PUBLIC SECTOR SUMMIT ONLINE

Designing serverless architectures for scale and speed

George Mao

Head of WW Solutions Architecture, Serverless Computing

Amazon Web Services

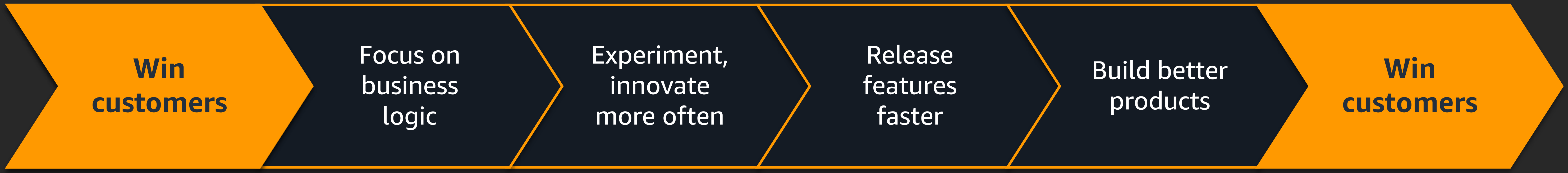
aws PUBLIC SECTOR SUMMIT ONLINE

Agenda

- The evolution of serverless computing
- What’s new with serverless

We are witnessing a paradigm shift

75% *of organizations use or plan to use serverless technologies within the next two years.*[1]



Win customers

Focus on business logic

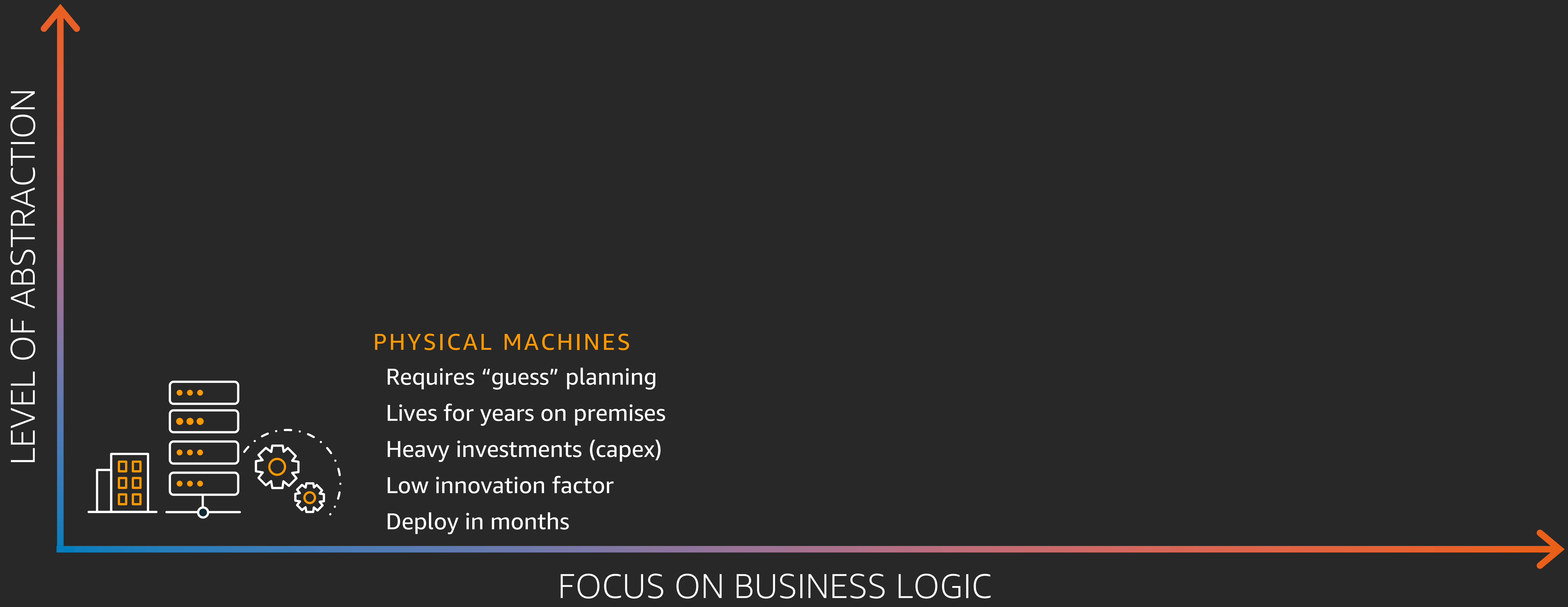
Experiment, innovate more often

Release features faster

Build better products

Win customers

Computing evolution – a paradigm shift



LEVEL OF ABSTRACTION

PHYSICAL MACHINES

Requires “guess” planning

Lives for years on premises

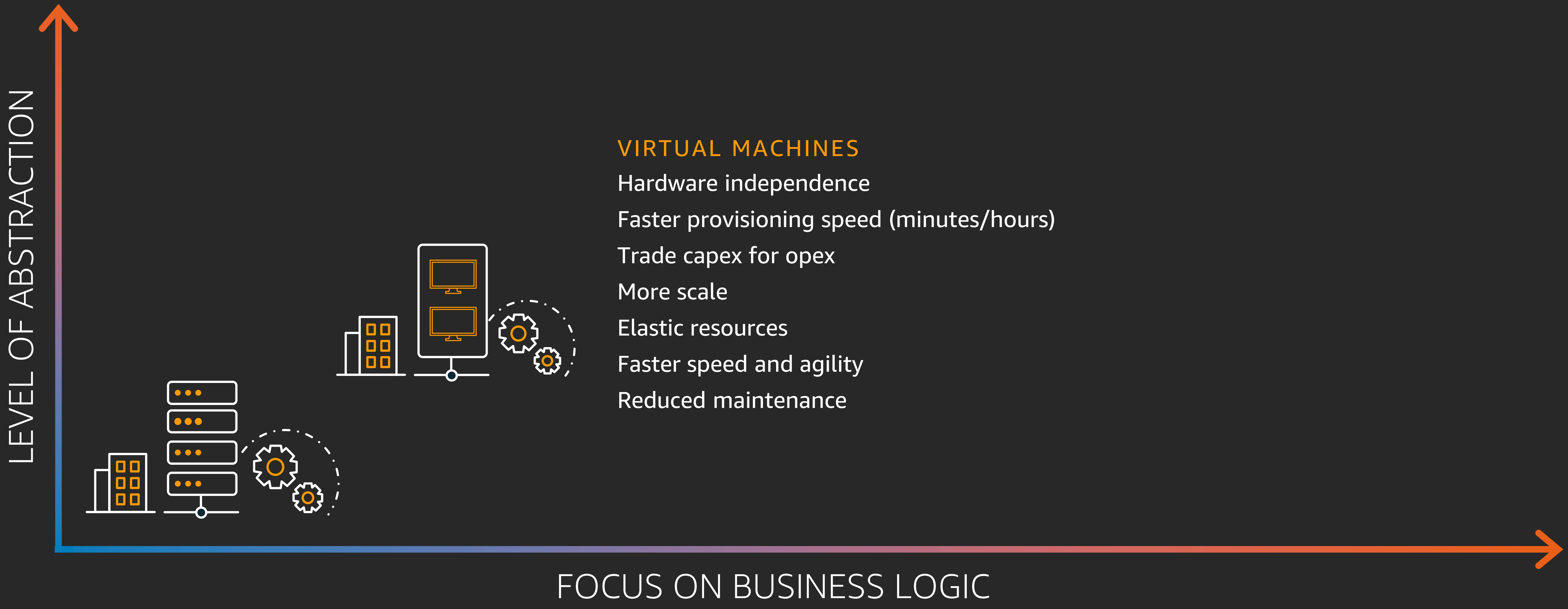
Heavy investments (capex)

Low innovation factor

Deploy in months

FOCUS ON BUSINESS LOGIC

Computing evolution – a paradigm shift



VIRTUAL MACHINES

Hardware independence

Faster provisioning speed (minutes/hours)

Trade capex for opex

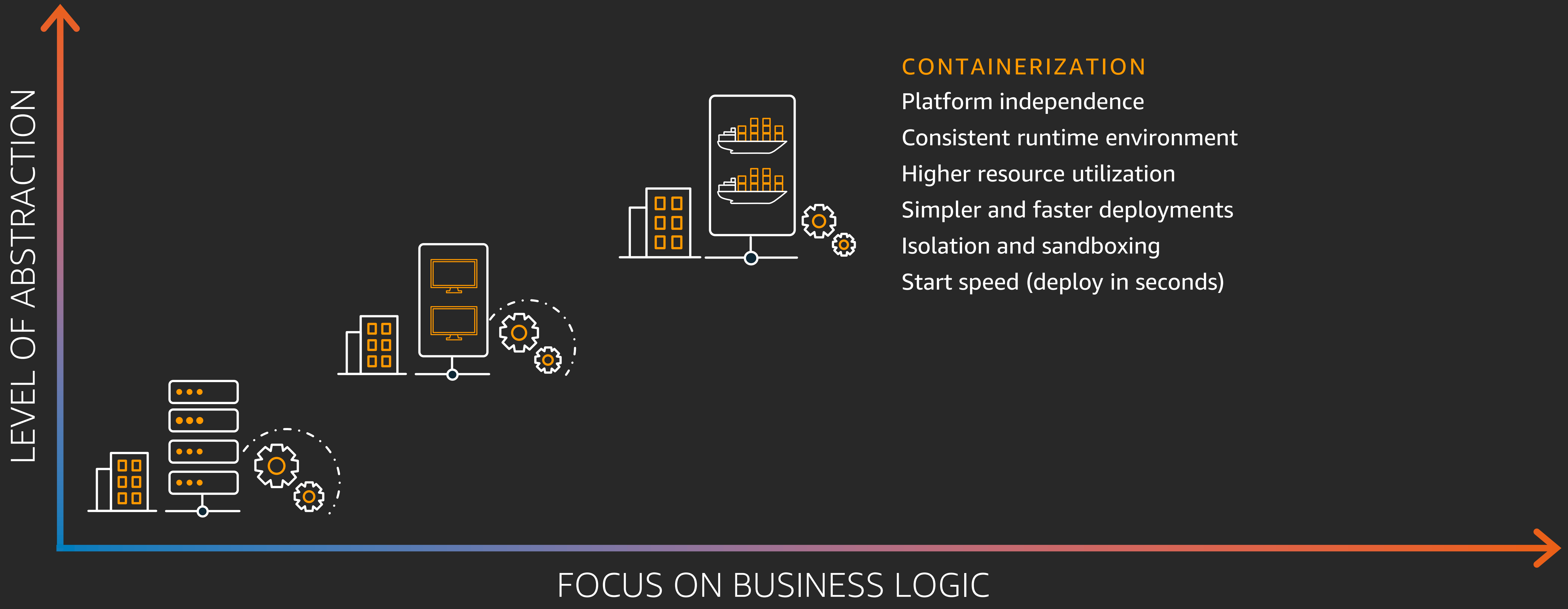
More scale

Elastic resources

Faster speed and agility

Reduced maintenance

Computing evolution – a paradigm shift



CONTAINERIZATION

Platform independence

Consistent runtime environment

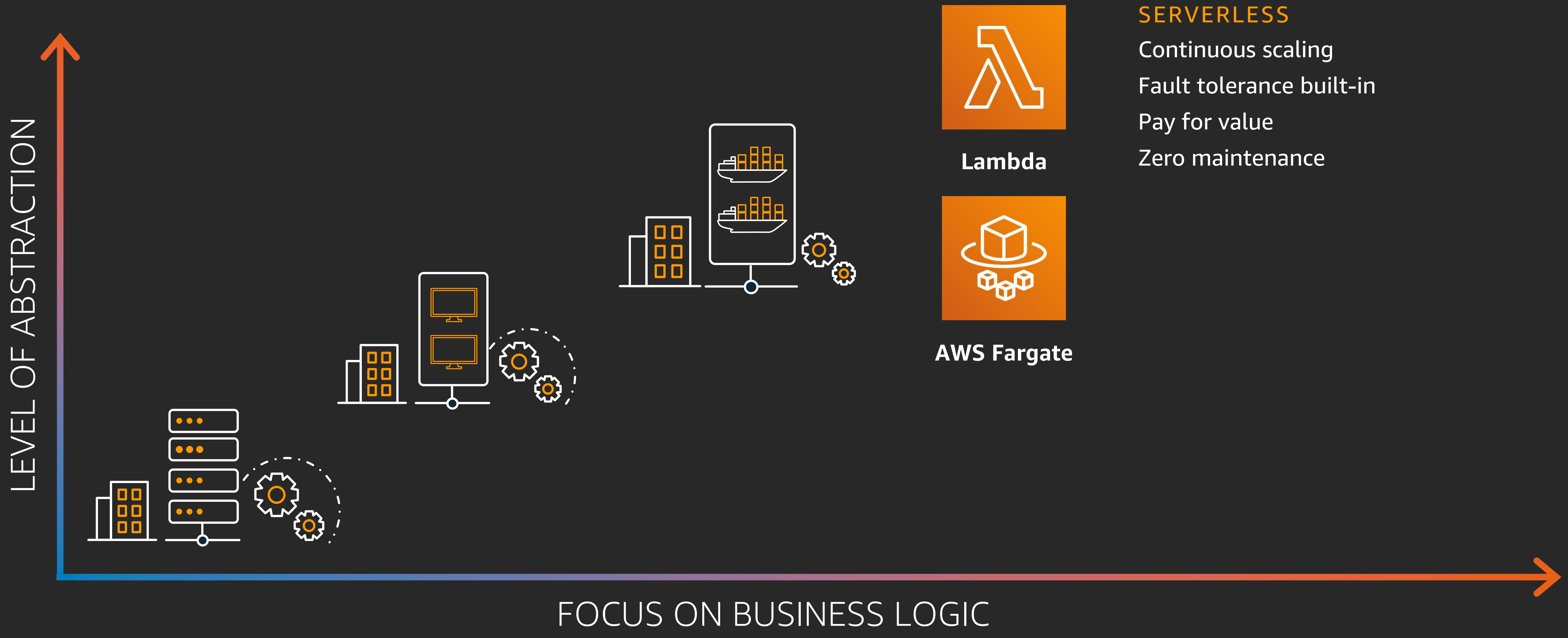
Higher resource utilization

Simpler and faster deployments

Isolation and sandboxing

Start speed (deploy in seconds)

Computing evolution – a paradigm shift



SERVERLESS

Continuous scaling

Fault tolerance built-in

Pay for value

Zero maintenance

Serverless launches

Launches

Lambda enhanced controls

- Async controls
- Enhanced streaming controls

Lambda Provisioned Concurrency

- Managed prewarming

Lambda Destinations

- Send execution results to a destination

API Gateway HTTP APIs

- Faster, cheaper APIs
- Simplified Lambda response format
- Better developer experience

Amazon RDS Proxy

- Managed connection pools

Express Step Functions

- Faster, cheaper Step Functions

Amazon Elastic File System (EFS) for Lambda

- Persistent, durable storage

AWS
Lambda

Async enhanced controls

Max event age

Configurable
60s → 6 hours

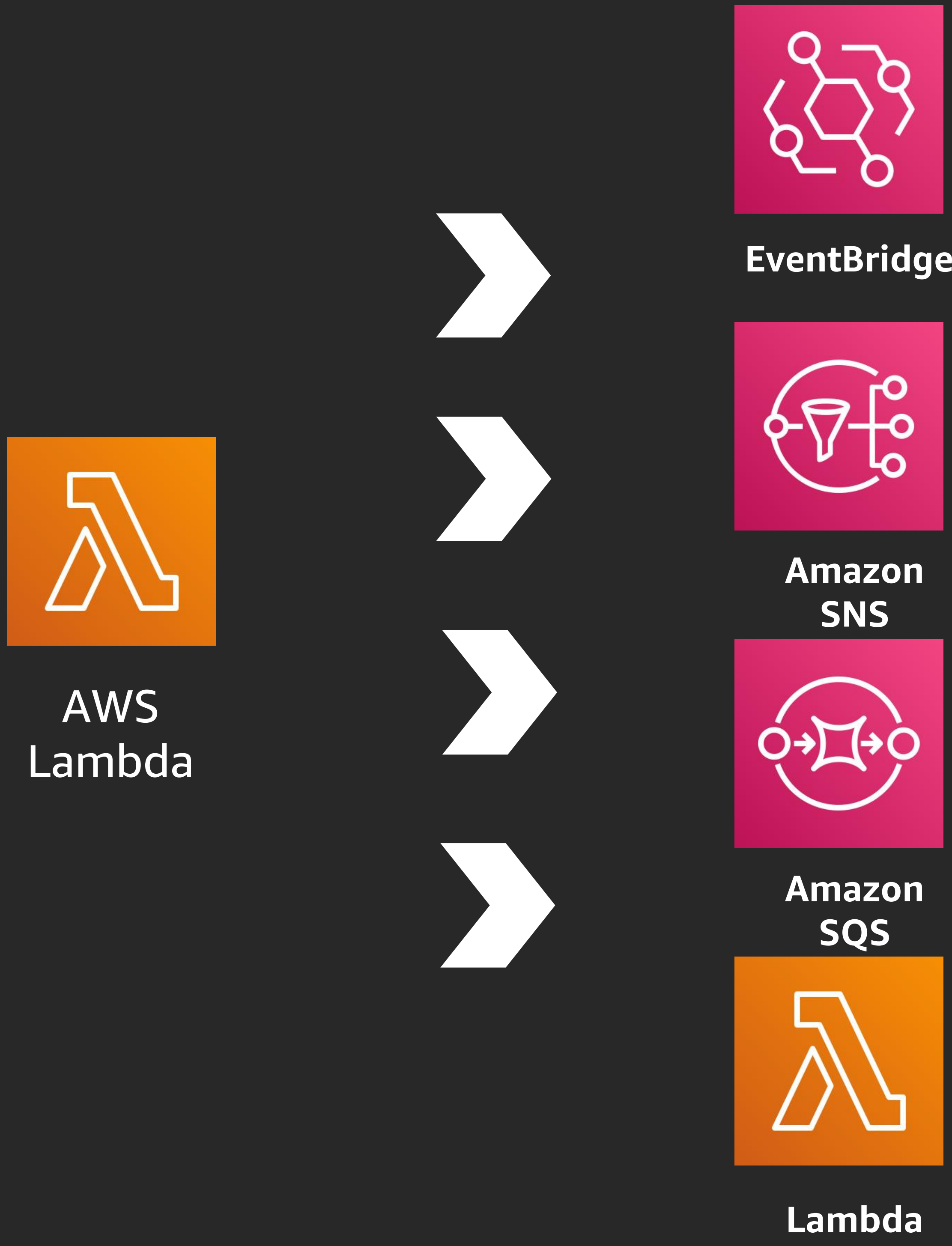
Max retry attempts

Configurable
0 → 2 times

These are optional settings
Defaults still apply

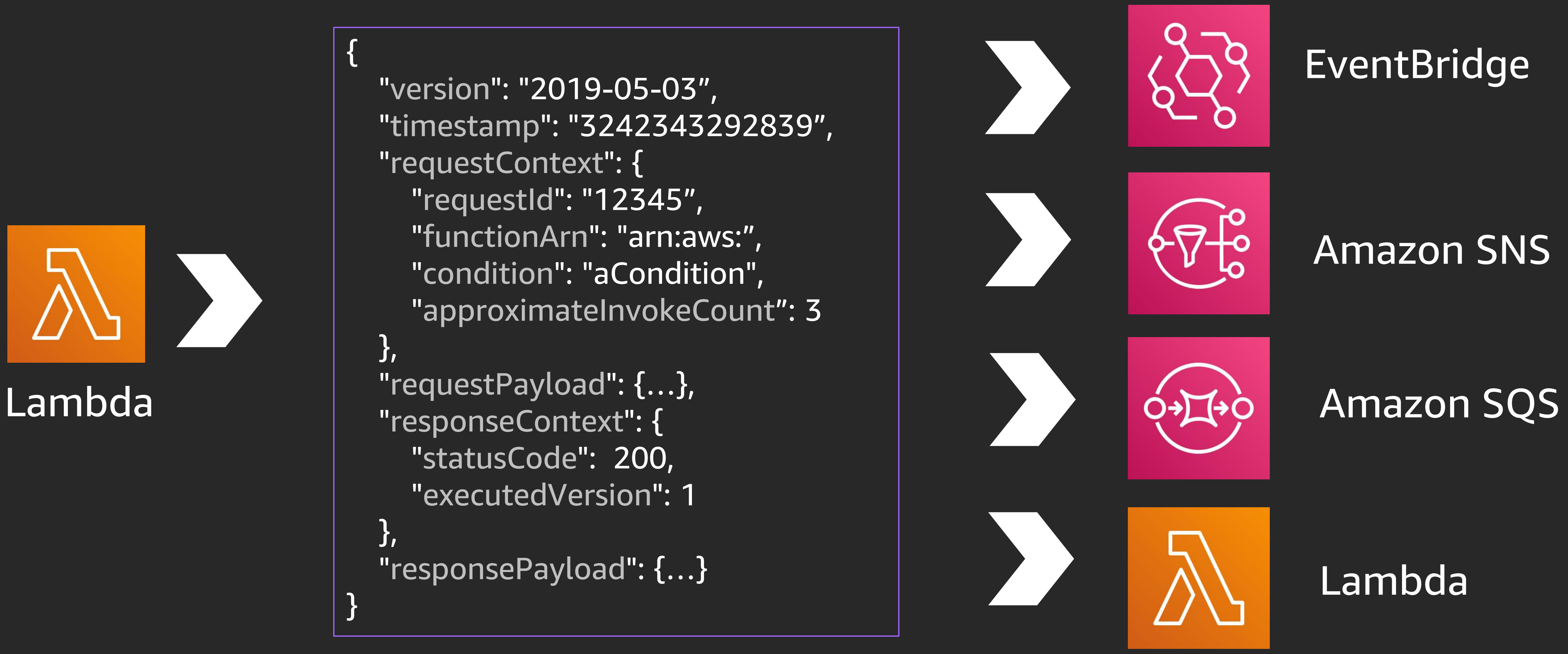
Invoke downstream without writing code

Lambda Destinations



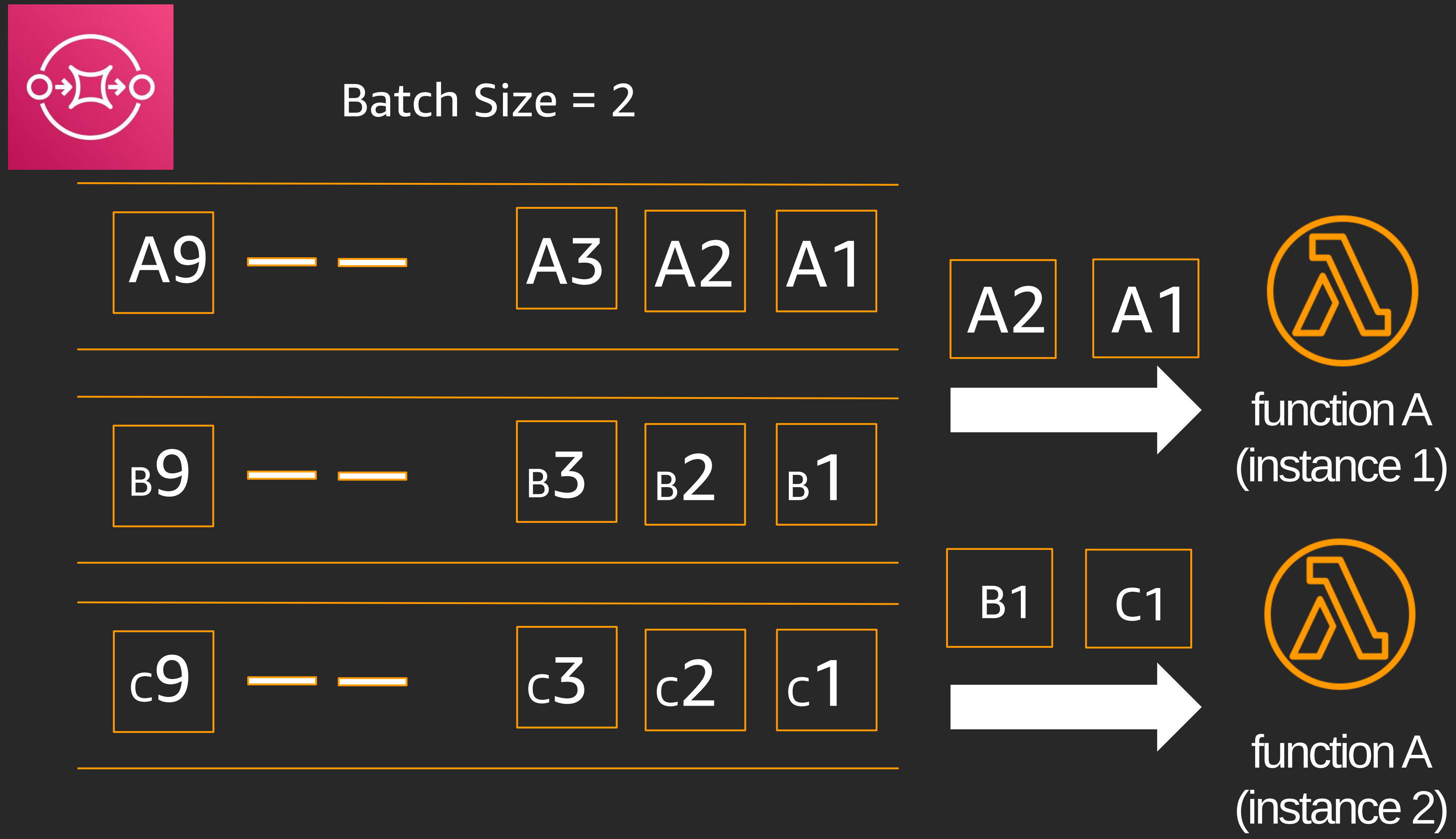
Do more with less: Use Lambda Destinations

Designate an asynchronous target for Lambda function invocation results (success or failure)



```
{
   "version": "2019-05-03",
   "timestamp": "3242343292839",
   "requestContext": {
      "requestId": "12345",
      "functionArn": "arn:aws:",
      "condition": "aCondition",
      "approximateInvokeCount": 3
   },
   "requestPayload": {…},
   "responseContext": {
      "statusCode":  200,
      "executedVersion": 1
   },
   "responsePayload": {…}
}
```

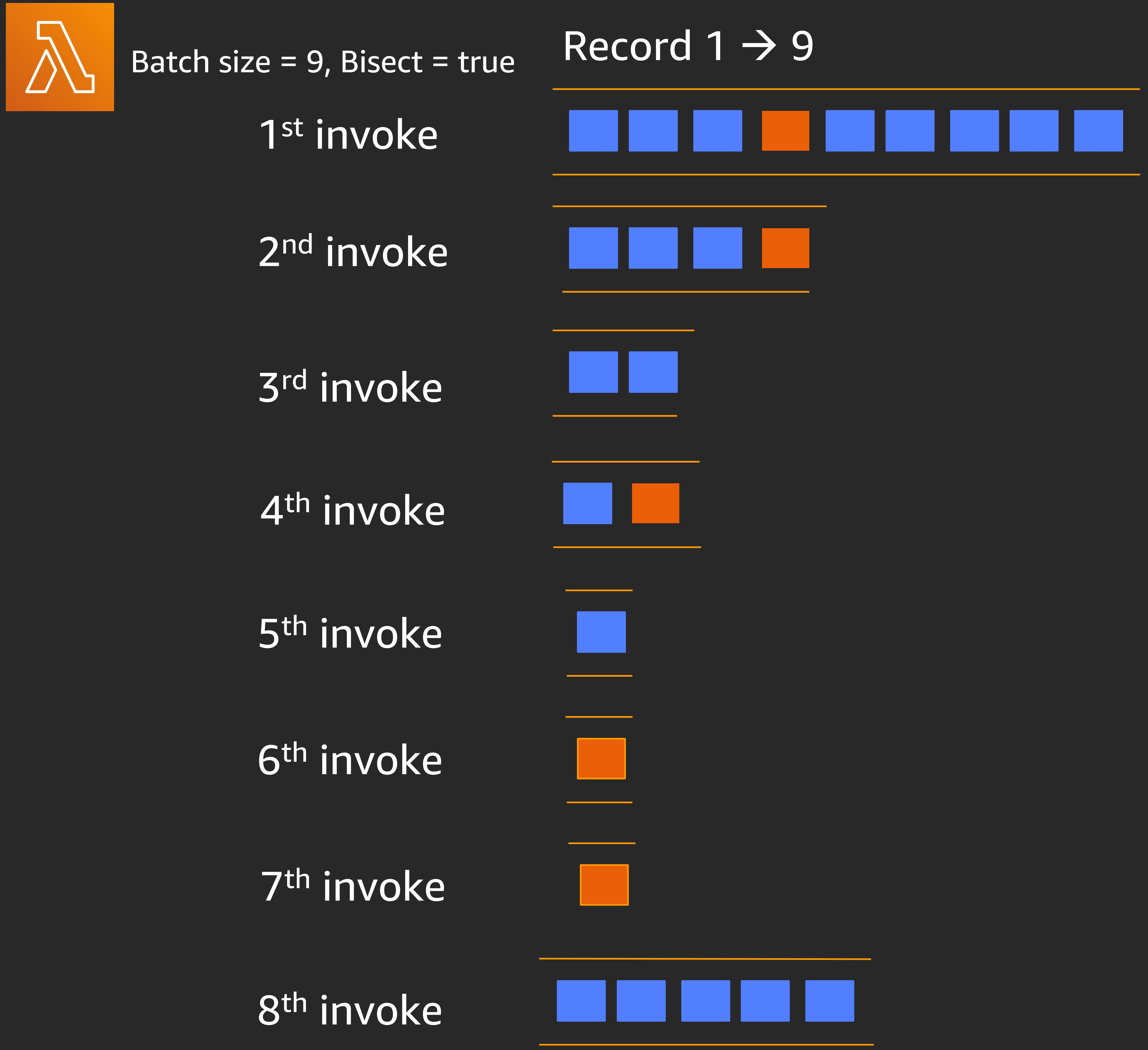
Amazon SQS FIFO queues as an event source



High-throughput, ordered, invoked in batch

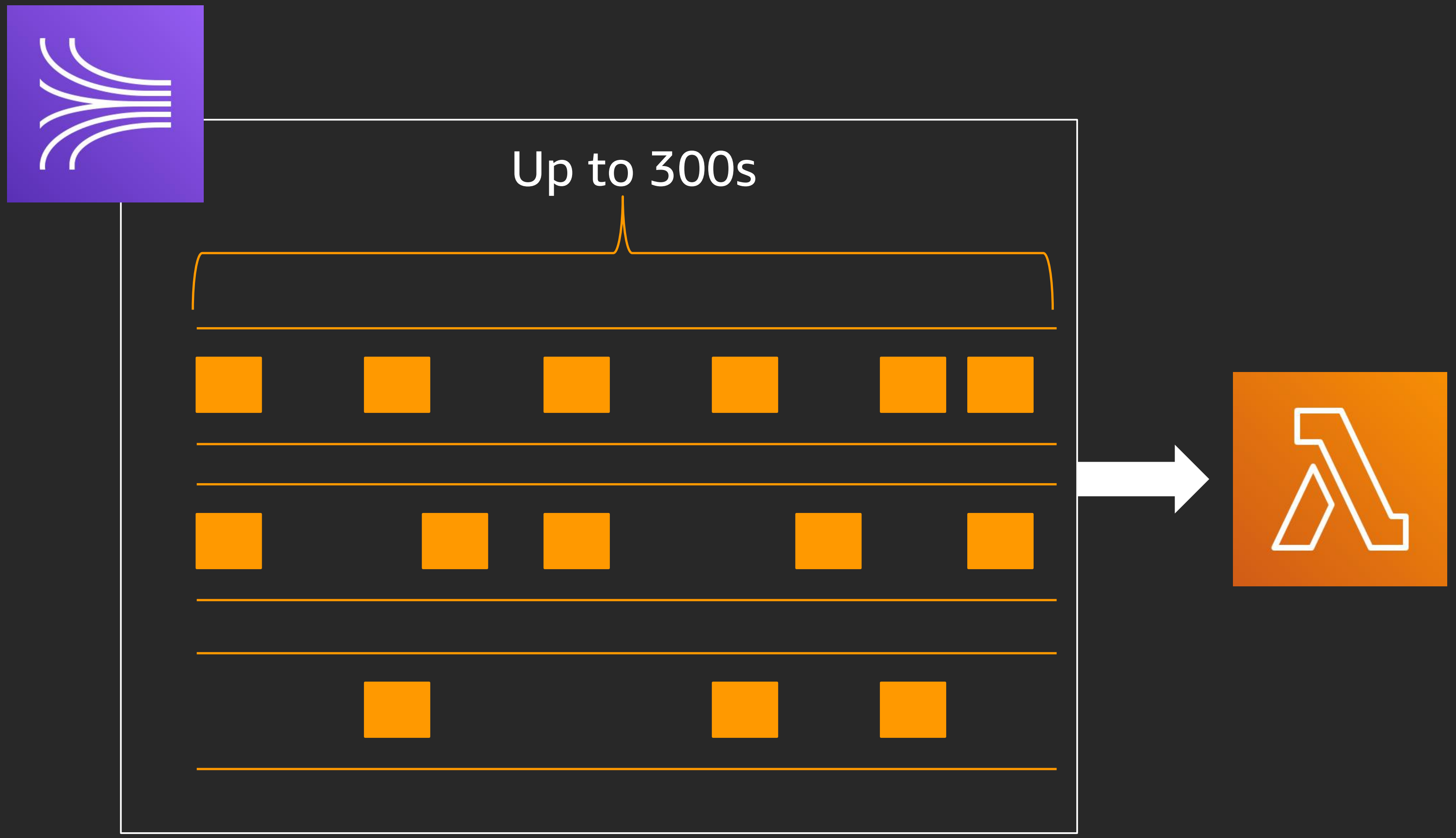
Enhanced controls on stream processing

BisectOnError



Dealing with sparse data streams off peak

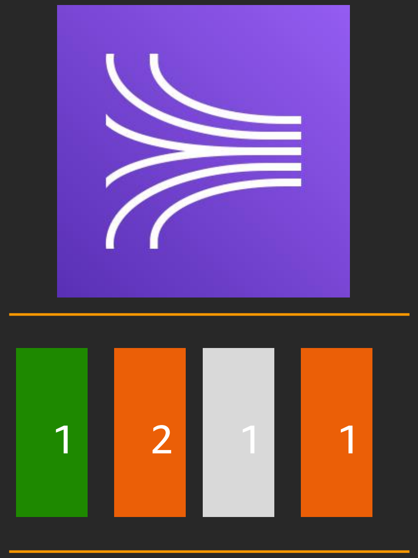
Batch window



Build a batch of records for up to 300 sec/5 min
Optimize for performance and cost

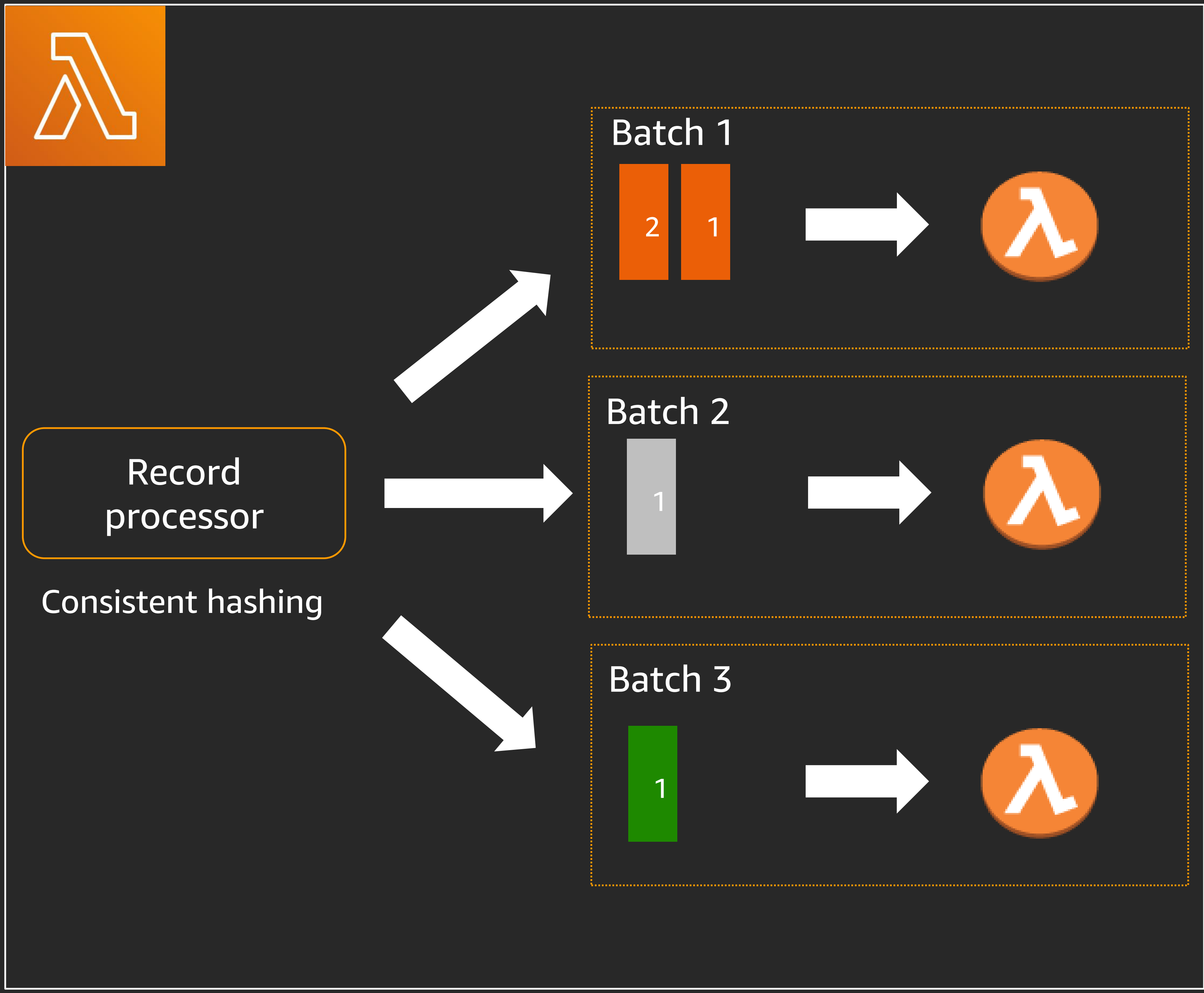
Parallelization factor

Dealing with data streams spikes



Amazon Kinesis

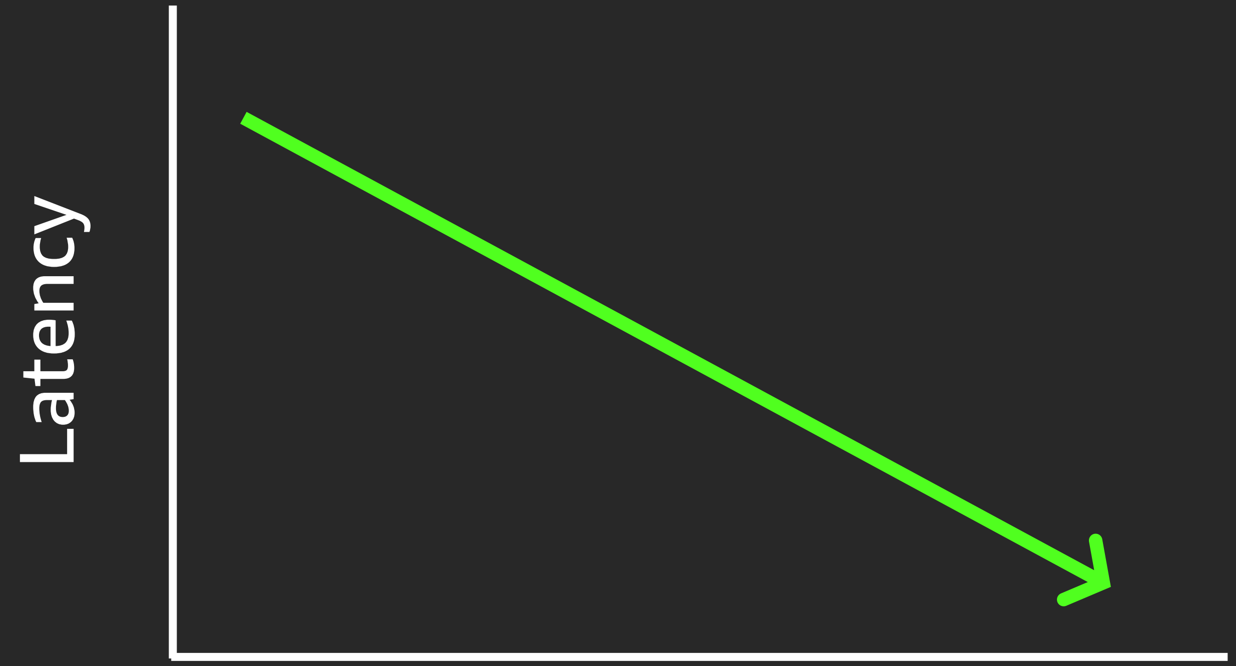
- By default, Lambda invokes one batch on one instance per shard
- Parallelization factor allows you to have up to 10 batches on 10 instances per shard
- Order is maintained at partition key level



Lambda Provisioned Concurrency

NEW

Provisioned Concurrency lets you **pre-initialize functions**



Years of performance tuning

Up to **1000** in **a few** minutes

Then **500** per minute

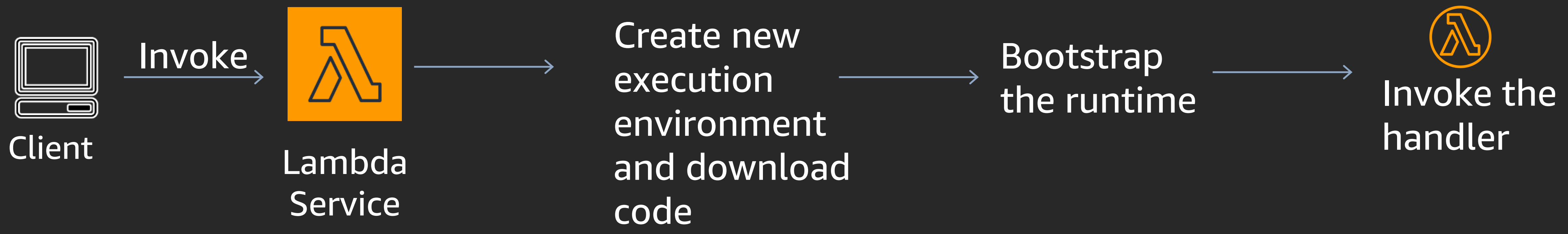
10,000 in **15** minutes

Useful for flash-sale workloads

Lambda Provisioned Concurrency

- Provisioned Concurrency keeps functions initialized and hyper-ready to respond in double-digit milliseconds
- You fully control when or how long to enable Provisioned Concurrency
- Taking advantage of Provisioned Concurrency requires no changes to your code

What happens on a cold start?


Client

Invoke

Lambda Service

Create new execution environment and download code

Bootstrap the runtime

Invoke the handler

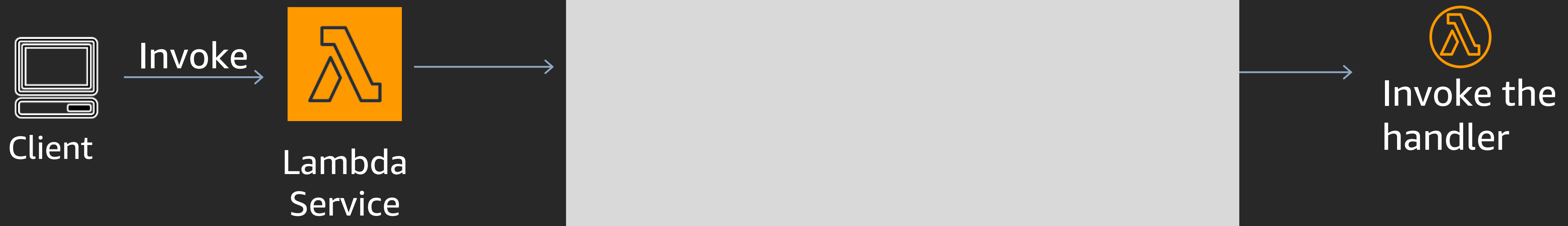
Prior to Provisioned Concurrency…

- Pre-warm your function with concurrent invocations, every 5 mins
- Pass in a test payload
- Create handler logic that doesn’t run the whole function
- Monitor CloudWatch Concurrency metrics

```
exports.handler = async (event) => {
        // if a warming event
        if (await warmer(event))
                return 'warmed’

        // else proceed with handler logic
        return 'Hello from Lambda’
}
```

Now what happens on a cold start?



Empty grey box drawn between the arrows, not included in figure box?

Provisioned Concurrency use cases: High-velocity traffic bursts

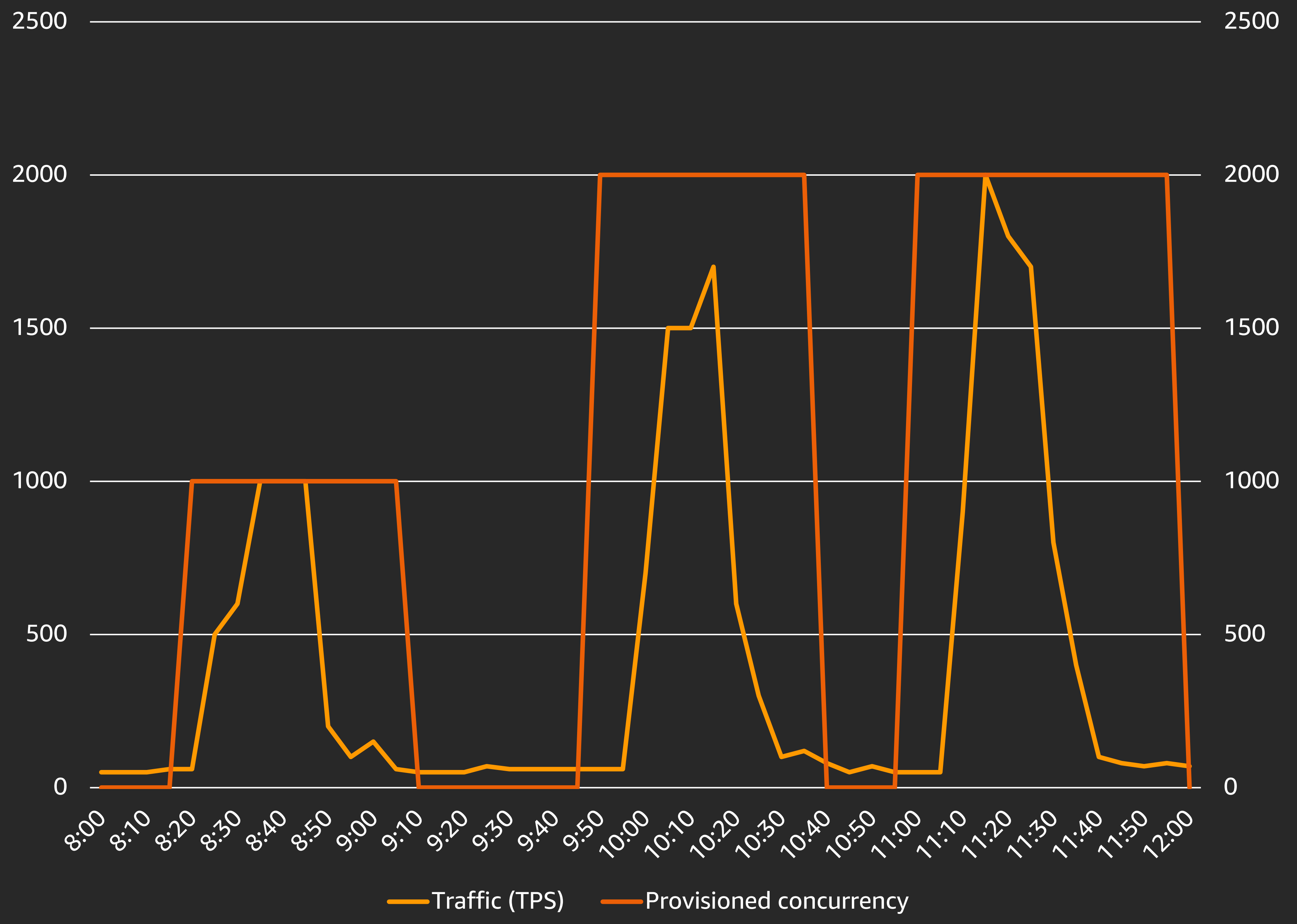
Applications that:

- Serve content such as ads during a live stream
- Mobile applications such as games
- Marketing blitzes or flash sales

Lambda will:

- Provisioning **scheduled** in advance
- Schedule in advance to allow for **scale-up**
- Provision in increments of **5 mins**
- Invocations above the provisioned concurrency are handled by **on-demand** Lambda

Provisioning Concurrency auto scaling



- Provisioning **scheduled** in advance
- Schedule in advance to allow for **scale-up**
- Provision in increments of **5 mins**
- Invocations above the provisioned concurrency are handled by **on-demand** Lambda

Amazon API Gateway: HTTP APIs

https://aws.amazon.com/blogs/compute/announcing-http-apis-for-amazon-api-gateway/

Up to 60% lower latency

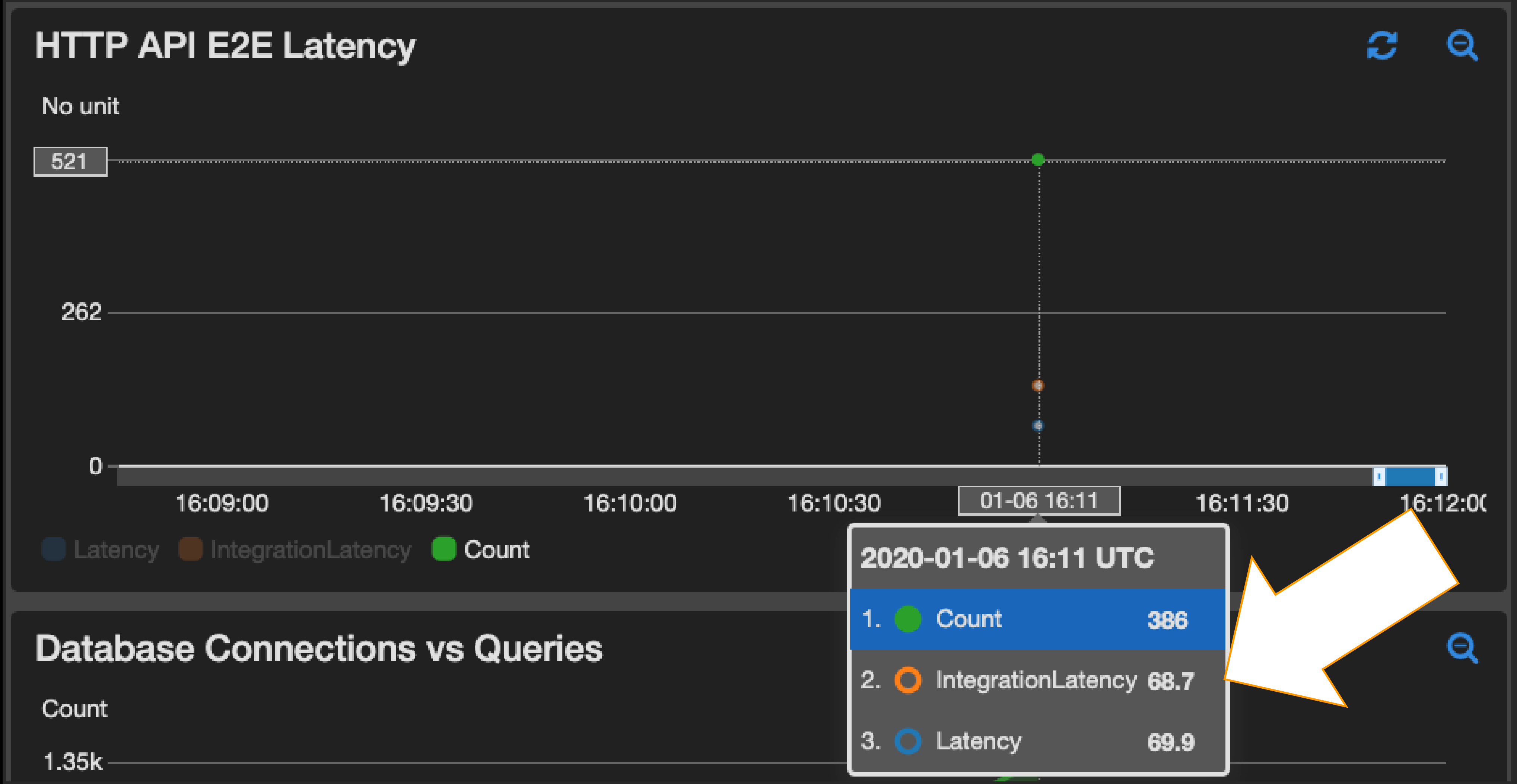
(single ms overhead)

Up to 71% cheaper

($3.50 vs $1)/M

Simpler dev experience

HTTP APIs



2020-01-06 16:11 UTC

1. Count 386
2. IntegrationLatency 68.7
3. Latency 69.9

Private integrations

HTTP APIs now offers developers the ability to integrate any private resource in a VPC

You choose from:

- Application load balancer
- Network load balancer
- AWS Cloud Map

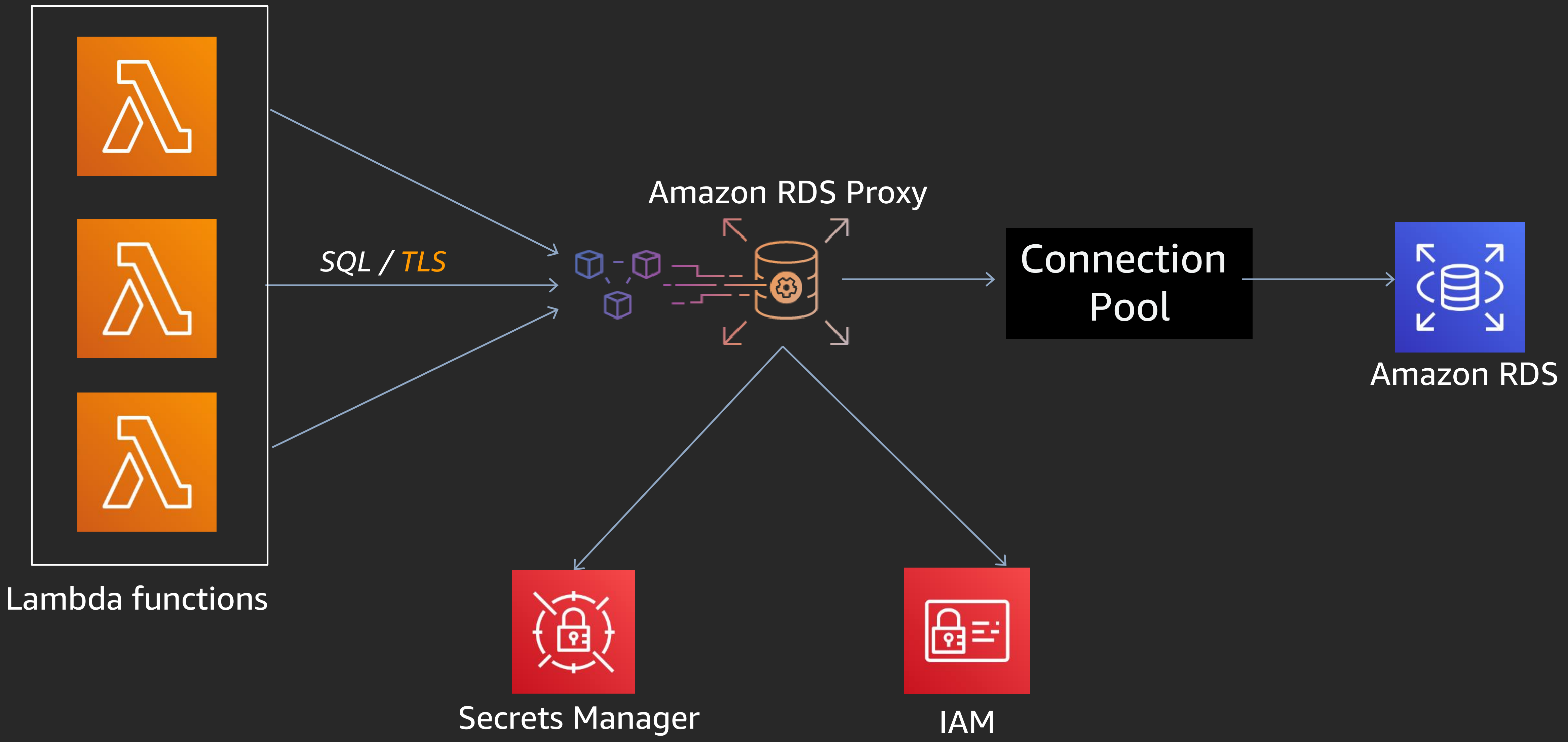
Lambda payload version 2.0

```
{
 ”statusCode": 200,
 ”body":  {
        ”Name": ”George Mao",
        ”Handle": ”@iamgmao"
  },
 ”Headers": {
        ”Access-Control-": ”…",
        ”X-API-Key": ”…",
  }
}
```

NEW!

```
{
        ”Name": ”George Mao",
        ”Handle": ”@iamgmao"
}
```

Amazon RDS Proxy



AWS Step Functions Express Workflows

Faster:
>100K state transitions per second

Designed for short duration workflows:
<5min

Cost effective at scale

Which one is right for your use case?

Standard Workflows

- IT automation
- Report generation
- Order processing
- Payment and billing
- Machine learning model training
- ELT and big data orchestration
- Media processing

Express Workflows

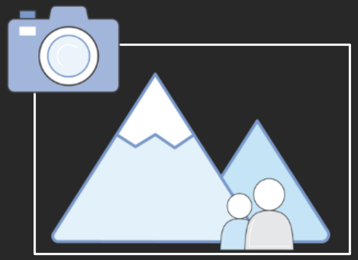
- Event driven microservice orchestration
- High volume data processing
- IoT data ingestion
- Order/cart validation
- payment reconciliation

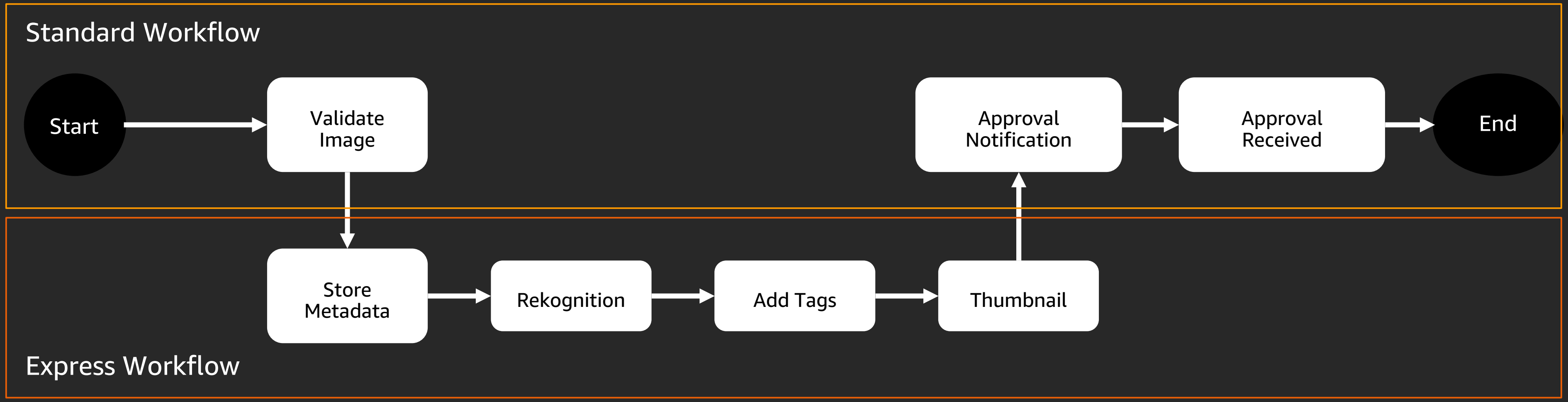
Standard vs express workflows

	Standard	Express
Maximum duration	365 days	5 mins
Start execution refill rate	300 per second	6,000 per second
State transition refill rate	1,300 per second	None
Execution semantics	Exactly-once workflows step execution	At-least-once workflow step execution

Standard + Express Workflows

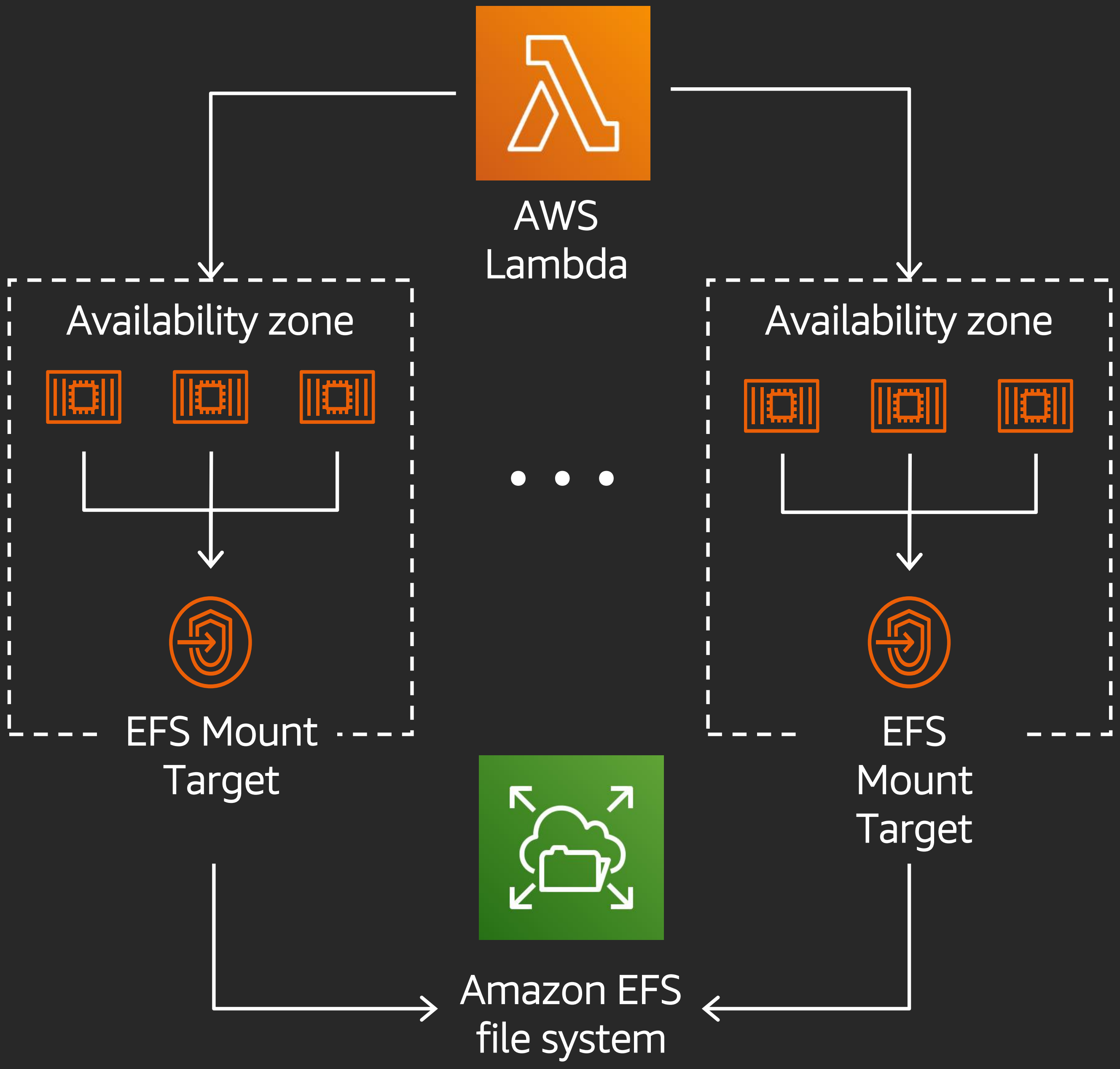
Express Workflows can be nested within a standard workflow enabling you to use Express Workflows for short periods of task execution and a Standard Workflow for long periods of task execution or waiting





Introducing Amazon EFS for Lambda

- Share data across 1,000s of function invocations
- Achieve high performance, highly available, durable storage with persistent volumes
- Pay only for what you use



New workloads on AWS Lambda

Simplify application architecture
Process files of any size
Reduce costs

Large file
data manipulation

Zip/Archives
Git

Large scale
media processing

High res images
HD videos

AI/ML
analytics

MXNet
TensorFlow

Realtime
applications

Content management
web apps

How do you get started?

What are your first steps?

Find your serverless developers today

Build a tiger team

Consider a lighthouse project

Abt Associates and the Department of Housing and Urban Development (HUD)

HUD, working with Abt Associates, developed and launched a tool called the Homelessness Data Exchange 2.0 (HDX), a web application that allows Continuums of Care (CoCs) to submit data on the cloud and access Stella—an interactive visualization tool analyzing their current homeless assistance system's performance and showing how homeless households flow through their service pathways

This serverless system, powered by Lambda paired with Java frameworks, allows the application's code to be fully distributed, resulting in a great increase in performance at a fraction of the cost

Comic Relief

The British charity, needed a simple, cost-conscious solution for managing donations during their public campaigns

They switched from multi-cloud on AWS and a third party, to serverless service, AWS Lambda, enabling them to take 350 donations per second during peak moments, and achieve a 93% cost reduction

Learn serverless with AWS Training and Certification

Resources created by the experts at AWS to help you learn about serverless applications

No-cost, on-demand courses on serverless, including *AWS Lambda Foundations* and *Deep Dive on AWS Fargate*

Learn to "think serverless" with new, intermediate-level, three-hour course: *Architecting Serverless Solutions*

Visit the learning library **at https://aws.training**

Conclusion

- We are in the midst of a paradigm shift in computing
- AWS provides the broadest and deepest platform for serverless
- The AWS pace of innovation provides new features help you build faster, cheaper, and better applications

Thank you!

George Mao

aws PUBLIC SECTOR SUMMIT ONLINE