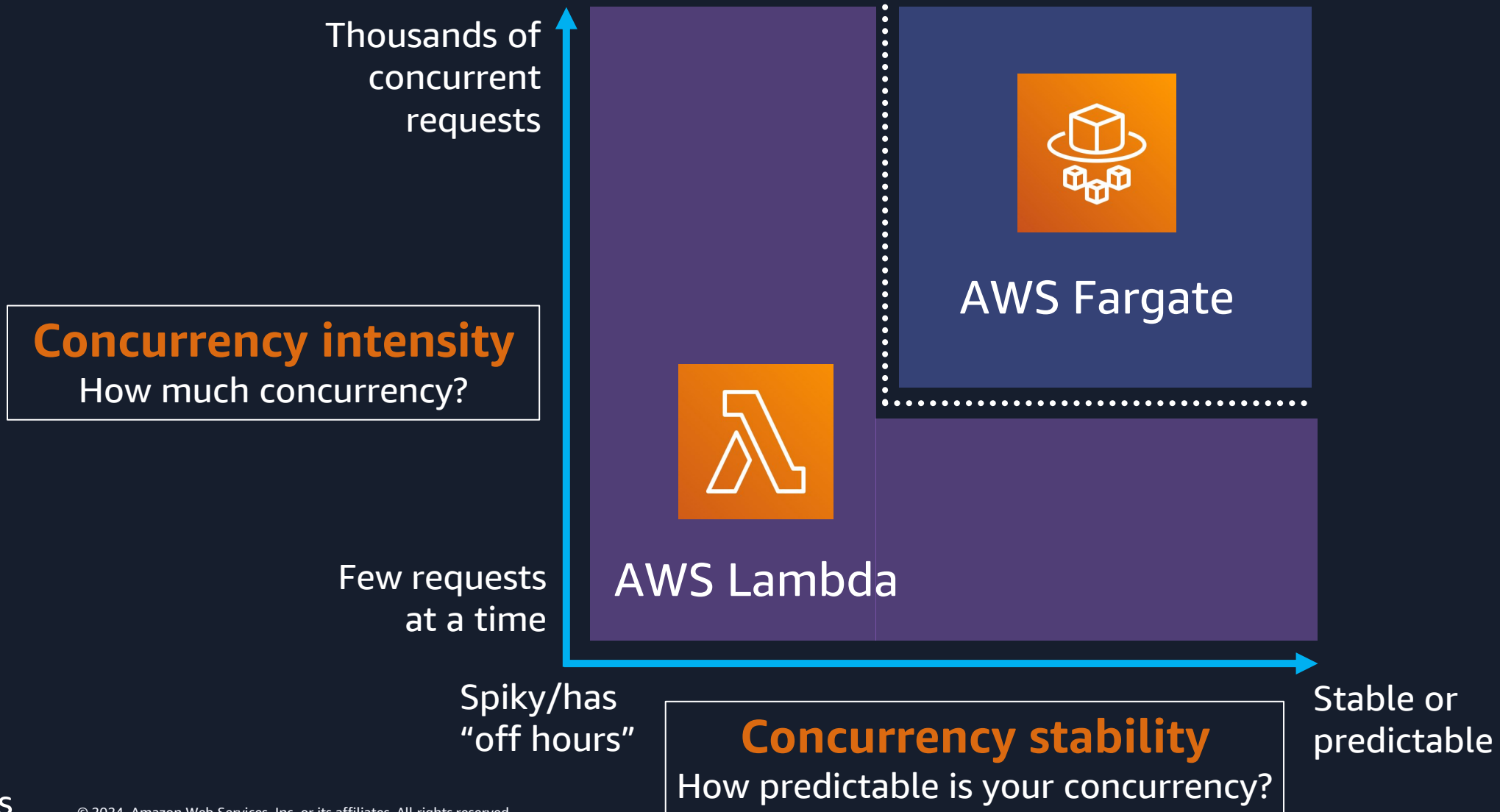
- Immediate startup time
- To trigger action on an event
- Support for varying utilisation
- Ability to handle unknown demand

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

But I can't do serverless, because....

Not for prod

Cost

It's different

Lock-in

Can't scale

Developer experience

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

ServerlessLand.com



Serverless Land

Content ▾

Learn

Code ▾

EDA

Search

Search

Welcome to **Serverless** Land

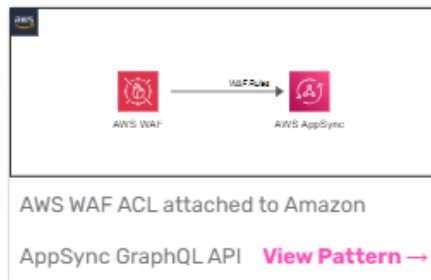
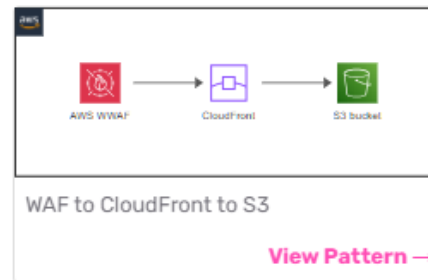
This site brings together the latest information, blogs, videos, code, and learning resources for AWS Serverless. Learn to use and build apps that scale automatically on low-cost, fully-managed serverless architecture.

Serverless **Patterns** Collection

Build integrations using infrastructure as code with **serverless patterns**

669

PATTERNS



Thank you!

Julian Wood

X @julian_wood

in linkedin.com/in/julianrwood

