AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SVS401-R1

Best practices for advanced serverless developers

Julian Wood (he/him)

Senior Developer Advocate, Serverless
AWS

About me

Julian Wood

Senior Developer Advocate – AWS Serverless

Recovering server“more” infrastructure engineer

- Enterprises and startups

You can’t scare me, I have twin girls!

From Cape Town via London

SVS401

Best practices for all ~~advanced~~ serverless developers

Julian Wood he/him

Senior Developer Advocate, Serverless
AWS

What are we talking about today?

Serverless is?

Event state

Service-full serverless

Fabulous functions

Configuration as code

From prototype to production

Resources

s12d.com/svs401-22

Serverless = ?

The start of "serverless"?

November 13, 2014 (8 years ago)

AWS introduces the preview of AWS Lambda, an event-driven computing service for dynamic applications

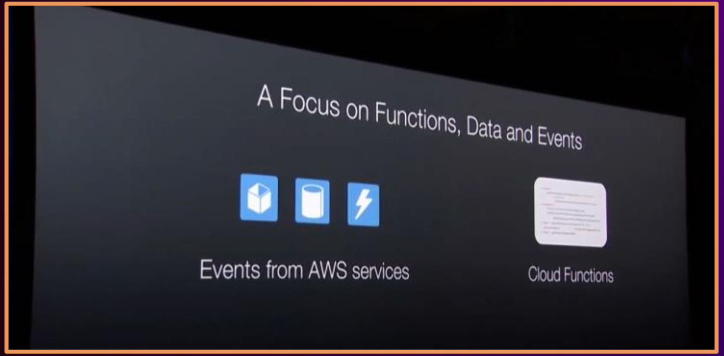
April 9, 2015

AWS Lambda becomes generally available for production use

No mention of "serverless"!

Introducing AWS Lambda

An event-driven computing service for dynamic applications



A Focus on Functions, Data and Events

Events from AWS services

Cloud Functions

Easy to Use, and Low Maintenance

Very Rapid Responses to Events

The start of “serverless”?

March 14, 2006 (17 years ago)

Amazon S3 launched as the first generally available AWS service

November 3, 2004 (18 years ago)

Amazon Simple Queue Service (Amazon SQS) beta

Production on **July 13, 2006 (17 years ago)**

August 25, 2006

Amazon Elastic Compute Cloud (Amazon EC2) beta

Production on **October 23, 2008**

What is “serverless”?

It’s not about having servers or the lack of servers

What is “serverless”?

- Consider “serverless” as:
  - A **mindset**, an approach, a practice, a culture, a way of working
- Focus on business value, rather than the enabling technology

- Benefits:
  - Faster time to market from prototype to production
  - Rapid, continuous experimentation and feedback
  - React and deliver business changes with a product mindset
- **Serverless = the best way to build and run modern applications**

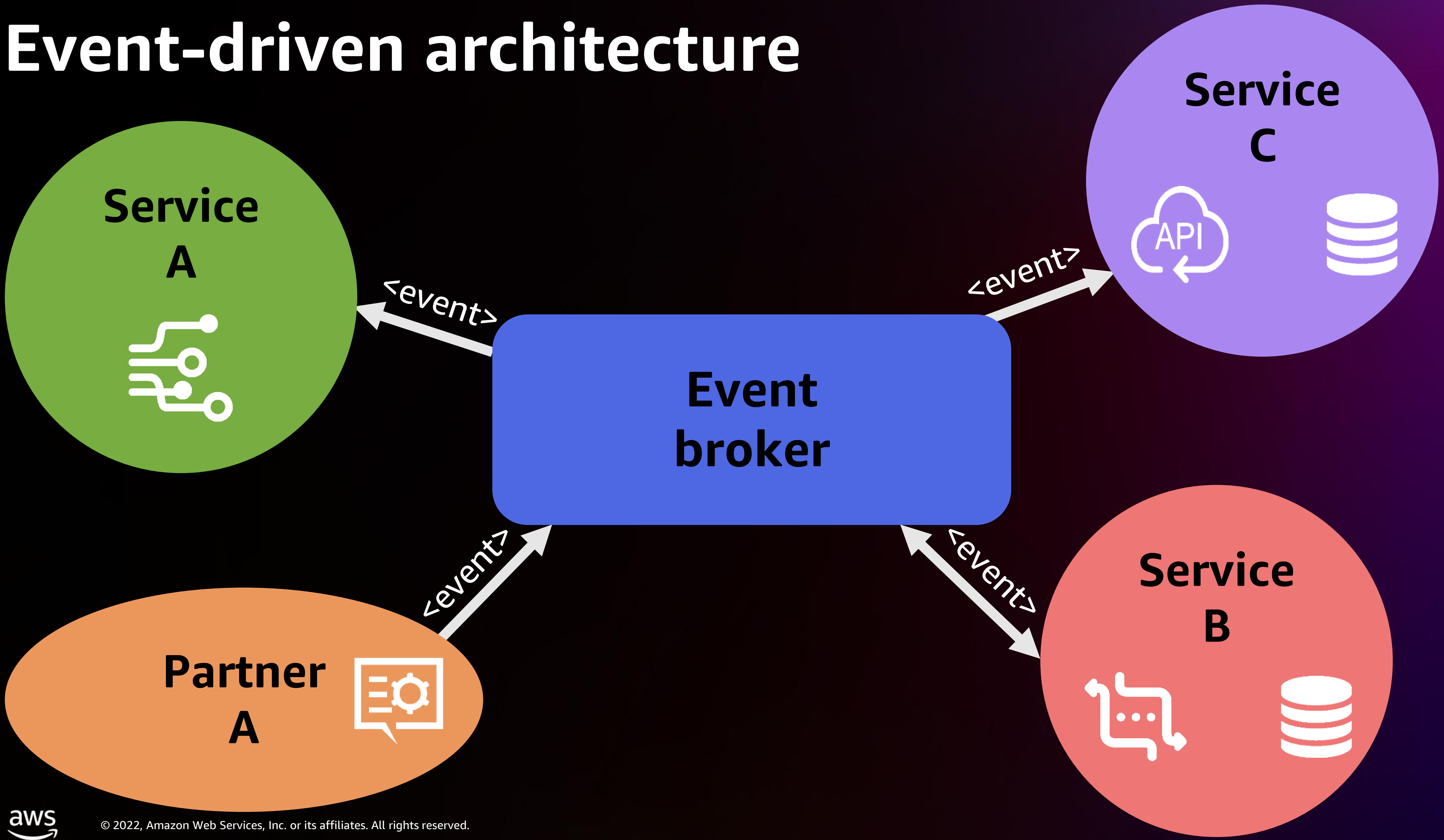
What is “serverless”?

- “Serverless” architecture is an **operational model**
  - Minimize taking on ongoing operational tasks, outsource system administration tasks
  - Every non-serverless component = ongoing ops responsibility
  - Optimize for long-term maintainability
  - Serverless does not mean "No Ops“!

- AWS as your platform (engineering) team
- Build **“in”** the cloud, not just “on” it

- **Concentrate on the flow of data and events**

Event-driven architecture



Serverless/EDA trade-offs

- Different way of designing applications
- More moving pieces
- Harder to understand dependencies
- Eventual consistency
- Different testing/monitoring/observability
- Relying on platform capabilities and security

Liberty Mutual: Serverless-first strategy

Liberty Mutual made a strategic business decision to pursue a serverless-first approach—designed to give it an edge in a competitive, global, and increasingly digital market

The company first built foundational elements of its serverless infrastructure on AWS, such as network security and deployment pipelines, and modernized application development. Many serverless-first projects have been completed, all of which use **AWS Lamb.da**.

Liberty Mutual has used serverless architecture on AWS to build several systems in just 3 months, compared to 1 year on premises. It also reduced computing costs per million transactions to just $60.

2018
2019
2020

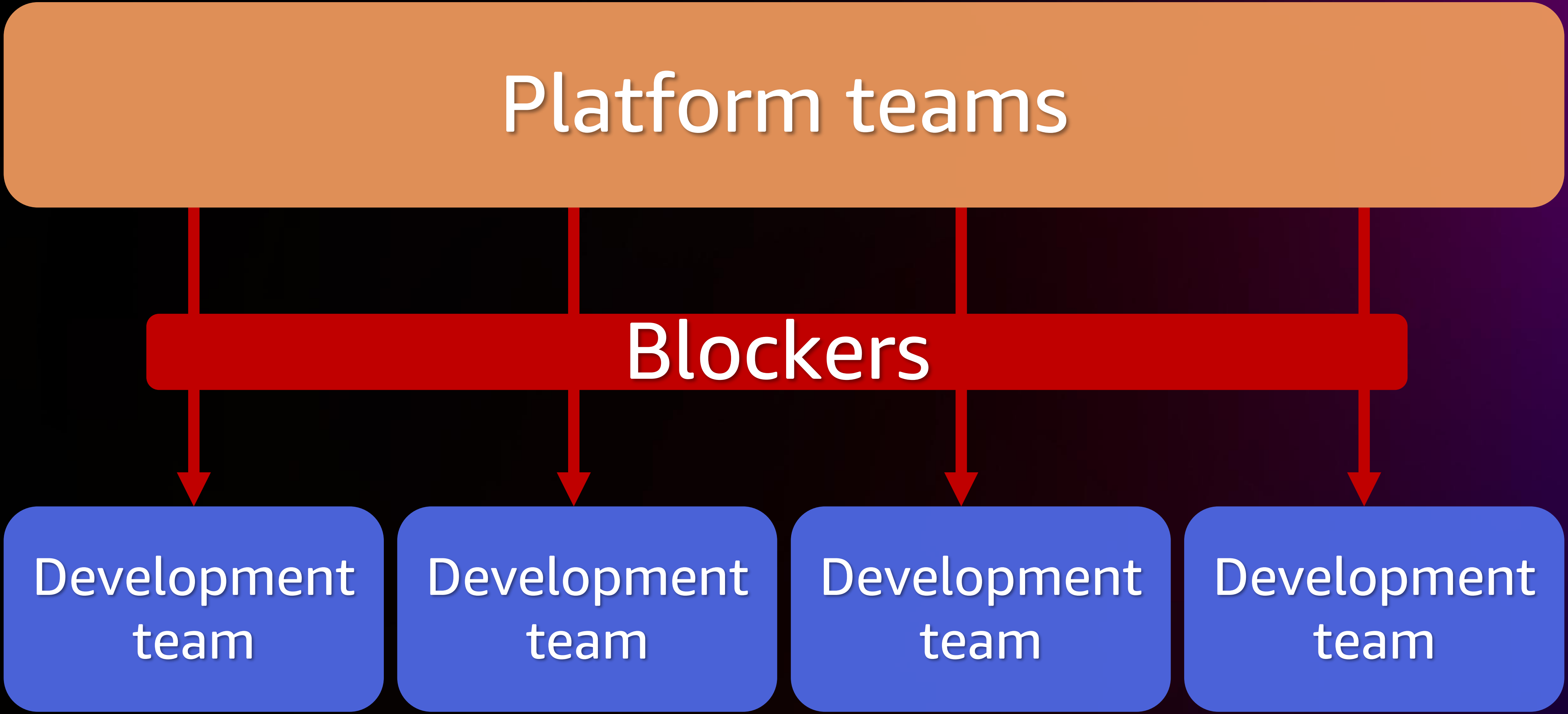
Liberty Mutual sets up a Cloud Center of Excellence (CCoE) for cloud-native workloads with AWS partnership, and reduces costs, improves agility by adopting the serverless-first approach

Enabling constraints

- Curated experience for builders = platform

- Allow for rapid development
- Fast feedback cycles
- Early course correction
- Paved paths
- Codified best practices with standard components
- Speed up the decision-making process

Team enablement


Platform teams

Blockers

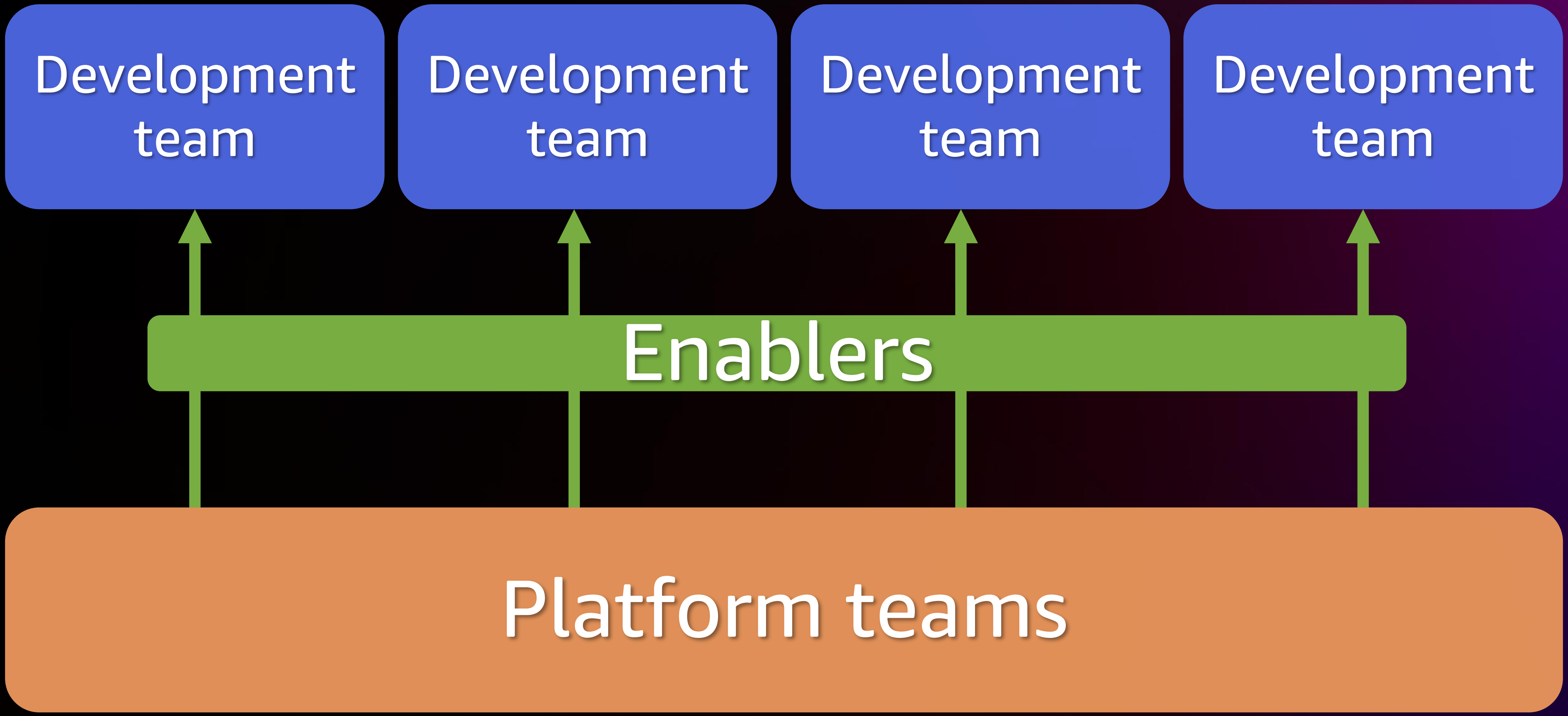
Development team

Development team

Development team

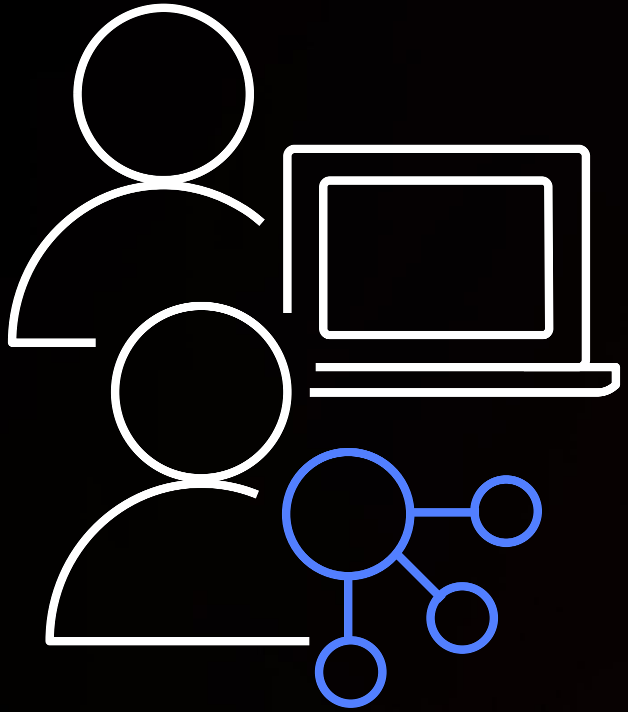
Development team

Team enablement



Platform team enablement

GIVE BUILDERS ENABLING ACCELERATORS WITHOUT BOTTLENECKS



- CI/CD pipelines
- Security guardrails
  - AWS Control Tower
  - Service control policies
  - Permissions boundaries
- Private networking
  - VPC configuration
  - Shared connectivity
- Reusable IaC patterns
  - CDK constructs
  - Serverless patterns

AWS Well-Architected Framework Serverless Applications Lens

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization

Building well-architected serverless applications

s12d.com/well-architected-serverless

Lock-in?

- What if I need to change my mind?
- Two levers to reduce “lock-in”:
  - Reducing switching cost
  - Reducing the likelihood of having to switch

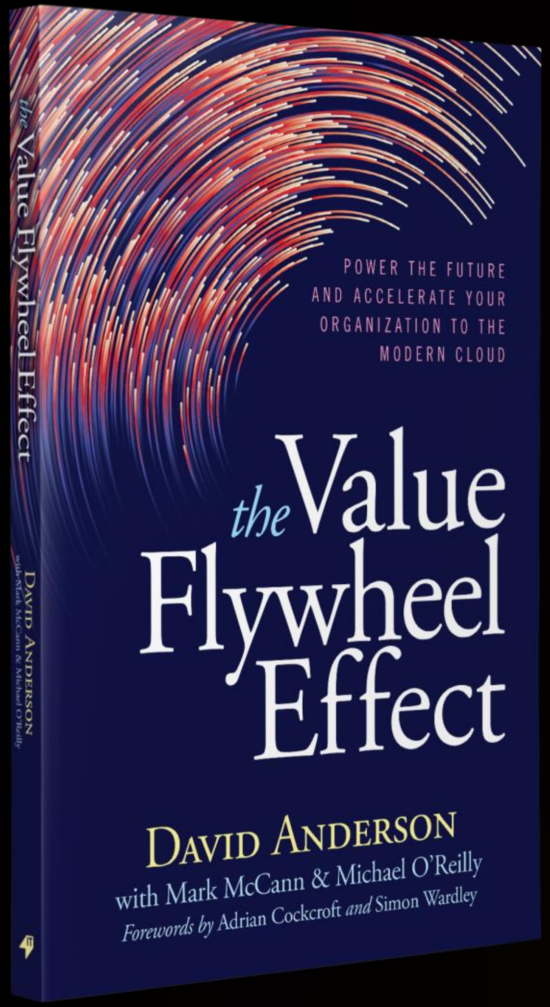
- Switching cost = your velocity: speed with which you can make changes
- Higher your velocity = lower switching cost

- Serverless + managed services + automation = high velocity
- ARC207-Modern cloud applications: Do they lock you in? - Gregor Hohpe

s12d.com/arc207-22

The Value Flywheel Effect Book

POWER THE FUTURE AND ACCELERATE YOUR ORGANIZATION TO THE MODERN CLOUD


POWER THE FUTURE
AND ACCELERATE YOUR
ORGANIZATION TO THE
MODERN CLOUD
the Value Flywheel Effect
DAVID ANDERSON
with Mark McCann & Michael O'Reilly
Forewords by Adrian Cockcroft and Simon Wardley

David Anderson

Mark McCann

Michael O'Reilly

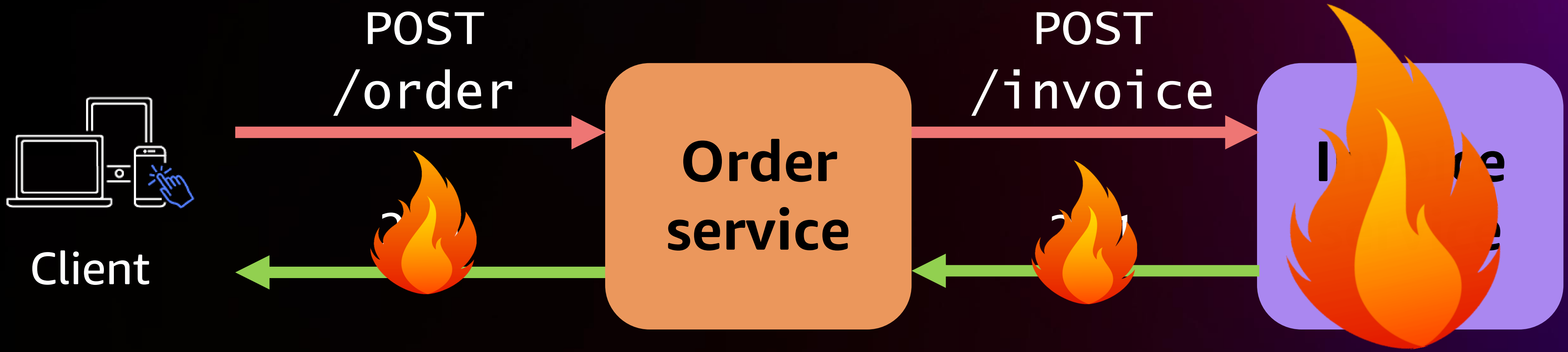
https://itrevolution.com/the-value-flywheel-effect/

Serverless is?: Best practices

- Consider “serverless” as a mindset
- Focus on business value, rather than the enabling technology
- Optimize for long-term maintainability
- Build “in” the cloud, not just “on” it
- Concentrate on the flow of data and events = EDA
- Create enabling platform teams
- Think serverless first

Event state

Synchronous APIs



Client

POST /order

Order service

POST /invoice

Async friends: Queues, topics, buses, streams

Amazon SQS

Queues
Fully managed

Message queuing service to decouple and scale distributed systems

Amazon SNS

Pub/sub topics
Fully managed

High-throughput, push-based, many-to-many messaging between distributed systems

Amazon EventBridge

Event bus
Fully managed

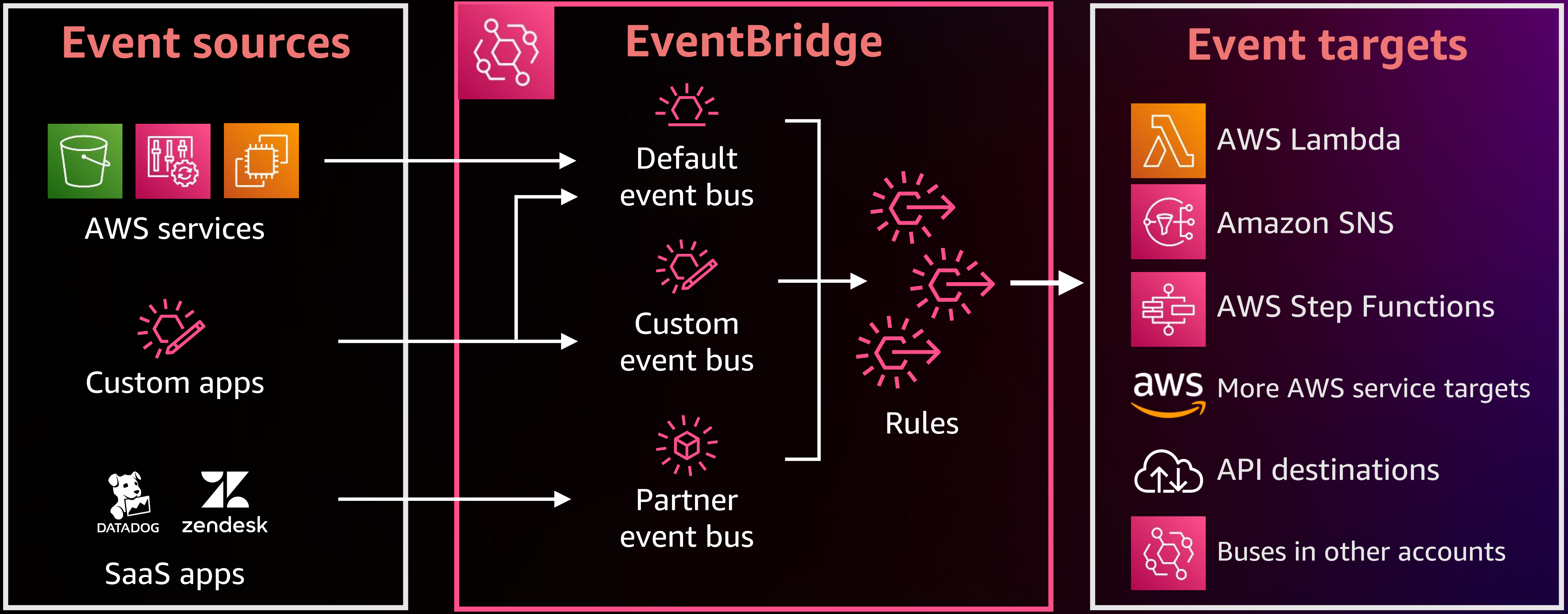
Build event-driven applications at scale for AWS services, your own applications, and SaaS providers

Amazon Kinesis

Streams
Fully managed

Collect, process, and analyze real-time, streaming data

Amazon EventBridge sources and targets

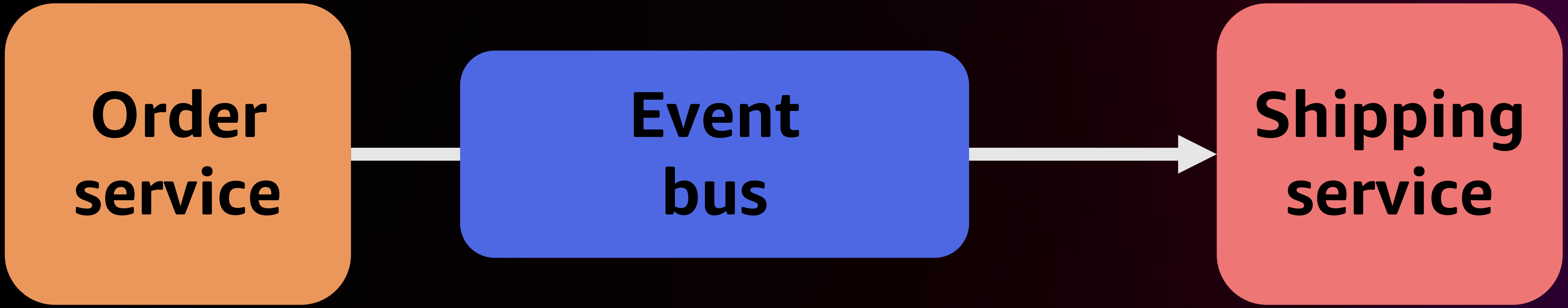


> “Events are the language of serverless applications.”

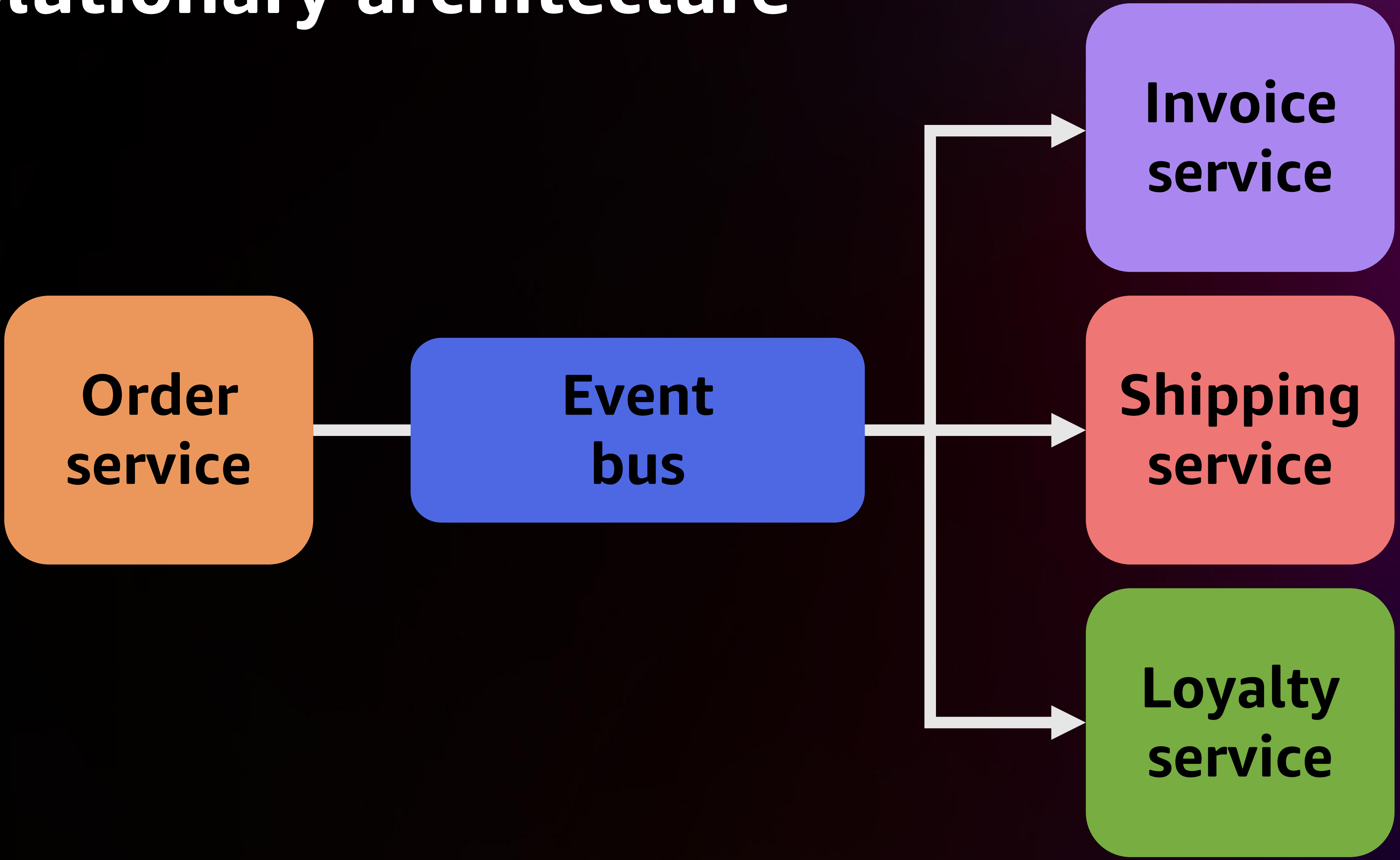
Dave Boyne

AWS Serverless Developer Advocate

Decoupled architectures



Evolutionary architecture



“What information should we put into our events?”

Everyone building event-driven applications

Event content

EVENT ENVELOPE AND DETAIL FIELD

```
{
  "version": “1",
  "id": "10ec61ab-d758-61f7-96a0-592d003f6b0b",
  "source": "MyCompany.MyServerlessApp",
  "detail-type": "Product.ProductInfoUpdated",
  "account": "111122223333",
  "time": "2022-09-15T19:47:52Z",
  "region": "us-west-2",
  "detail": {
    ...
  }
{
```

Event content

ADD RESOURCES TO EVENT ENVELOPE

```
{
  "version": “1",
  "id": "10ec61ab-d758-61f7-96a0-592d003f6b0b",
  "source": "MyCompany.MyServerlessApp",
  "detail-type": "Product.ProductInfoUpdated",
  "account": "111122223333",
  "time": "2022-09-15T19:47:52Z",
  "region": "us-west-2",
  "resources": [
    "MyServerlessApp-Product-ProductInfoUpdatedFunction-VAv4YNEz6ojM"
  ],
  "detail": {
    ...
  }
{
```

Event content

EVENT DETAIL FIELD

```
{
  ...,
  "detail": {
      ...,
     }
  }
}
```

Event content

ADD EVENT METADATA

```
{
  ...,
  "detail": {
      "metadata": {
        "correlation_id": "6f9552ee-4e22-46dc-a385-c1995c11d882",
        "domain": "MyServerlessApp",
        "service": "Product",
        "environment": "prod"
     }
  }
}
```

Event content

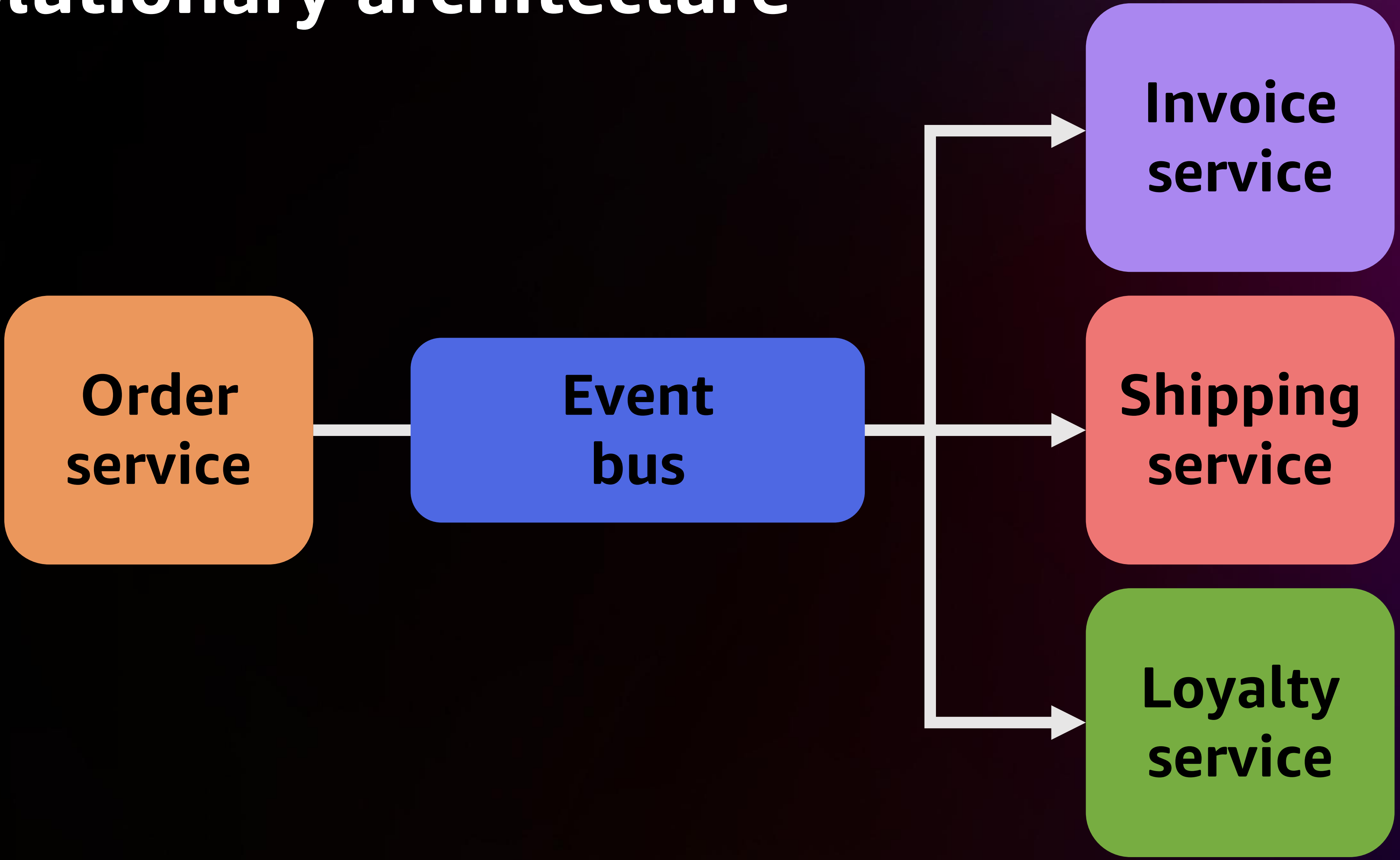
ADD EVENT DATA DETAIL

```
{
  ...,
  "detail": {
      "metadata": {
        "correlation_id": "6f9552ee-4e22-46dc-a385-c1995c11d882",
        "domain": "MyServerlessApp",
        "service": "Product",
        "environment": "prod"
     },
    "data": {
      "orderId": "a6a06b7b-c79b-4c10-b98e-5ac3e31da09f",
      "userId": "1c813a4f-1692-4901-b59a-ba8f4b790ce3"
    }
  }
}
```

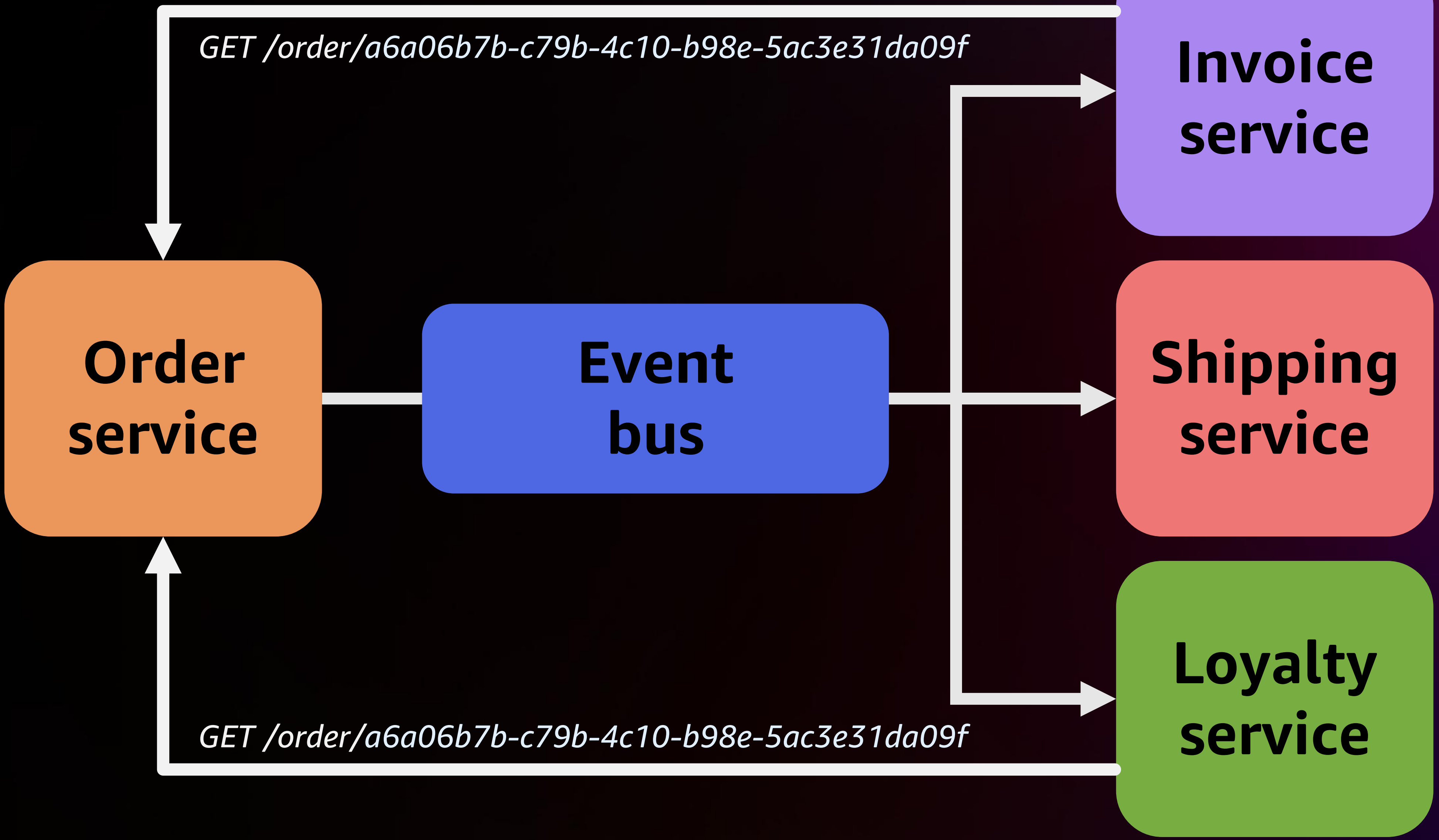
Add orderId/userId

to downstream services

Evolutionary architecture



Evolutionary architecture


GET /order/a6a06b7b-c79b-4c10-b98e-5ac3e31da09f
Invoice service
Order service
Event bus
Shipping service
Loyalty service
GET /order/a6a06b7b-c79b-4c10-b98e-5ac3e31da09f

Event content

ADD MORE DATA DETAIL

```
{
  ...,
  "detail": {
      "metadata": {
        ...,
     },
  "data": {
    "order": {
      "id": "3c947443-fd5f-4bfa-8a12-2aa348e793ae",
      "amount": 50,
      "deliveryAddress": {
        "postCode": "SW1A 1BA"
      }
    },
    "user": {
      "id": "09586e5c-9983-4111-8395-2ad5cfd3733b",
      "firstName": "Charles",
      "lastname": "Windsor",
      "email": "TheKing@royal.uk"
    }
```

Add order detail

Add user detail

Event content

ADD MORE DATA DETAIL

```
{
   ...,
   "detail": {
       "metadata": {
          ...,
      },
   "data": {
    "order": {
      "id": "3c947443-fd5f-4bfa-8a12-2aa348e793ae",
      "amount": 50,
      "deliveryAddress": {
        "postCode": "SW1A 1BA"

      }
    },
    "user": {
      "id": "09586e5c-9985-4111-8395-2ad5cfd3733b",
      "firstName": "Charles",
      "lastname": "Windsor",
      "email": "TheKing@royal.uk"
    }
```

Event-carried state transfer

Add order detail

Add user detail

Event content

ADD MORE DATA DETAIL

```
{
  ...,
  "detail": {
      "metadata": {
          ...,
      },
  "data": {
    "order": {
      "id": "3c947443-fd5f-4bfa-8a12-2aa348e793ae",
      "amount": 50,
      "deliveryAddress": {
        "postCode": "SW1A 1BA"
      }
    },
    "user": {
      "id": "09586e5c-9985-4111-8395-2ad5cfd3733b",
      "firstName": "Charles",
      "lastname": "Windsor",
      "email": "TheKing@royal.uk"
    }
```

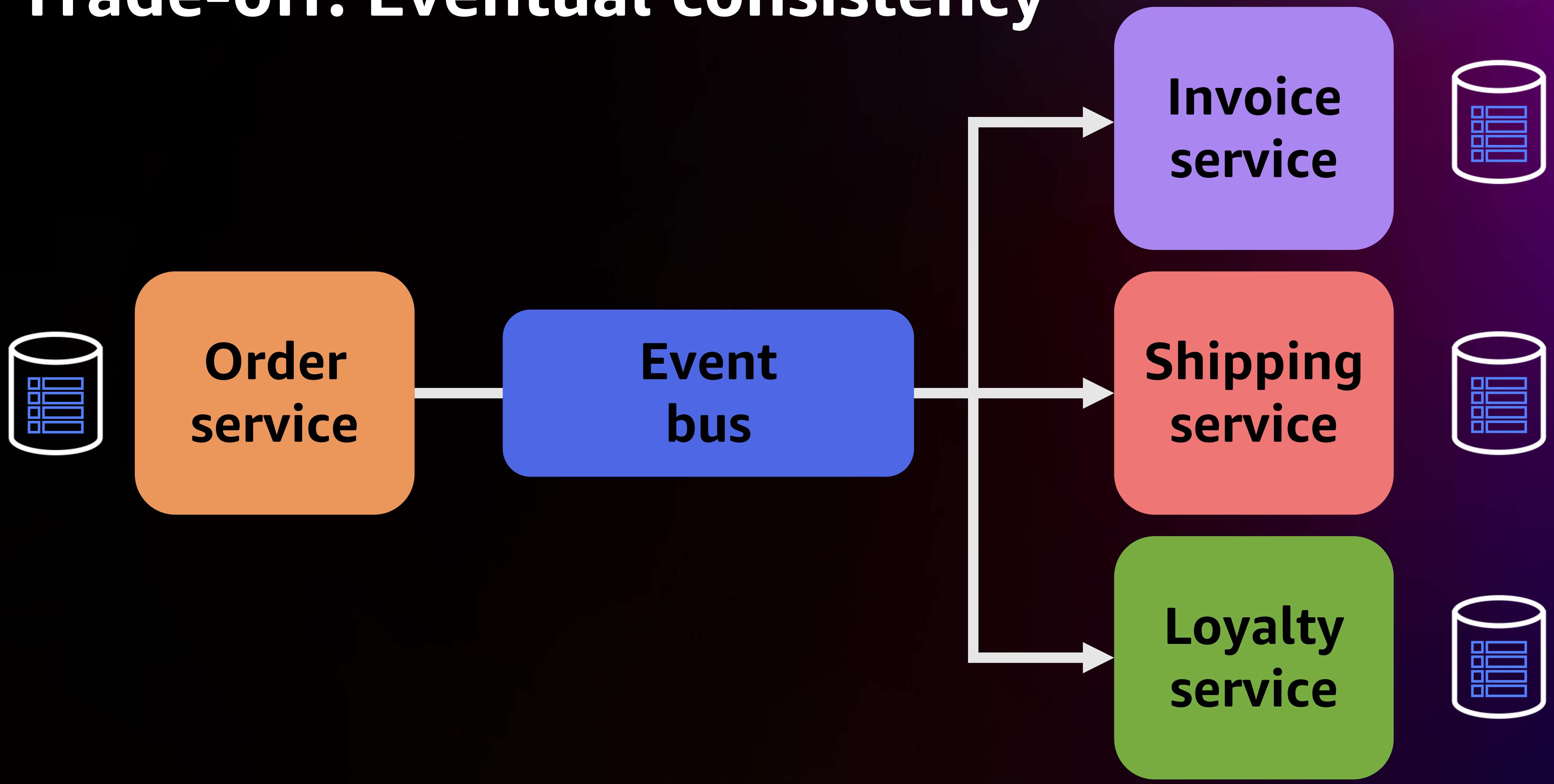
Add order detail

Add user detail

Event-carried state transfer

Invoice service

Trade-off: Eventual consistency



Exposing implementation details

```
{
 ....,
 "detail": {
   "metadata": {
    ....,
  },
 "data": {
  "order": {
   "id": "3c947443-fd5f-4bfa-8a12-2aa348e793ae",
   "amount": 50,
   "deliveryAddress": {
    "postCode": "SW1A 1BA"
   }
  },
  "user": {
   "id": "09586e5c-9983-4111-8395-2ad5cfd3733b",
   "firstName": "Charles",
   "lastname": "Windsor",
   "email": "TheKing@royal.uk"
  }
```

amount in what?

address details enough?

PII information

Event state: Best practices

- Events are the language of serverless applications
- Embrace asynchronous and eventual consistency
- Use one or multiple messaging services
- Enrich events with content and metadata
- Event-carried state transfer: pass state as events
- Consider trade-offs

Service-full serverless

(compose, configure, then code)

A serverless application

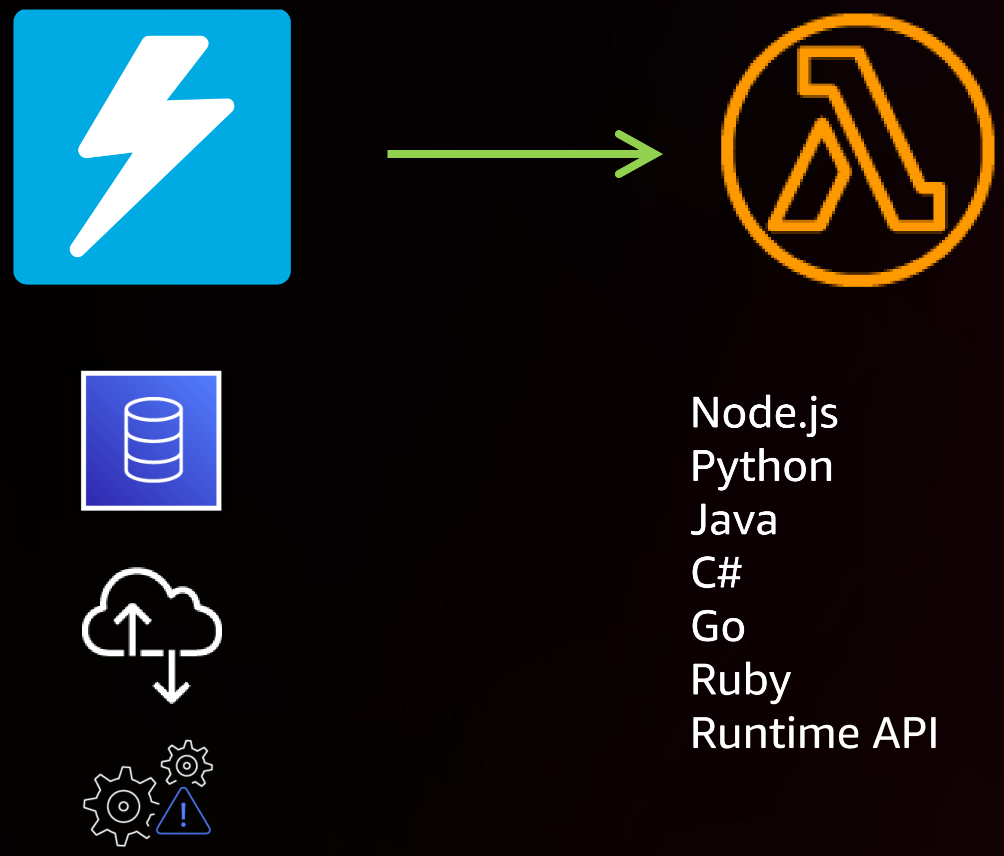
AWS Lambda

A serverless application

Function

Node.js
Python
Java
C#
Go
Ruby
Runtime API

A serverless application



Event source

Function

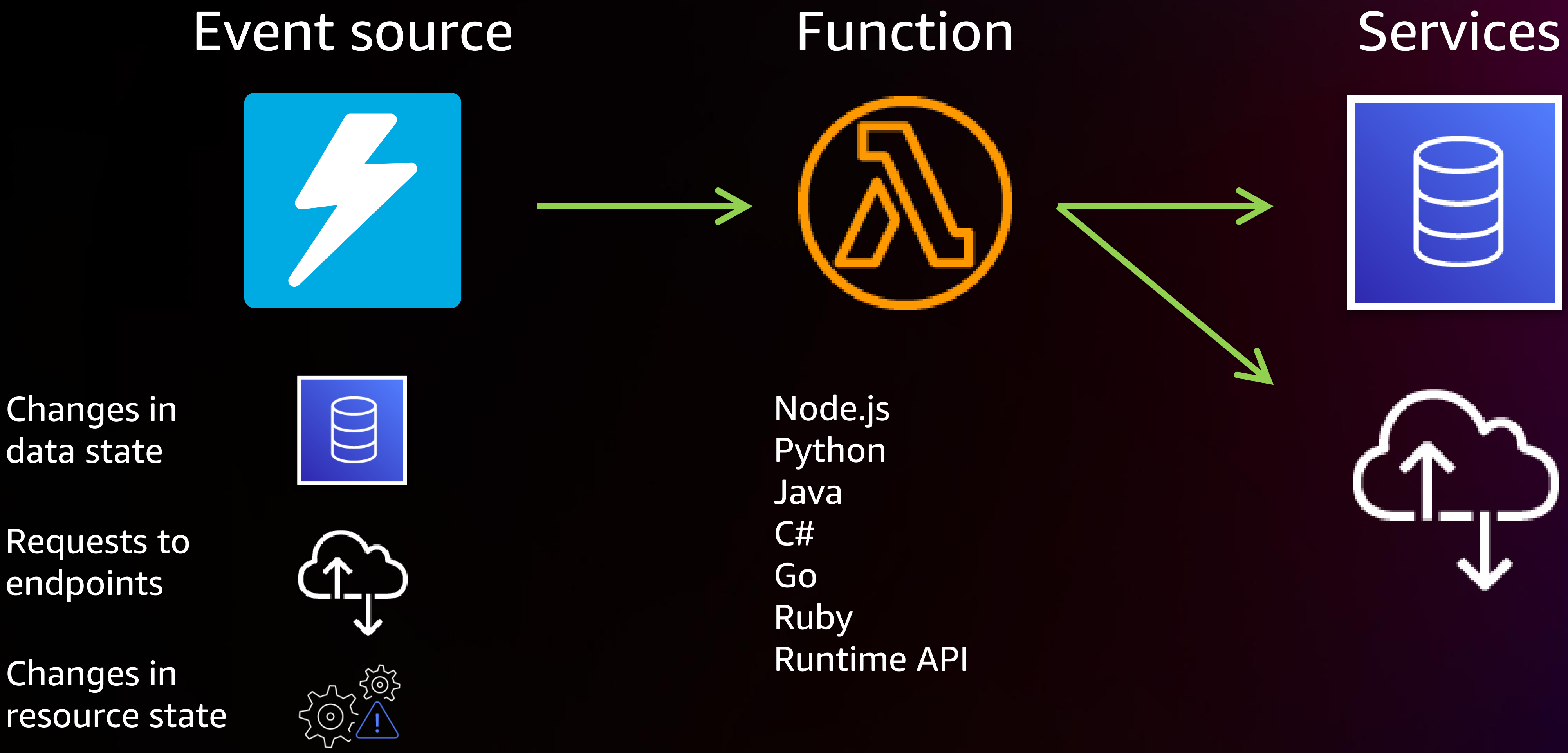
Changes in
data state

Requests to
endpoints

Changes in
resource state

Node.js
Python
Java
C#
Go
Ruby
Runtime API

A serverless application



Event source

Function

Services

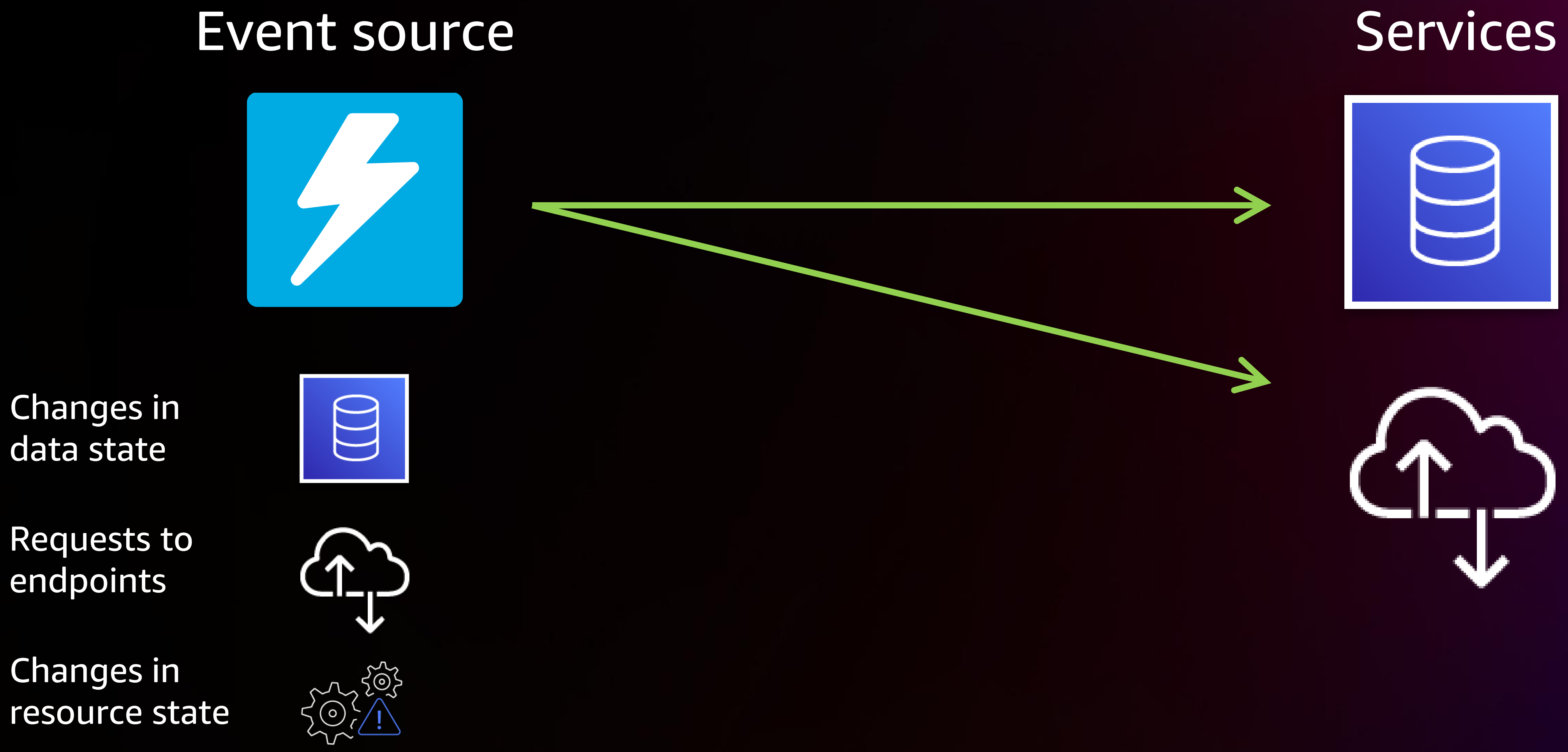
Changes in data state

Requests to endpoints

Changes in resource state

Node.js
Python
Java
C#
Go
Ruby
Runtime API

A serverless application



Event source

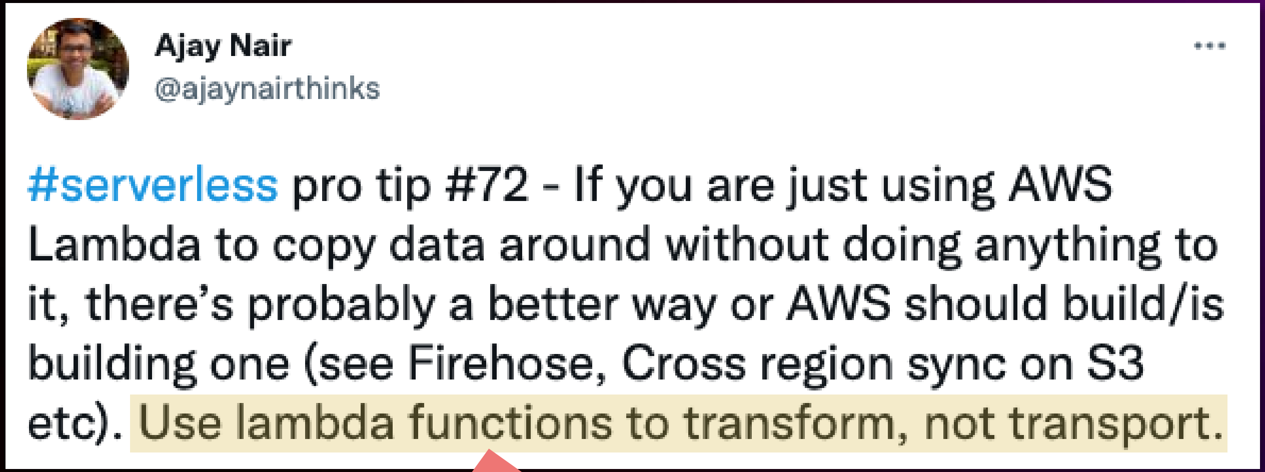
Services

Changes in data state

Requests to endpoints

Changes in resource state

Is an AWS Lambda function even needed?



Ajay Nair
@ajaynairthinks

#serverless pro tip #72 - If you are just using AWS Lambda to copy data around without doing anything to it, there's probably a better way or AWS should build/is building one (see Firehose, Cross region sync on S3 etc). Use lambda functions to transform, not transport.

Use AWS Lambda functions to transform, not transport

Image by Clker-Free-Vector-images from Pixabay

How much

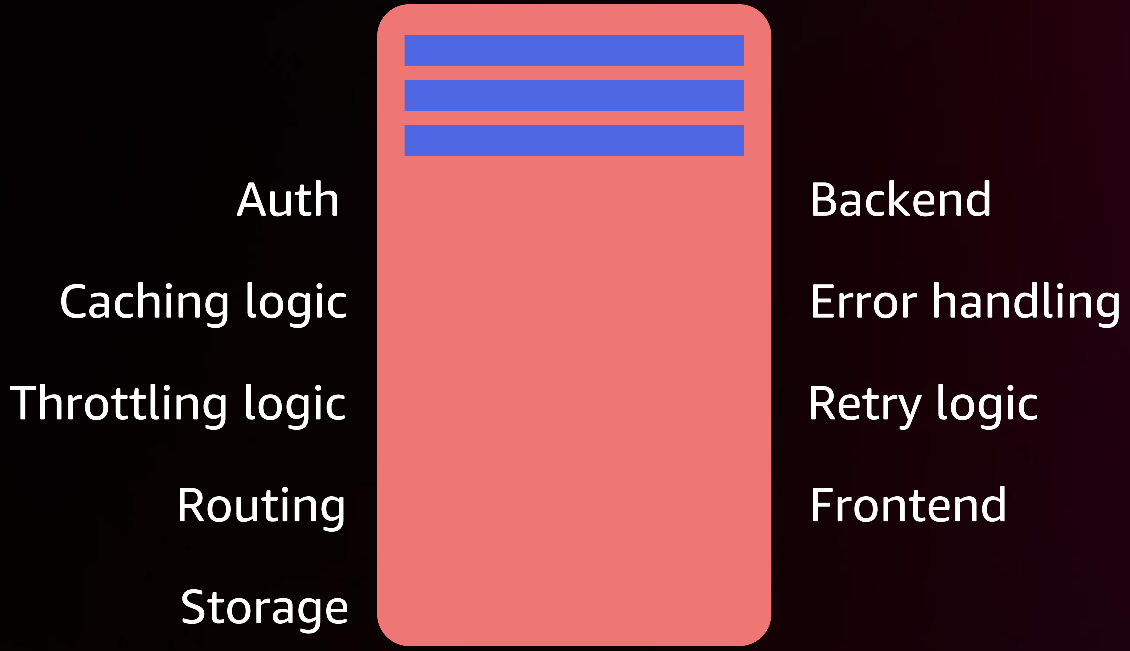
LOGIC

are you squeezing in to your code?

How little

are you actually invoking an AWS Lambda function for?

Consider a “traditional” app



Auth

Caching logic

Throttling logic

Routing

Storage

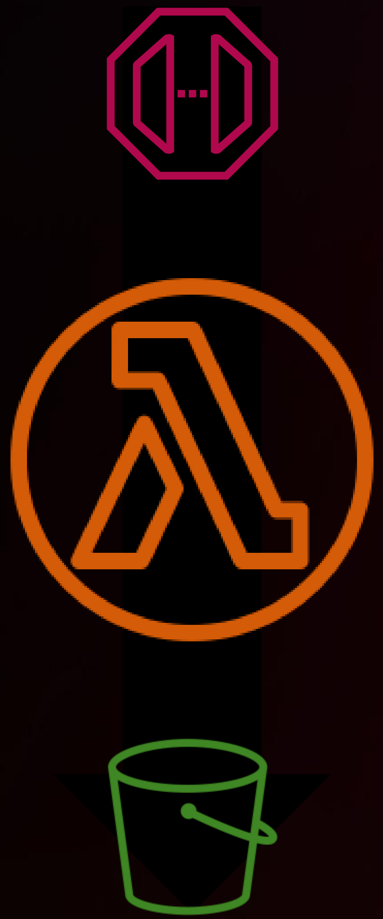
Backend

Error handling

Retry logic

Frontend

The lift and shift



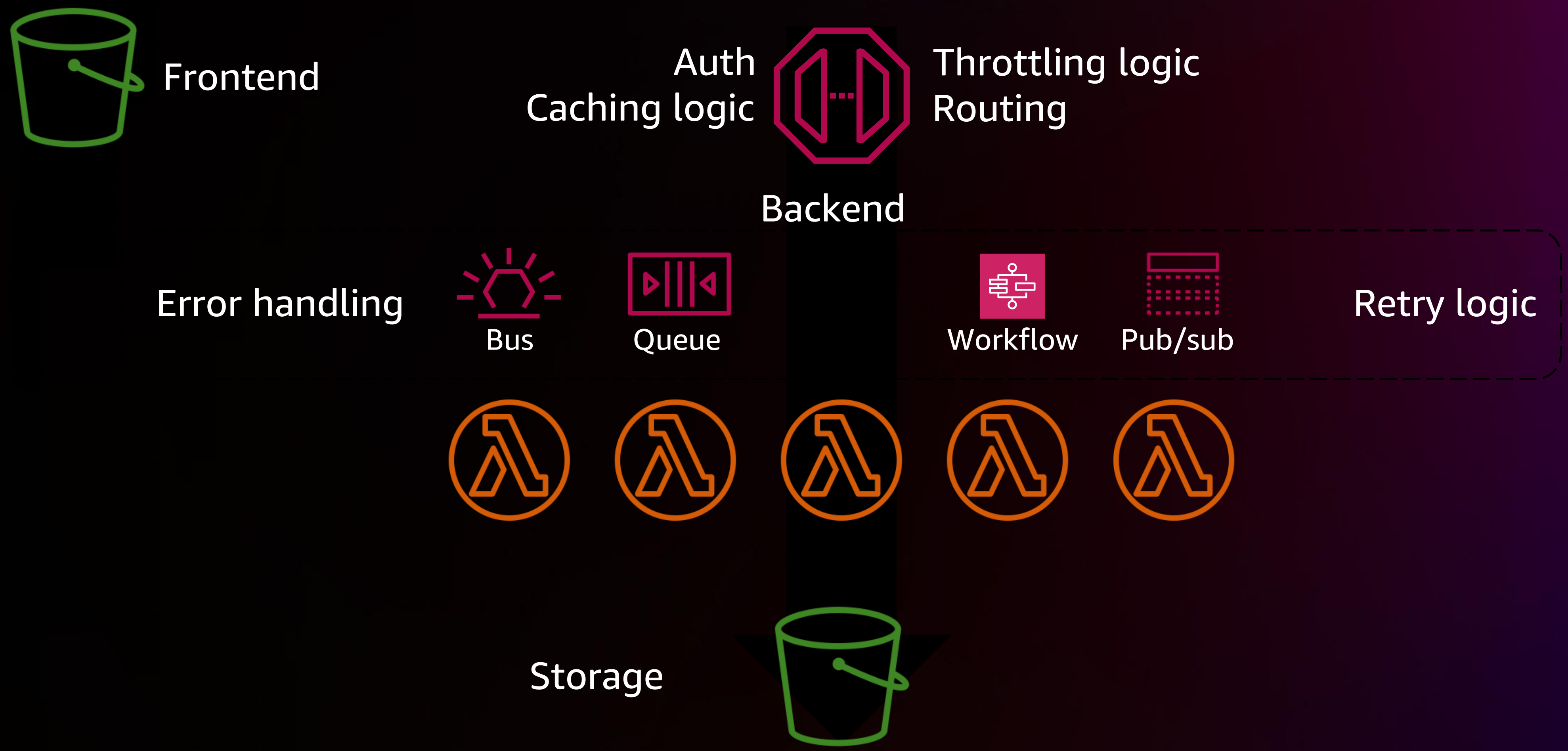
Auth
Caching logic
Throttling logic
Routing

Backend
Error handling
Retry logic
Frontend

Storage

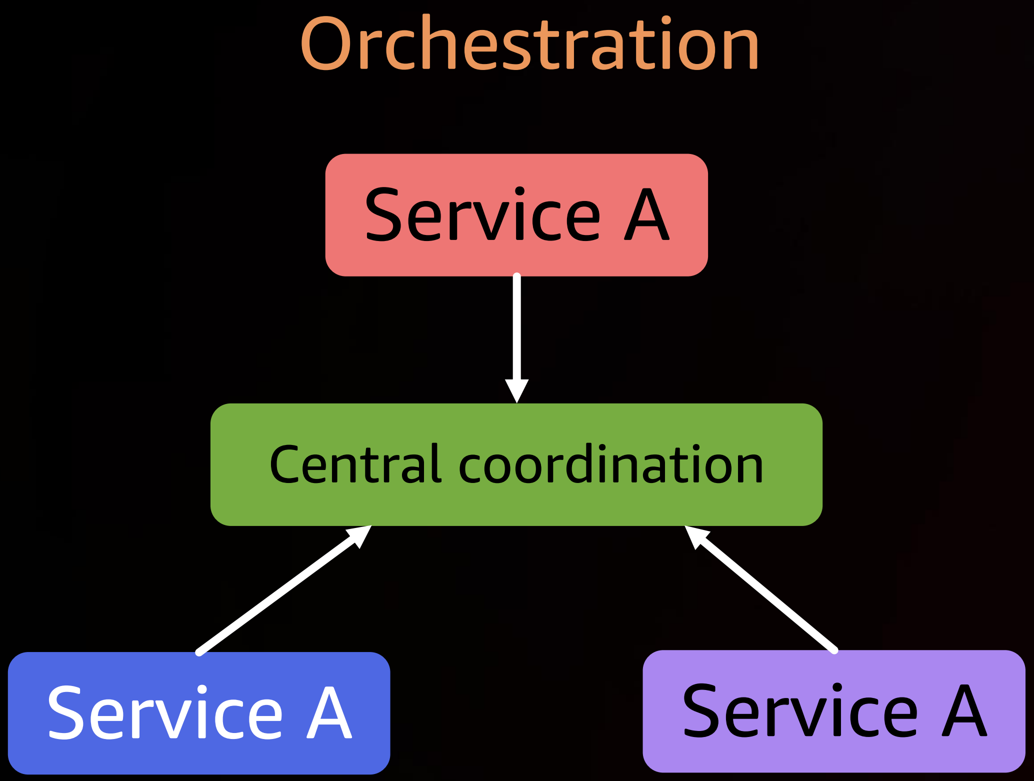
The migration

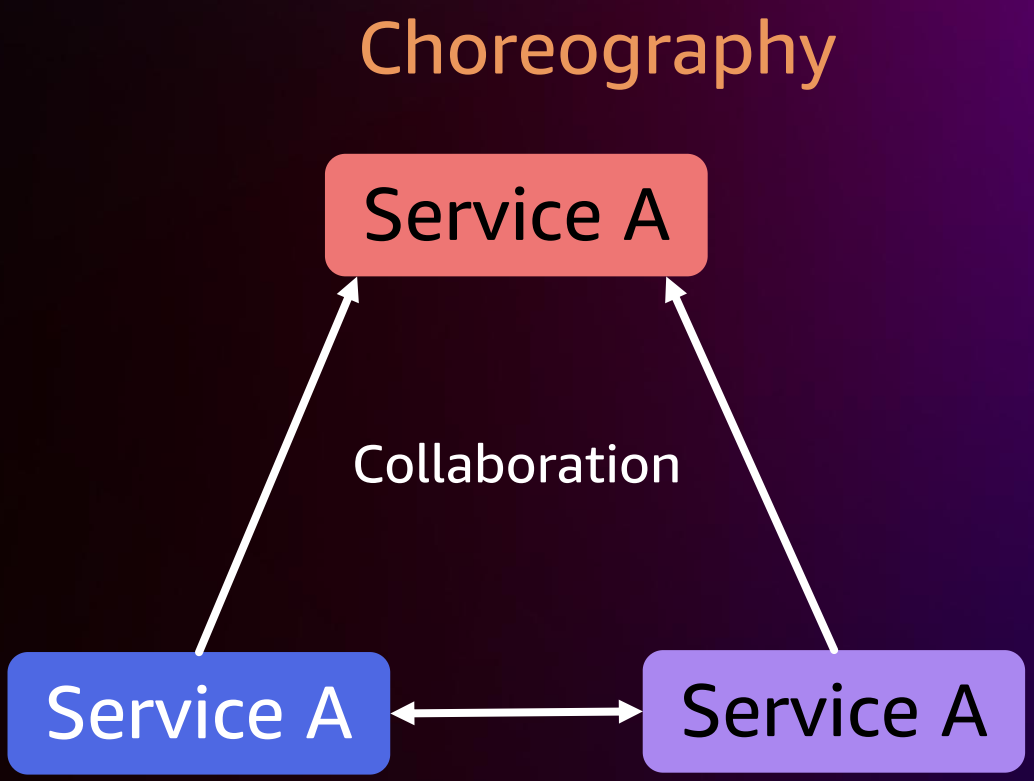
Offload processes to managed services



Orchestration/choreography as configuration

AS YOU GROW AND EXPAND, COORDINATION IS MORE CHALLENGING



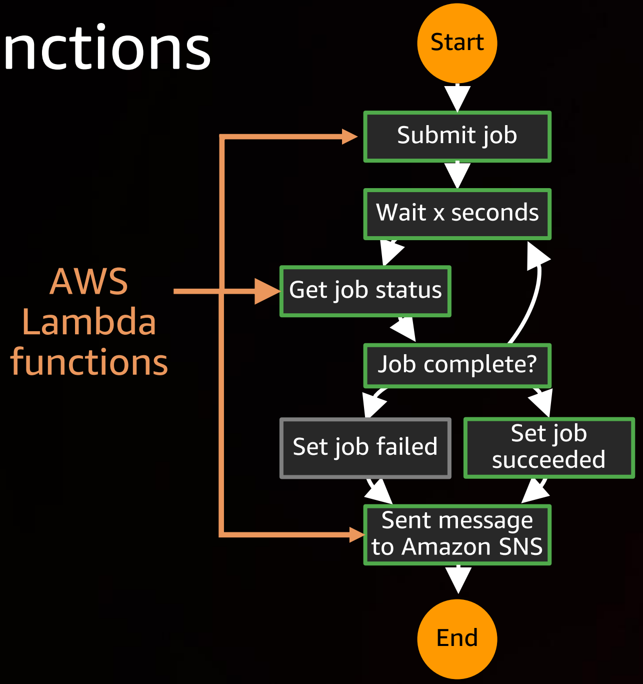


Orchestration/choreography as configuration

AS YOU GROW AND EXPAND, COORDINATION IS MORE CHALLENGING

Orchestration

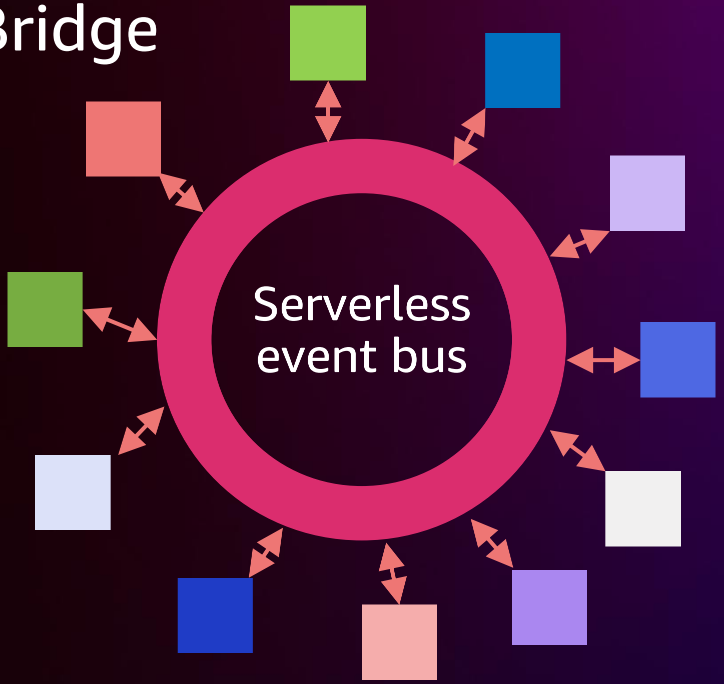
AWS Step Functions



Coordinate the components of distributed applications and microservices using visual workflows

Choreography

Amazon EventBridge



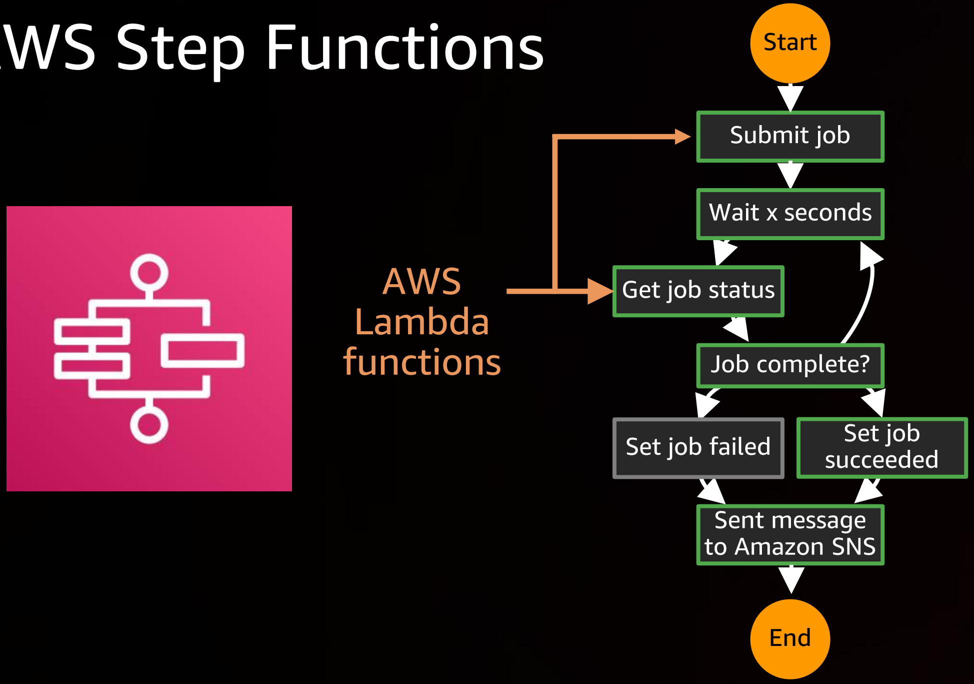
Produce and consume messages from a serverless event bus; services don’t need to know about each other, just about the bus

Orchestration/choreography as configuration

AS YOU GROW AND EXPAND, COORDINATION IS MORE CHALLENGING

Orchestration

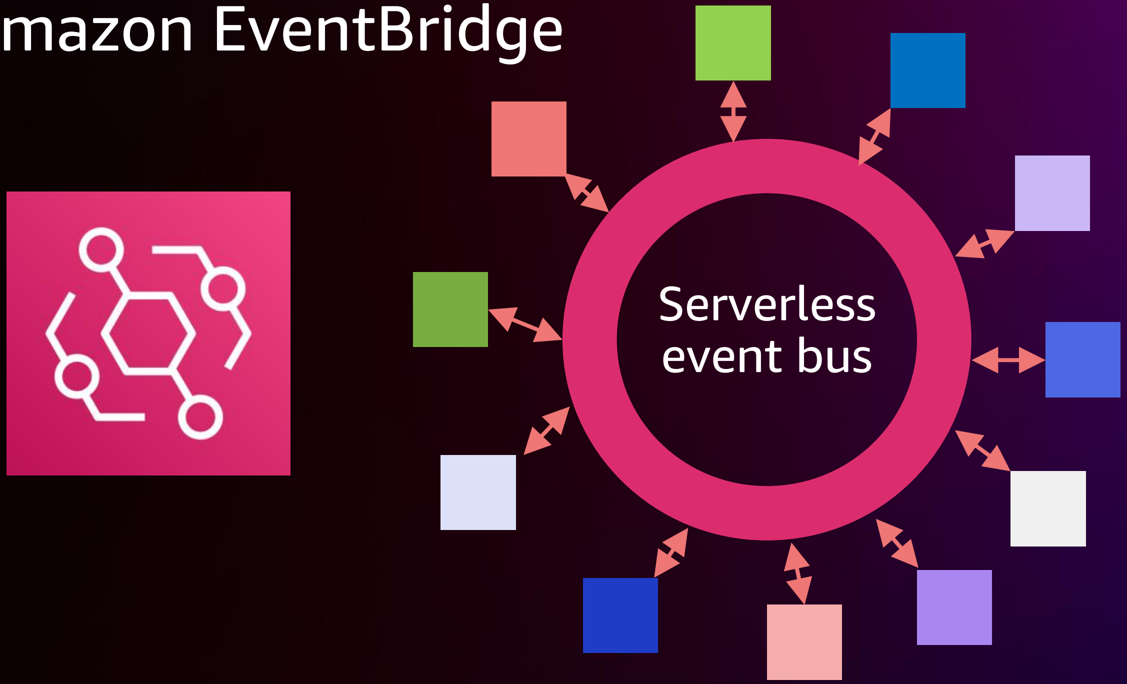
AWS Step Functions



AWS SDK integration

Choreography

Amazon EventBridge



API destinations

AWS Step Functions intrinsic functions

Arrays

JSON data manipulation

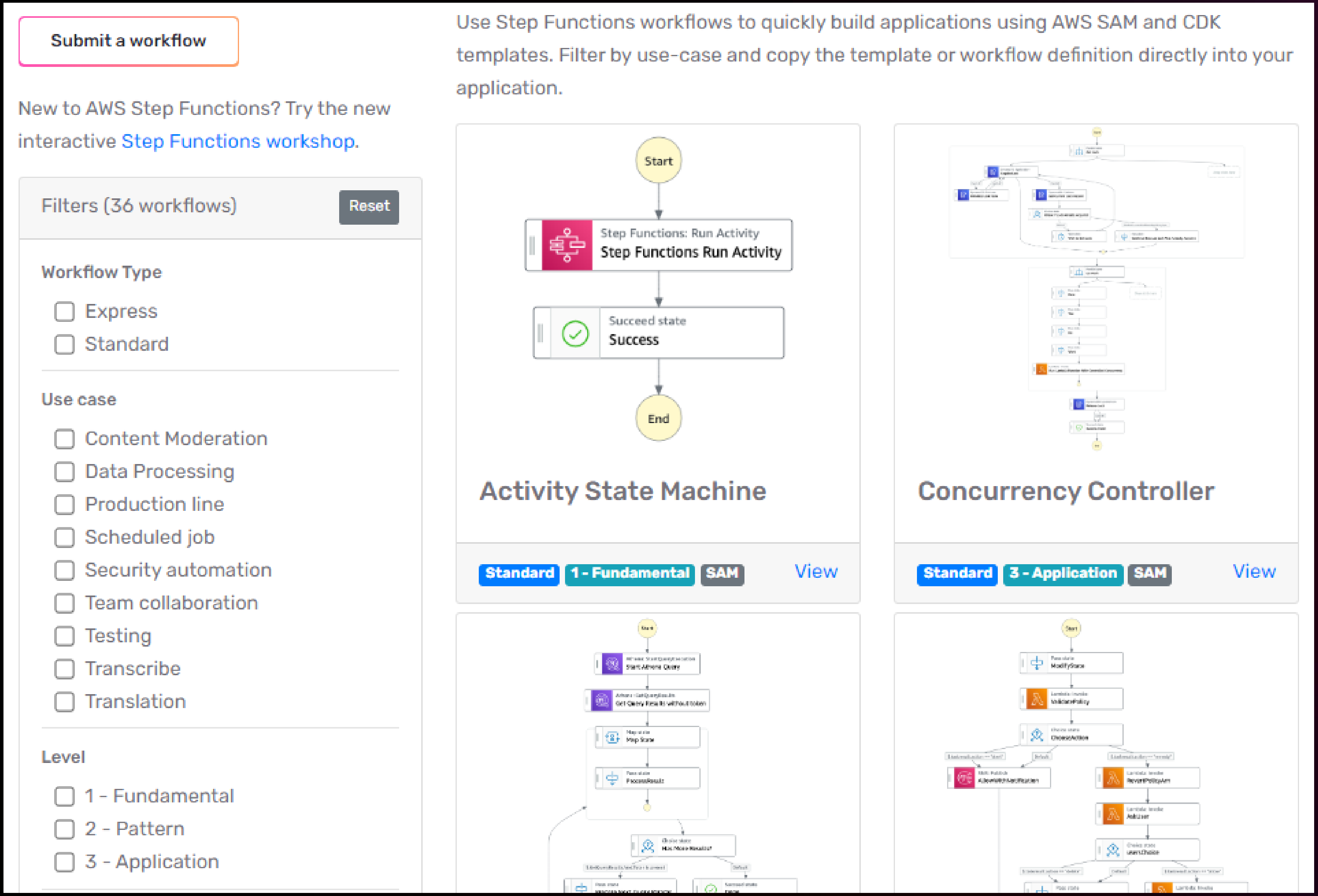
Encoding and decoding

Math operations

String operations

Unique identifier generation

Serverless workflows collection



Submit a workflow

New to AWS Step Functions? Try the new interactive Step Functions workshop.

Filters (36 workflows) Reset

Workflow Type

- □ Express
- □ Standard

Use case

- □ Content Moderation
- □ Data Processing
- □ Production line
- □ Scheduled job
- □ Security automation
- □ Team collaboration
- □ Testing
- □ Transcribe
- □ Translation

Level

- □ 1 - Fundamental
- □ 2 - Pattern
- □ 3 - Application

Use Step Functions workflows to quickly build applications using AWS SAM and CDK templates. Filter by use-case and copy the template or workflow definition directly into your application.

Activity State Machine

Standard | 1 - Fundamental | SAM — View

Concurrency Controller

Standard | 3 - Application | SAM — View

s12d.com/workflows

The fastest and lowest-cost Lambda function is the one you remove and replace with a built-in integration

Service-full serverless: Best practices

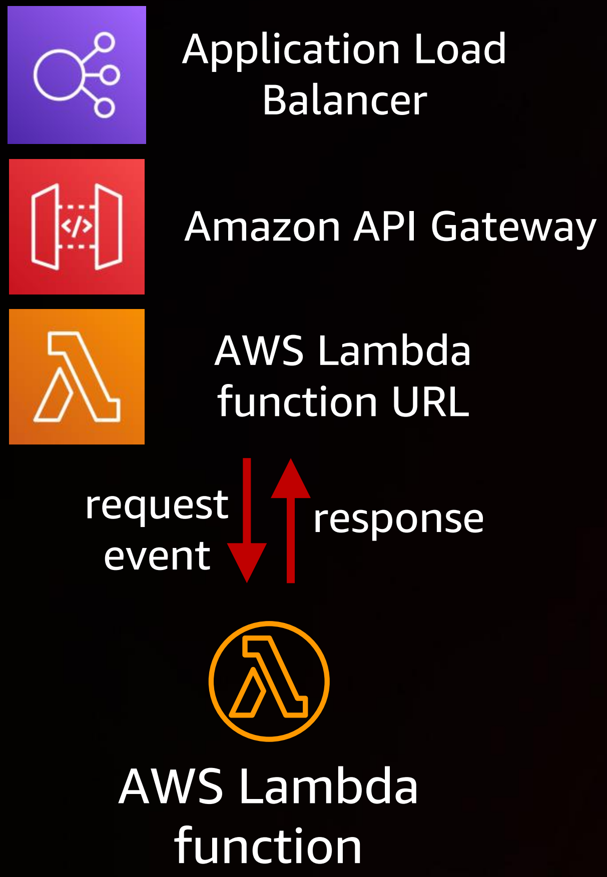
- Use service integrations where possible
- Code is a liability = prefer configuration over code
- Use AWS Lambda to transform, not transport
- Avoid monolithic services and functions
- Orchestrate workflows with Step Functions
- Choreograph events with EventBridge

Fabulous functions

The AWS Lambda invocation model

Synchronous

request/response



Application Load Balancer

Amazon API Gateway

AWS Lambda function URL

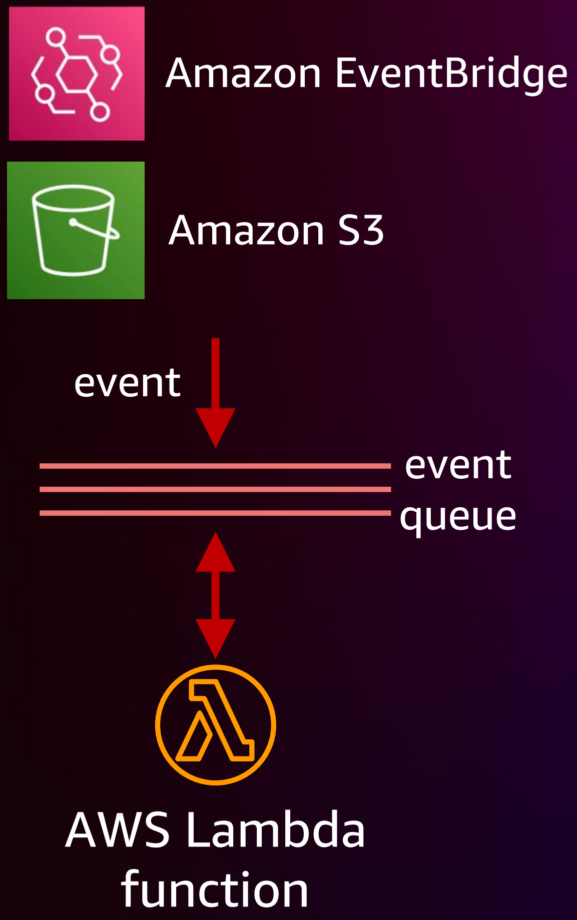
request event

response

AWS Lambda function

Asynchronous

event



Amazon EventBridge

Amazon S3

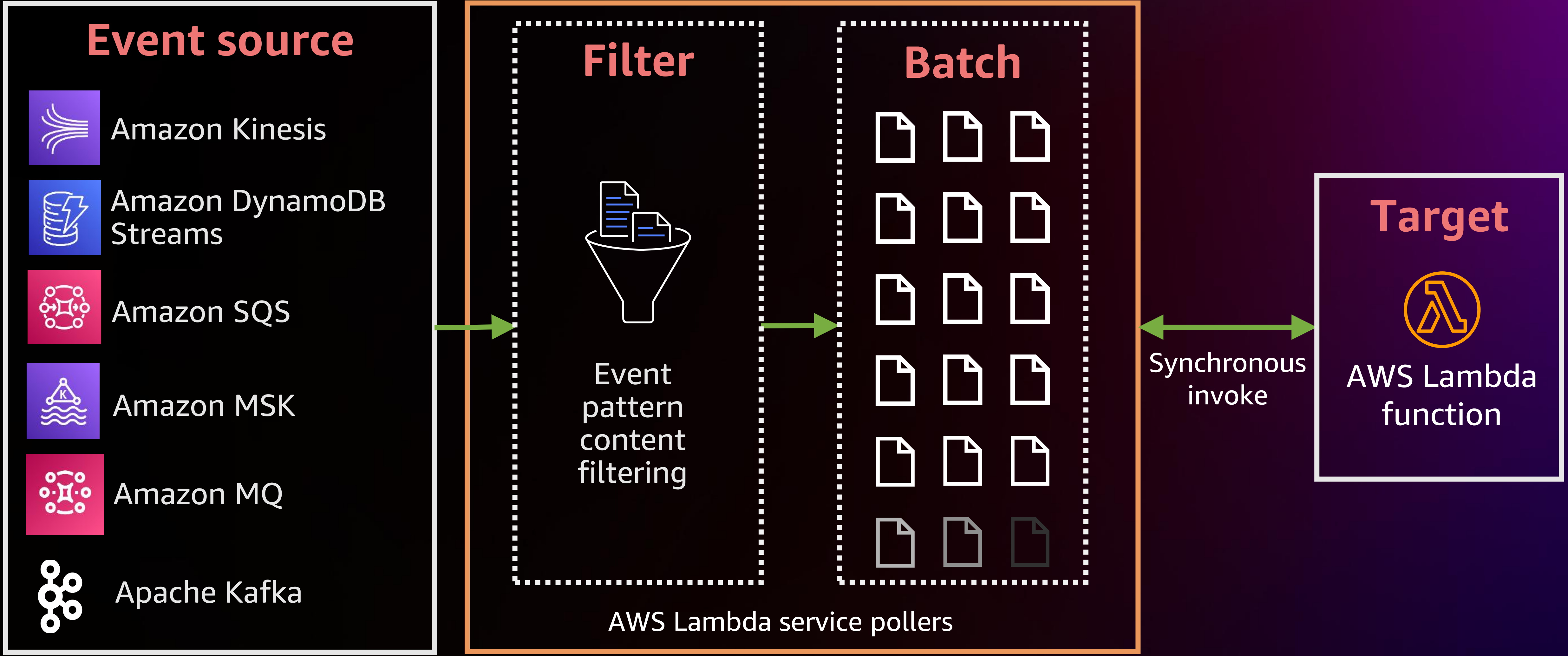
event

event queue

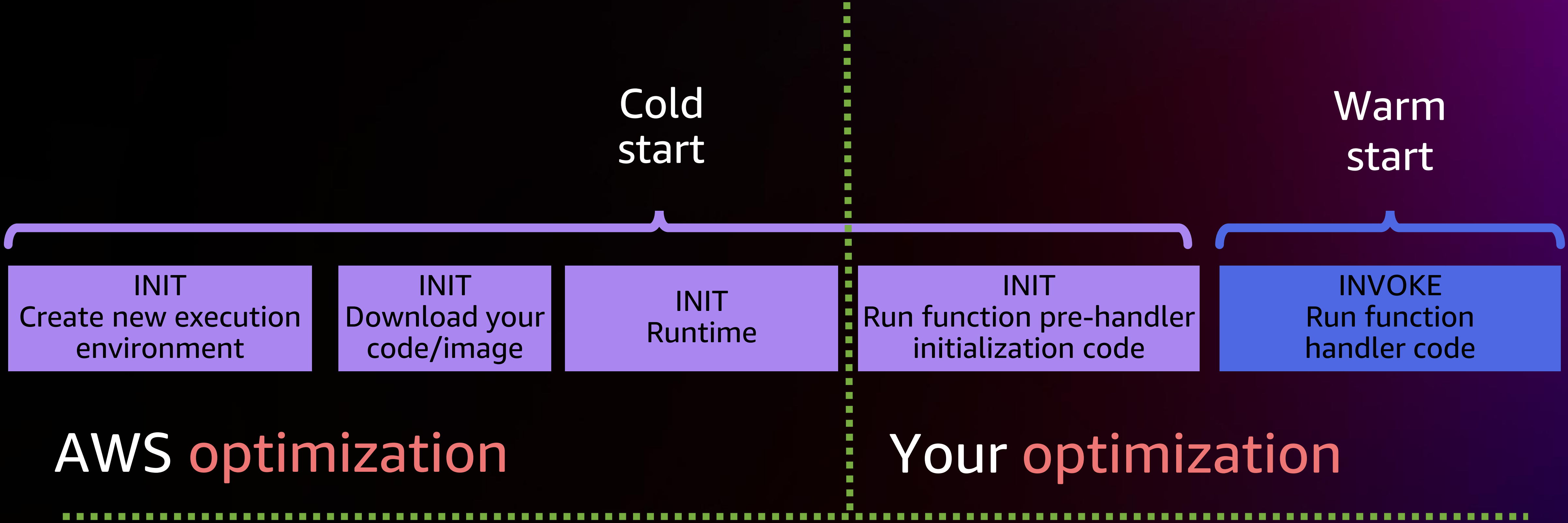
AWS Lambda function

Lambda event source mappings

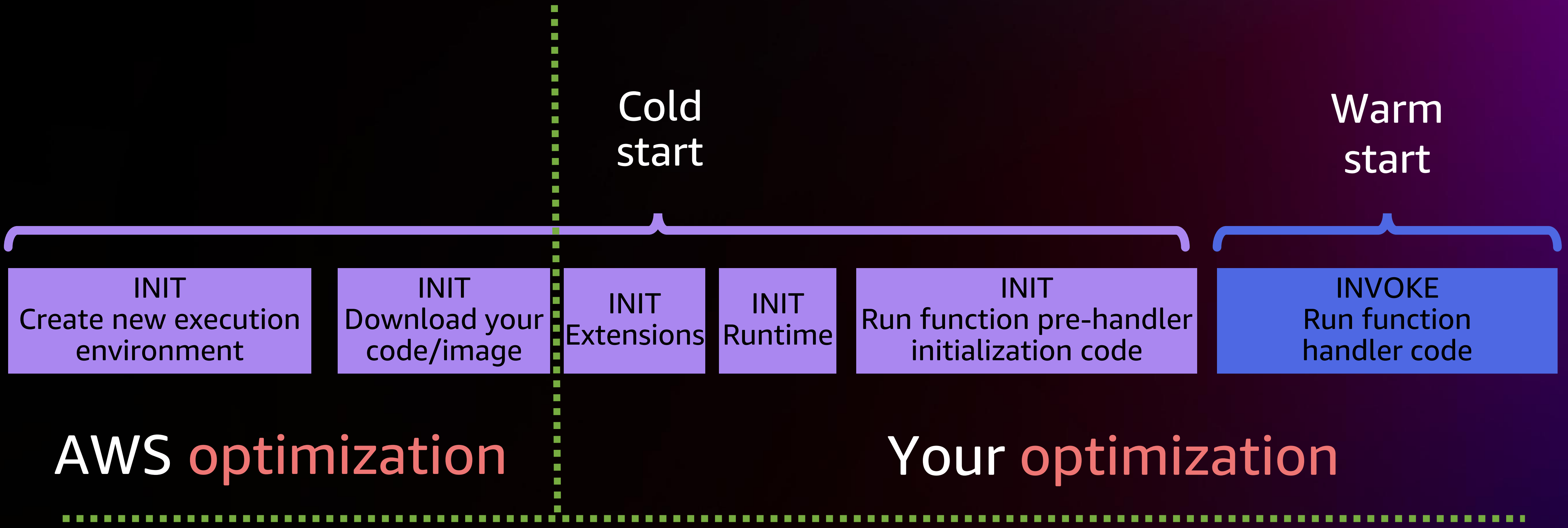
LAMBDA RESOURCE THAT READS FROM AN EVENT SOURCE AND INVOKES A LAMBDA FUNCTION



AWS Lambda execution environment lifecycle



AWS Lambda execution environment lifecycle



Cold start

Warm start

INIT
Create new execution environment

INIT
Download your code/image

INIT
Extensions

INIT
Runtime

INIT
Run function pre-handler initialization code

INVOKE
Run function handler code

AWS optimization

Your optimization

When using extensions/runtime modifications

AWS Lambda cold starts and you

OCCURS WHEN A NEW EXECUTION ENVIRONMENT IS REQUIRED IN RESPONSE TO AN EVENT/REQUEST

Cold starts occur when . . .

- You
  - Scaling up
  - Configuring provisioned concurrency
  - Updating code/config
- AWS
  - Environment is refreshed
  - Failure in underlying resources
  - Rebalancing across Availability Zones

Typically

- Varies from <100 ms to >1 sec
- <1% of production workloads

Significantly reduced for VPC integration

AWS Lambda cold starts and you

OCCURS WHEN A NEW EXECUTION ENVIRONMENT IS REQUIRED IN RESPONSE TO AN EVENT/REQUEST

Cold starts occur when . . .

- You
  - Scaling up
  - Configuring provisioned concurrency
  - Updating code/config
- AWS
  - Environment is refreshed
  - Failure in underlying resources
  - Rebalancing across Availability Zones

Typically

- Varies from <100 ms to >1 sec
- <1% of production workloads

Significantly reduced for VPC integration

Runs pre-handler INIT code

```
Import sdk
Import http-lib
Import cheese-sandwich

Pre-handler-secret-getter()
Pre-handler-db-connect()

Function myhandler(event, context) {
....
```

Pre-handler INIT code: Best practices

Don’t load it if you don’t need it

- Import only what you need
- Optimize dependencies, SDKs, and other libraries to the specific modules required
- Minify/uglify production code
- Reduce deployment package size
- Avoid “monolithic” functions

Lazy initialize shared libraries

- Helps if there are multiple functions per file

Establishing connections

- Handle reconnections in handler
- Keep-alive in AWS SDKs

State during environment reuse

- Keep data for subsequent invocations
- Don’t store data and secrets you don’t want for subsequent invocations

Use provisioned concurrency

- On individual functions, no code changes

Optimize dependencies

Only use libraries that are specific to your workload

```
// const AWS = require('aws-sdk’)
const DynamoDB = require('aws-sdk/clients/dynamodb’) // 125ms faster


// const AWSXRay = require('aws-xray-sdk’)
const AWSXRay = require('aws-xray-sdk-core’) // 5ms faster


// const AWS = AWSXRay.captureAWS(require('aws-sdk’))
const dynamodb = new DynamoDB.DocumentClient()
AWSXRay.captureAWSClient(dynamodb.service) // 140ms faster
```

AWS SDK for JavaScript v3

s12d.com/jssdkv3

- Modularized packages: only import the dependencies you need
- ~3 MB package rather than 8 MB for v2
- TCP connection reuse on by default
    - Disable with `AWS_NODEJS_CONNECTION_REUSE_ENABLED=false`
- Middleware API using `.middlewareStack` method
- New Command API with `.send({command})`
- AWS X-Ray support with `.captureAWSv3Client`

- ES5: `const { DynamoDBClient } = require ("@aws-sdk/client-dynamodb");`
- ES6: `import { DynamoDBClient } from "@aws-sdk/client-dynamodb";`

Lazy initialize shared libraries: Python and boto3

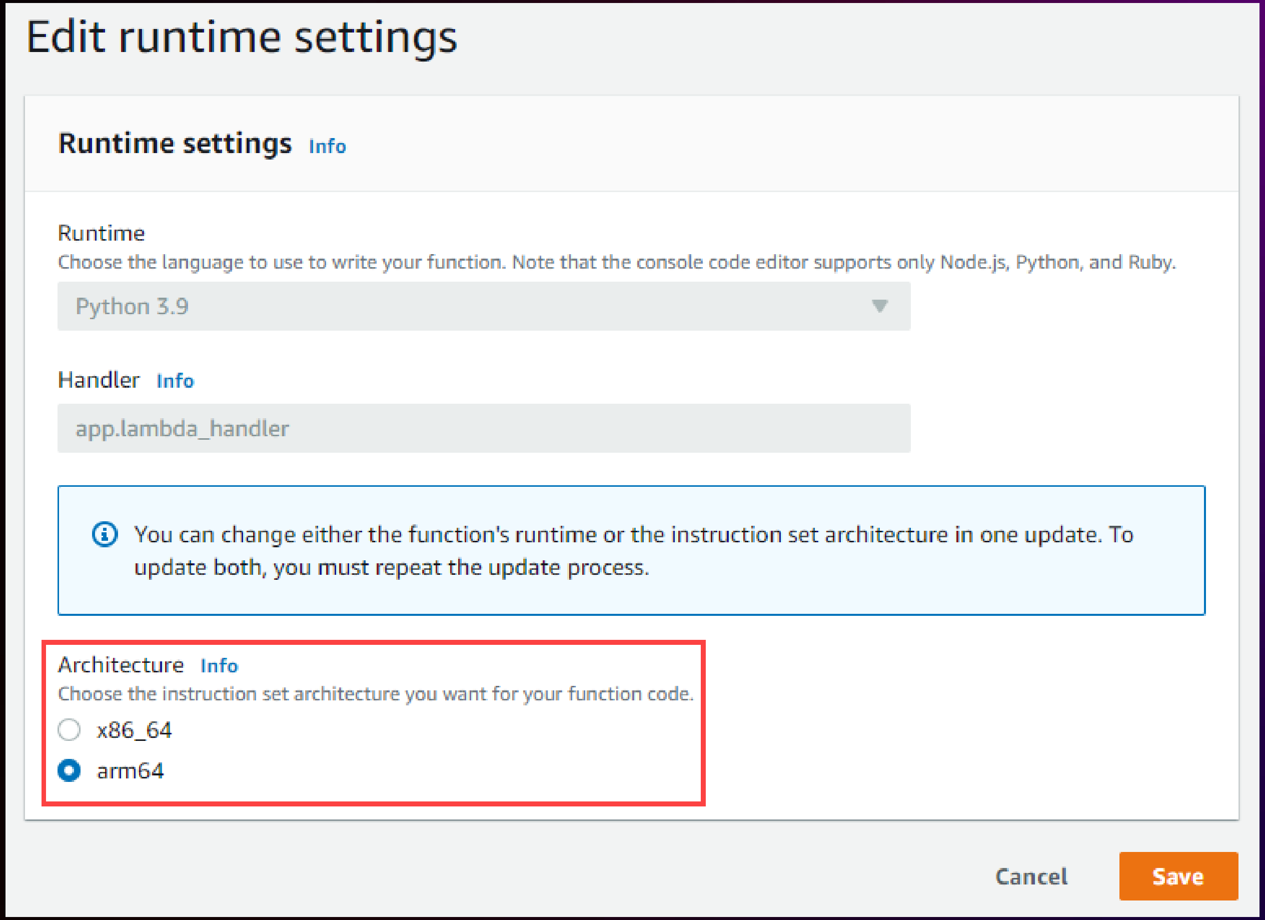
```
import boto3
s3_client = None
ddb_client = None

def get_objects_handler(event, context):
    if not s3_client:
        s3_client = boto3.client("s3")
    # business logic

def get_items_handler(event, context):
    if not ddb_client:
        ddb_client = boto3.client(”dynamodb")
    # business logic
```

AWS Lambda functions on AWS Graviton2

UP TO 34% BETTER PRICE-PERFORMANCE OVER X86-BASED AWS LAMBDA

You can target functions deployed with a container image or .zip file to run on x86-based or ARM-based processors powered by AWS Graviton2

- Interpreted and compiled-bytecode languages can run without modification
- Compiled languages and container images need to be recompiled for arm64
- Most AWS tools and SDKs support AWS Graviton2 transparently



Edit runtime settings

Runtime settings Info

Runtime

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.9

Handler Info

app.lambda_handler

You can change either the function's runtime or the instruction set architecture in one update. To update both, you must repeat the update process.

Architecture Info

Choose the instruction set architecture you want for your function code.

○ x86_64

● arm64

AWS Lambda function memory “power”

AWS Lambda exposes only a memory configuration control

- Between 128 MB and 10 GB in 1 MB increments

AWS Lambda proportionally allocates:

- CPU power
- Network bandwidth

If your code is memory-, CPU-, or network-bound, add more memory, which may improve performance and reduce cost

AWS Lambda: Larger functions

You can now configure AWS Lambda functions for

- **10 GB** in memory with up to
- **6 vCPUs** proportional to memory configuration

Run memory intensive workloads

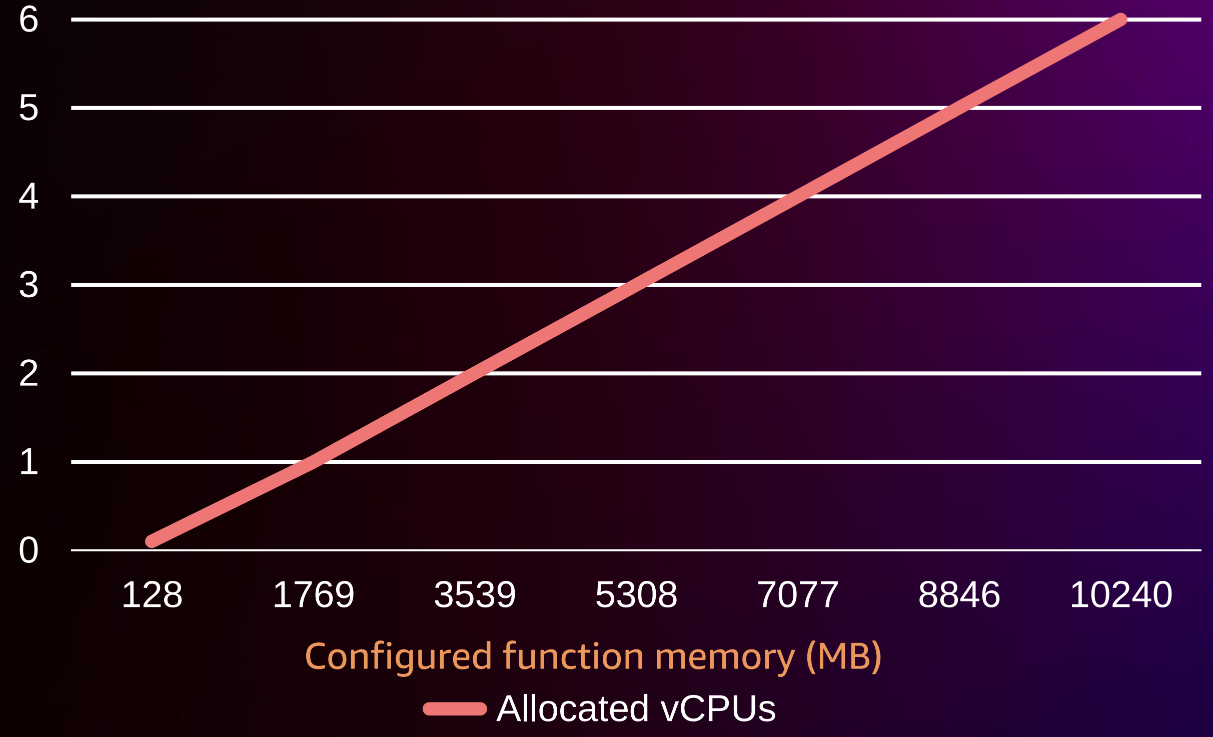
- Batch, ETL, analytics, media processing

Run compute intensive workloads

- Machine learning, genomics, gaming, HPC

>1 core: CPU-bound workloads will see gains, but need to multi-thread

Memory vs. number of vCPUs*



*Numbers are an approximation of vCPU power

Smart resource allocation

Match resource allocation to business logic

Stats for AWS Lambda function that calculates:

1,000 times all prime numbers **<= 1,000,000**

128 MB	11.722965 sec	$0.024628
256 MB	6.678945 sec	$0.028035
512 MB	3.194954 sec	$0.026830
1,024 MB	1.465984 sec	$0.024638

Green = Best **Red** = Worst

Smart resource allocation

Match resource allocation to business logic

Stats for AWS Lambda function that calculates:

1,000 times all prime numbers **<= 1,000,000**

128 MB	11.722965 sec	$0.024628
256 MB	6.678945 sec	$0.028035
512 MB	3.194954 sec	$0.026830
1,024 MB	1.465984 sec	$0.024638

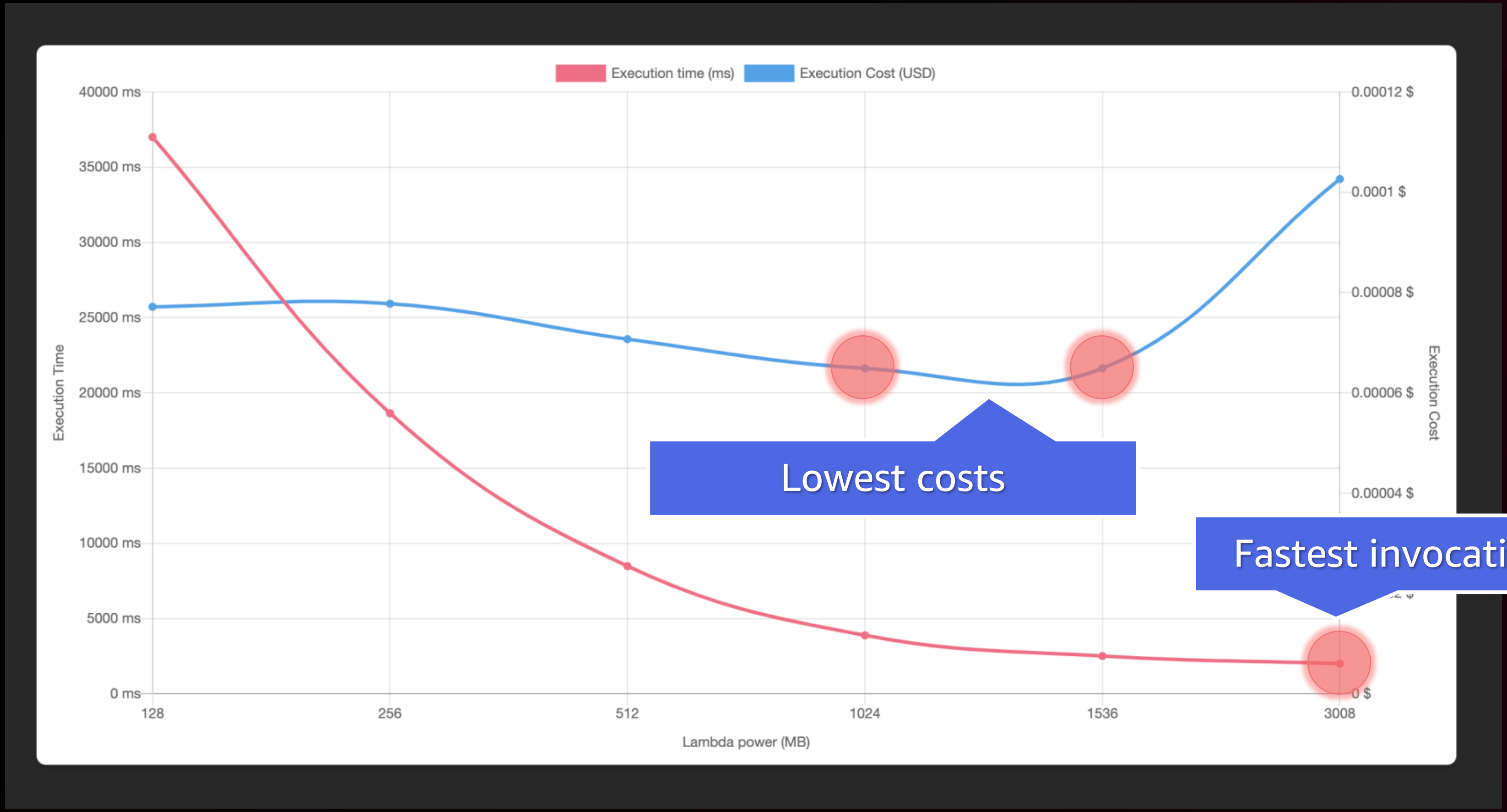
-10.256981 sec

+$0.00001

Green = Best **Red** = Worst

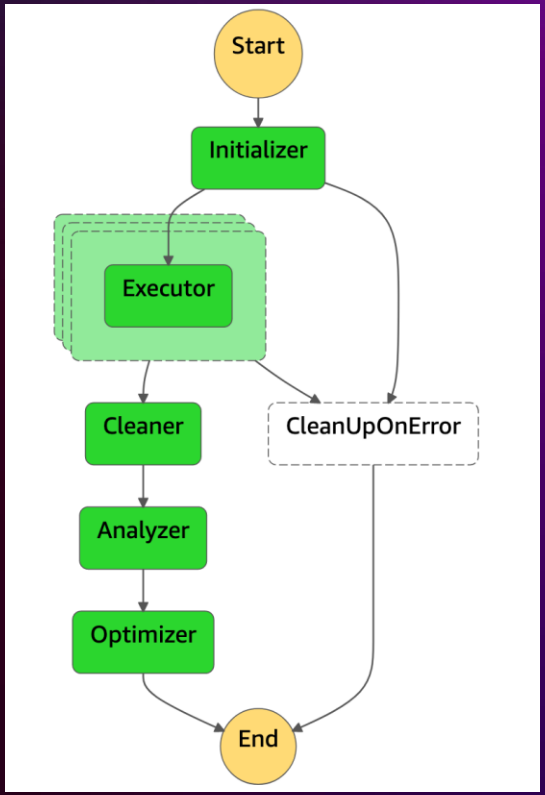
AWS Lambda Power Tuning

DATA-DRIVEN COST AND PERFORMANCE OPTIMIZATION FOR AWS LAMBDA



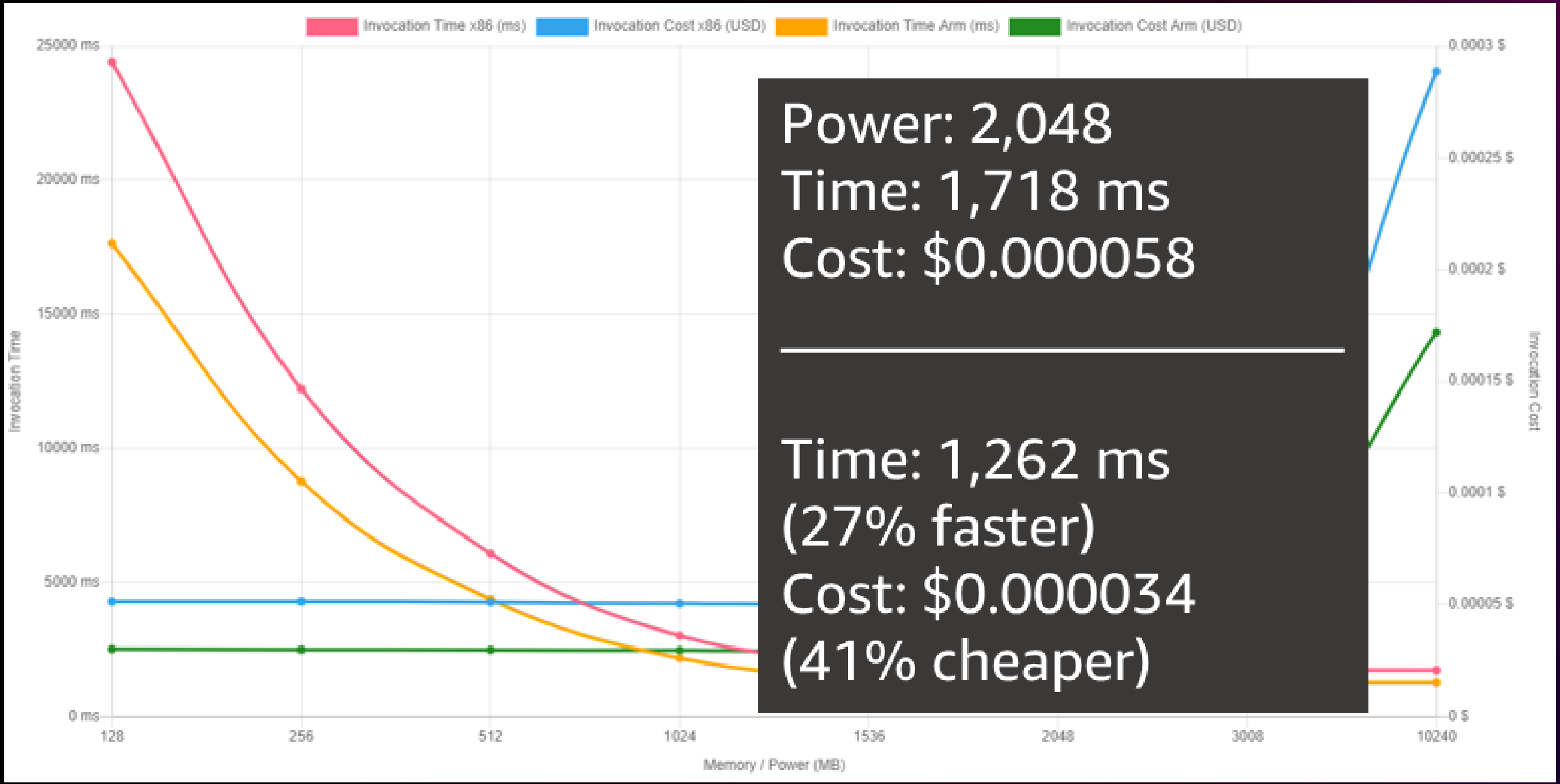
Lowest costs

Fastest invocation time



s12d.com/lambda-power-tuning

Comparing x86 to ARM/AWS Graviton2

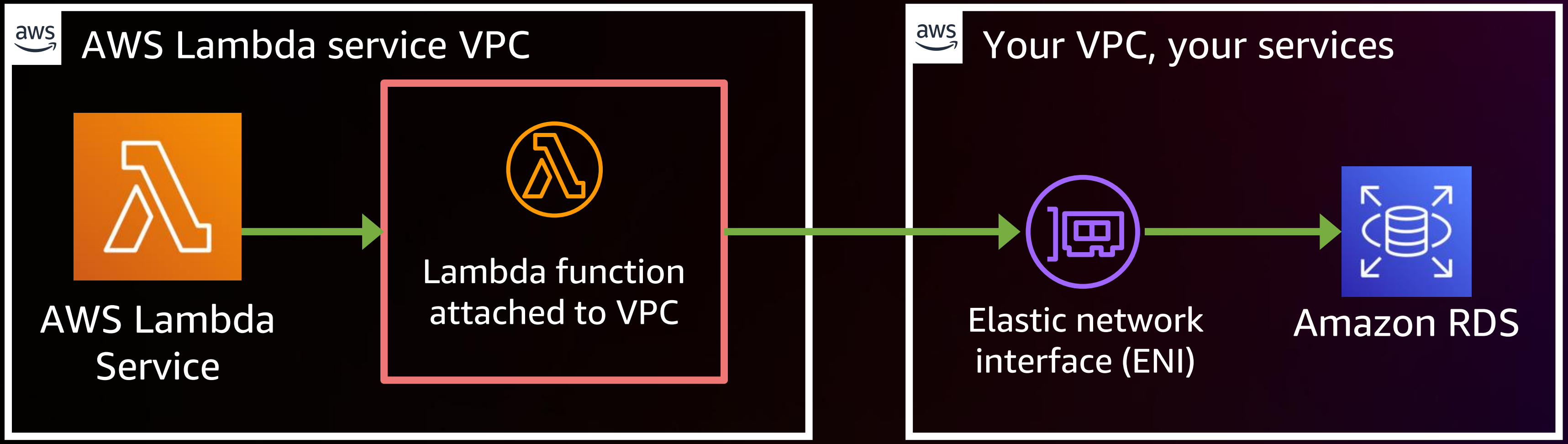


Power: 2,048
Time: 1,718 ms
Cost: $0.000058

Time: 1,262 ms
(27% faster)
Cost: $0.000034
(41% cheaper)

Attaching Lambda functions to your VPC

- If you need to connect to services within a VPC
- Only if you need additional network routing, access control, and auditing
- No need to avoid connectivity to AWS services over the internet
- All internal AWS traffic uses the AWS global backbone rather than traversing the internet
- If you do need to use a VPC, Lambda works great!



Understanding concurrency

Concurrency is the number of requests that your function is serving at any given time

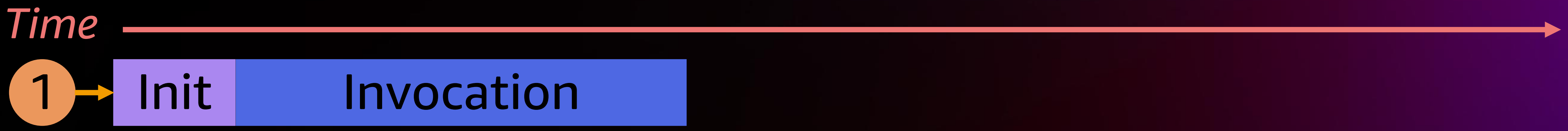
A single AWS Lambda execution environment can only process a single event at a time

- Regardless of event source or invocation model
- Batches pulled from Kinesis Data Streams, Amazon SQS, or Amazon DynamoDB Streams count as a single event

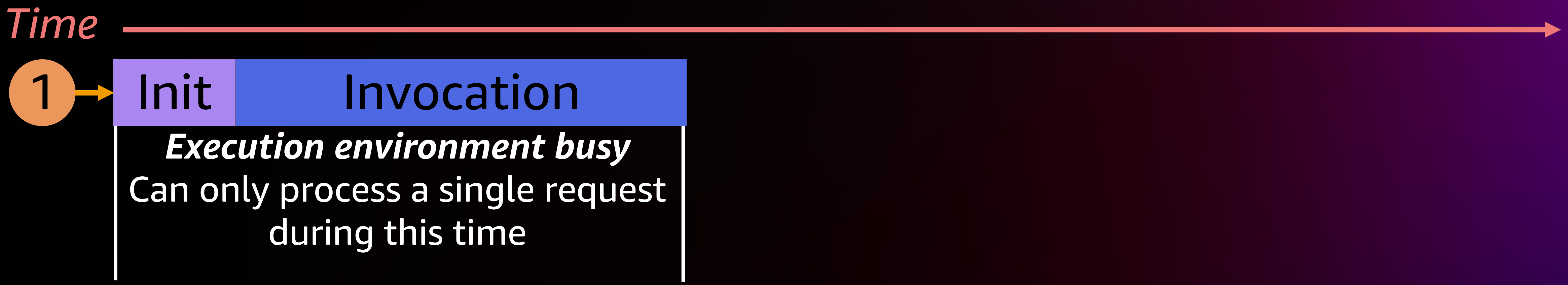
Concurrent requests require new execution environments to be created

- Limited in concurrency by burst rate per account per Region

Understanding concurrency



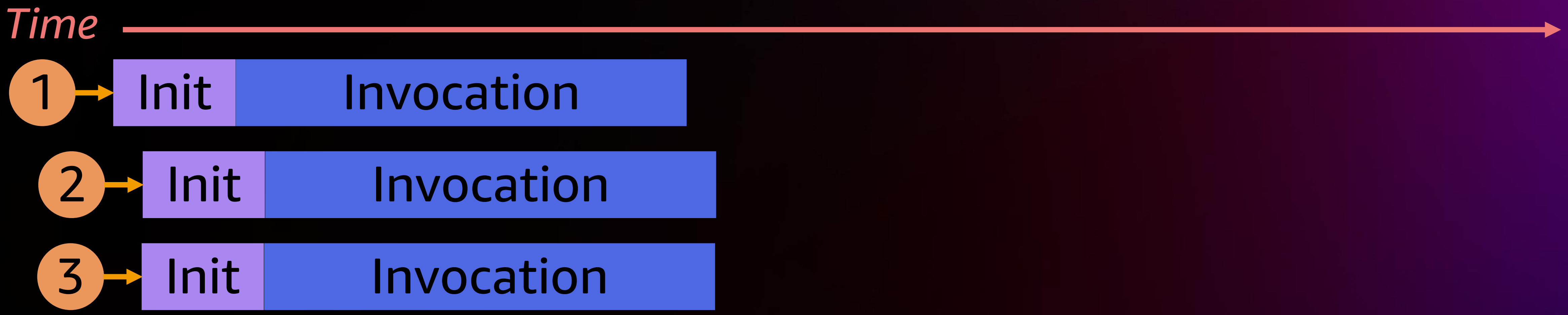
Understanding concurrency



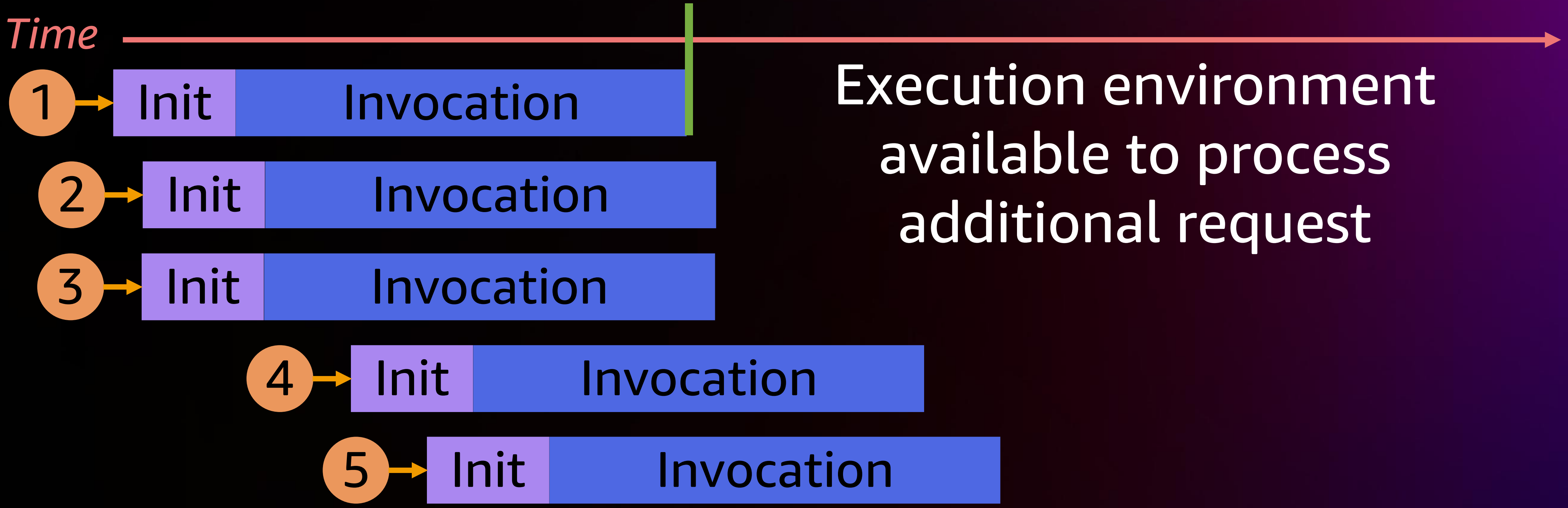
Execution environment busy

Can only process a single request during this time

Understanding concurrency

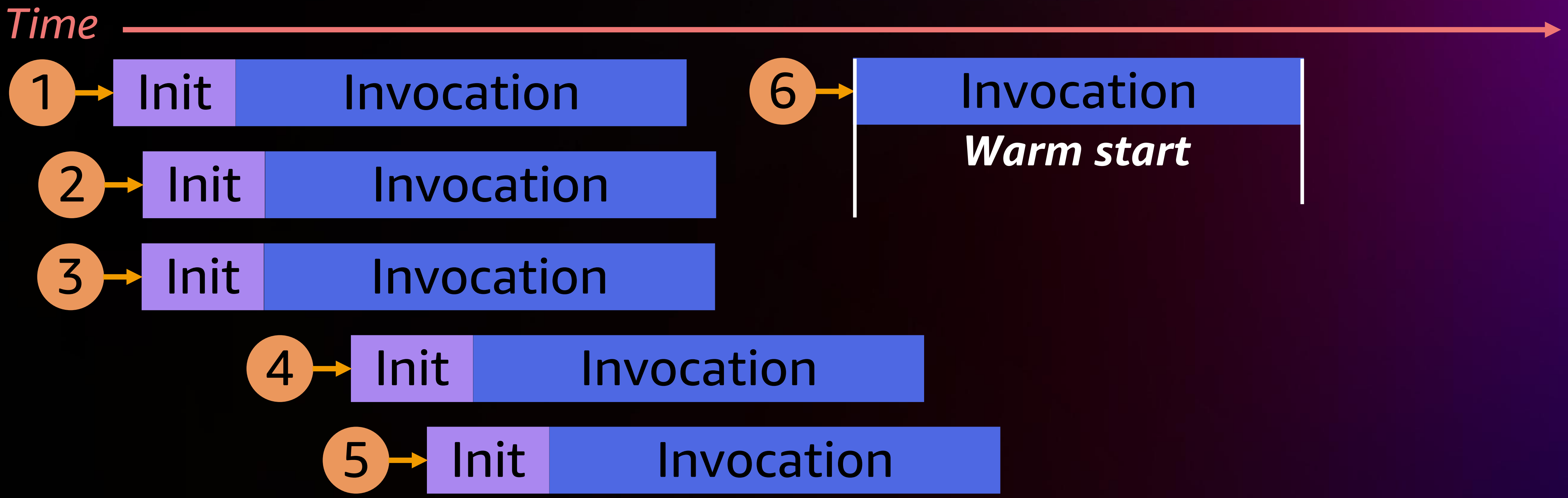


Understanding concurrency

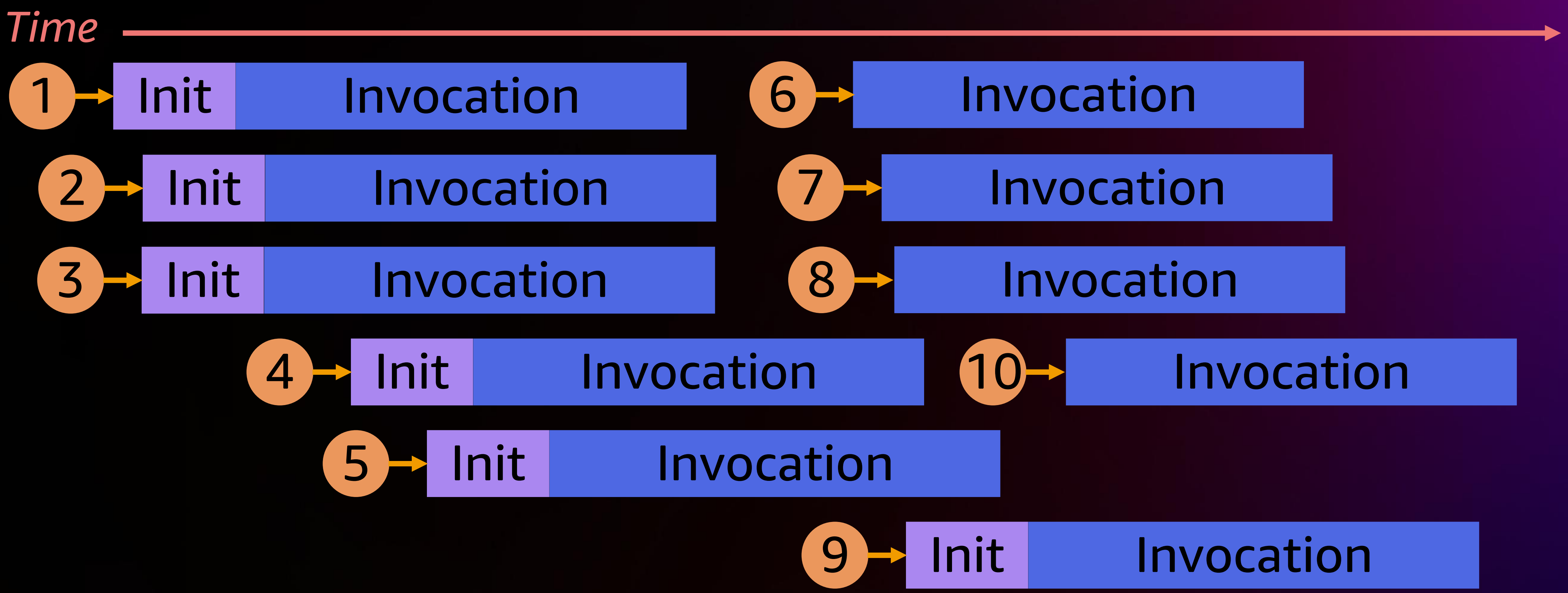


Execution environment available to process additional request

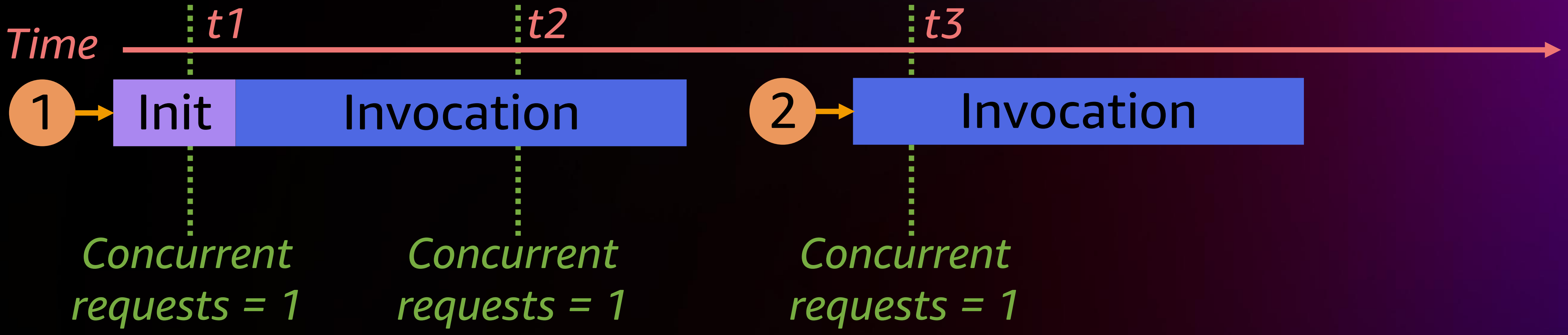
Understanding concurrency



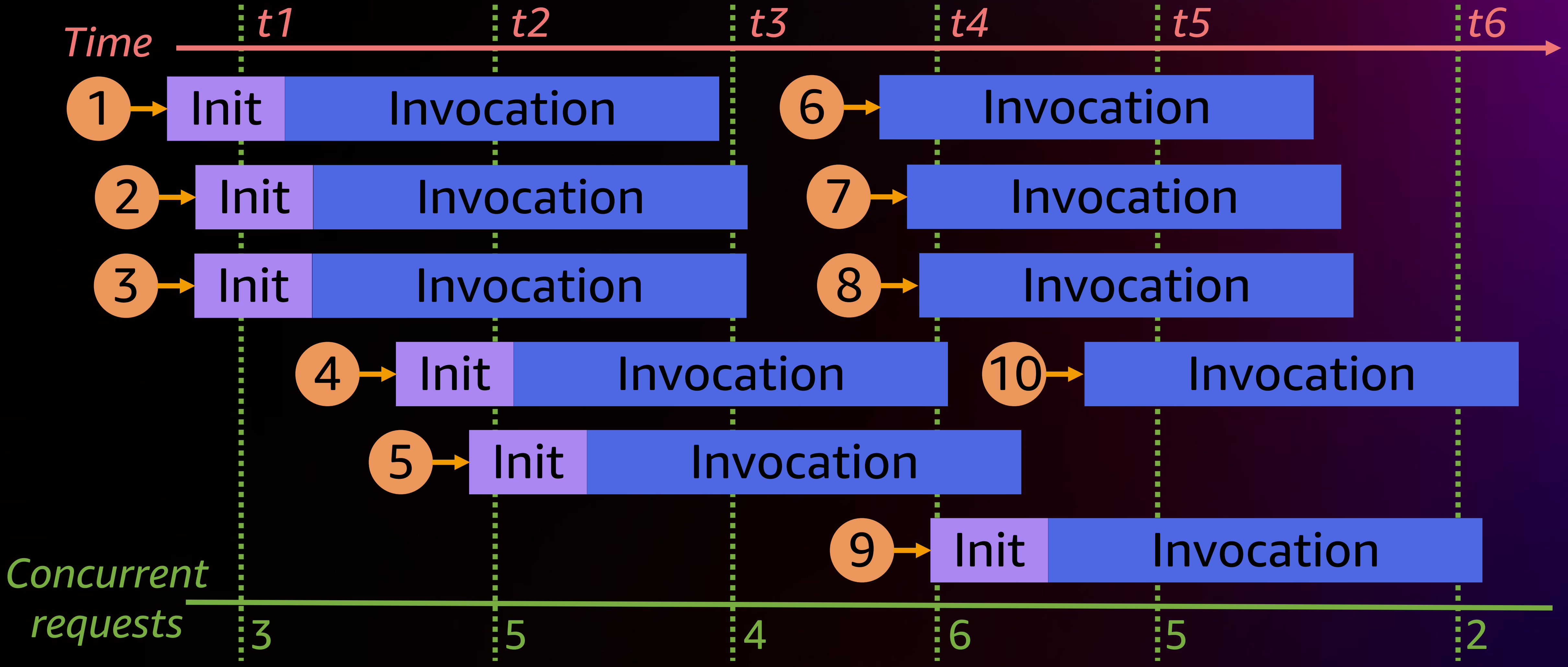
Understanding concurrency



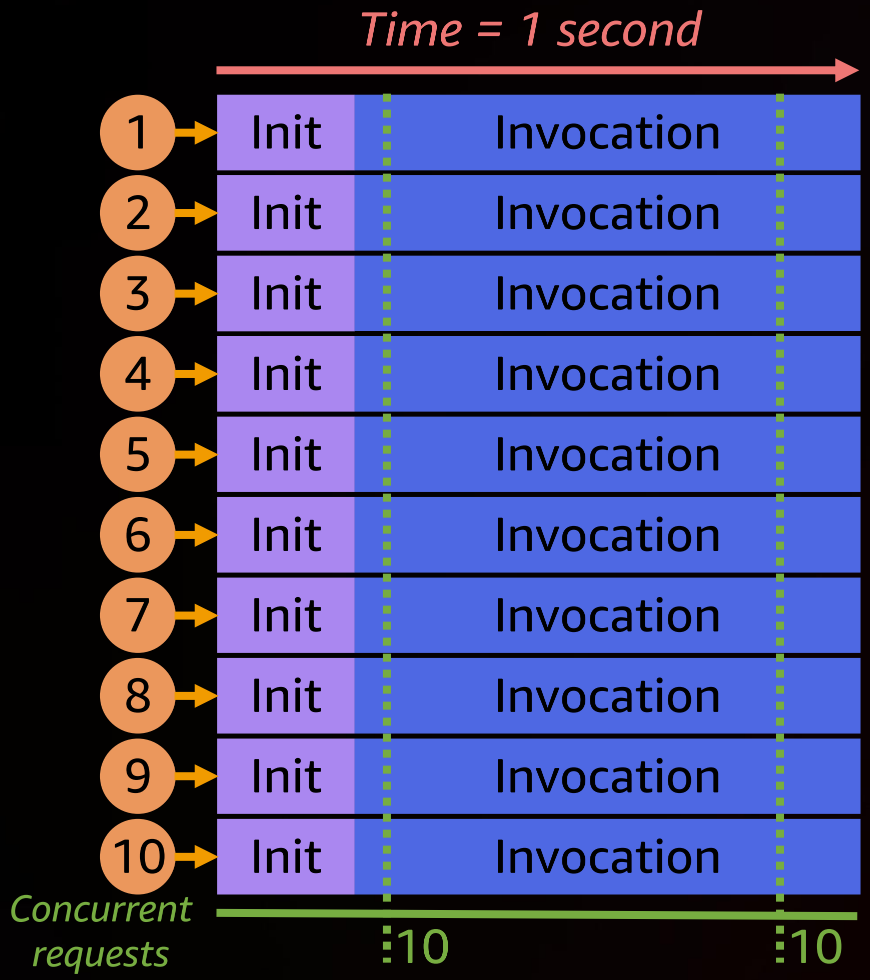
Understanding concurrency



Understanding concurrency

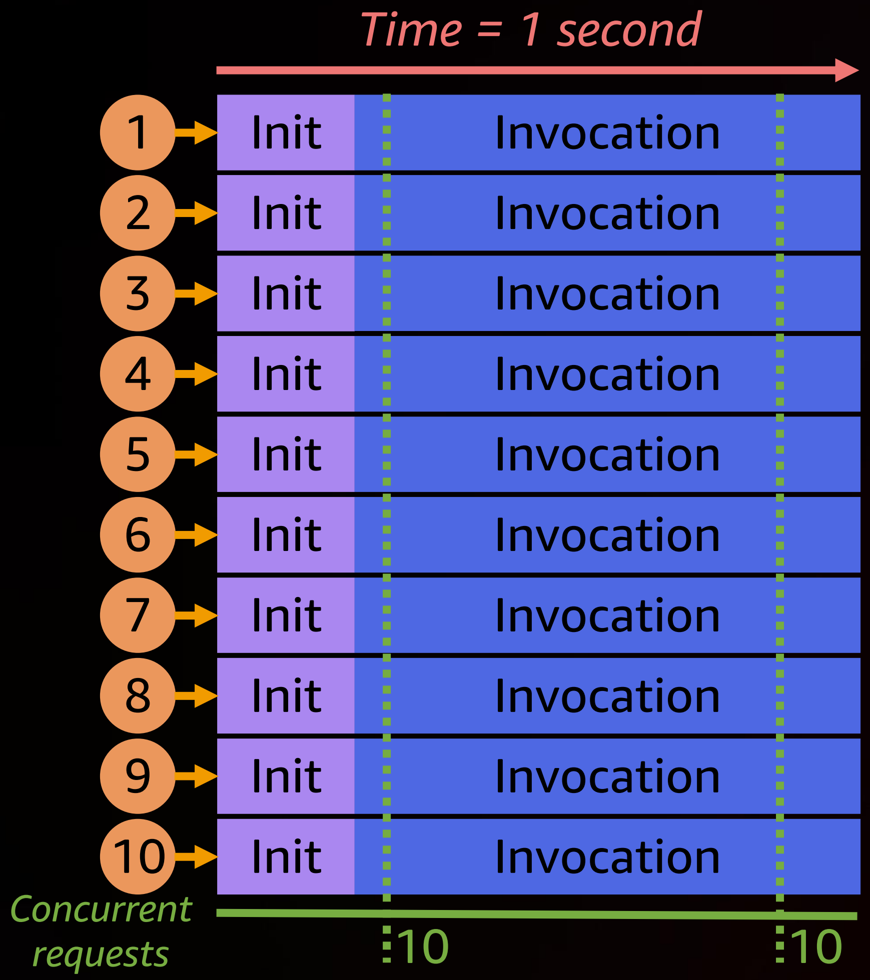


Understanding transactions per second

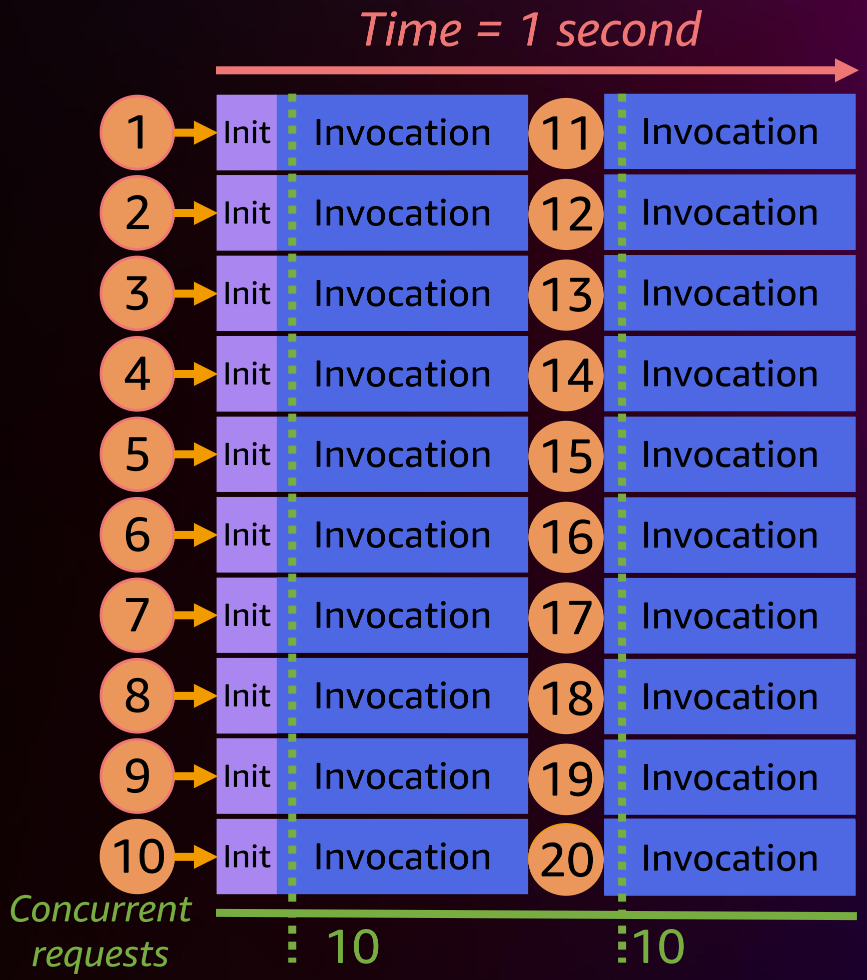


Transactions per second = 10

Understanding transactions per second

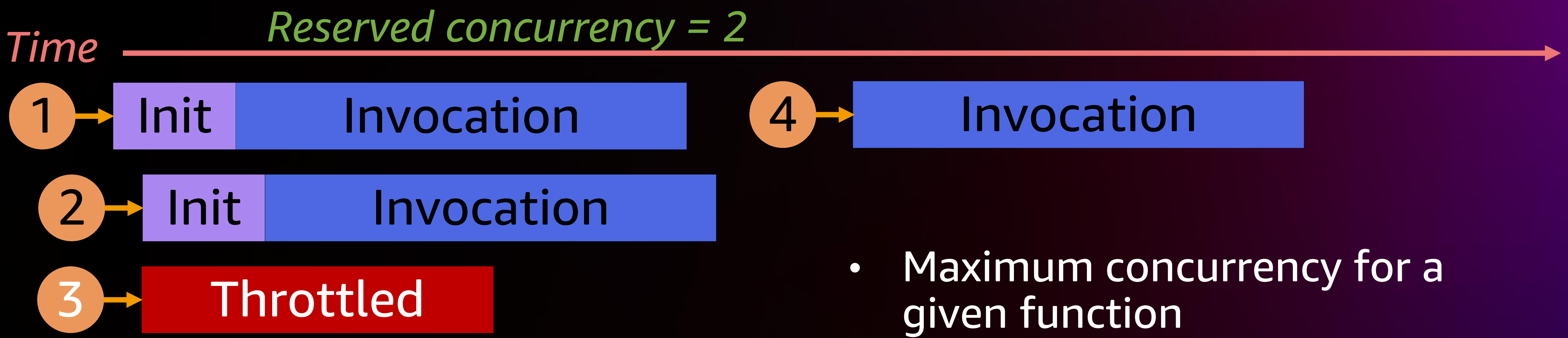


Transactions per second = 10



Transactions per second = **20**

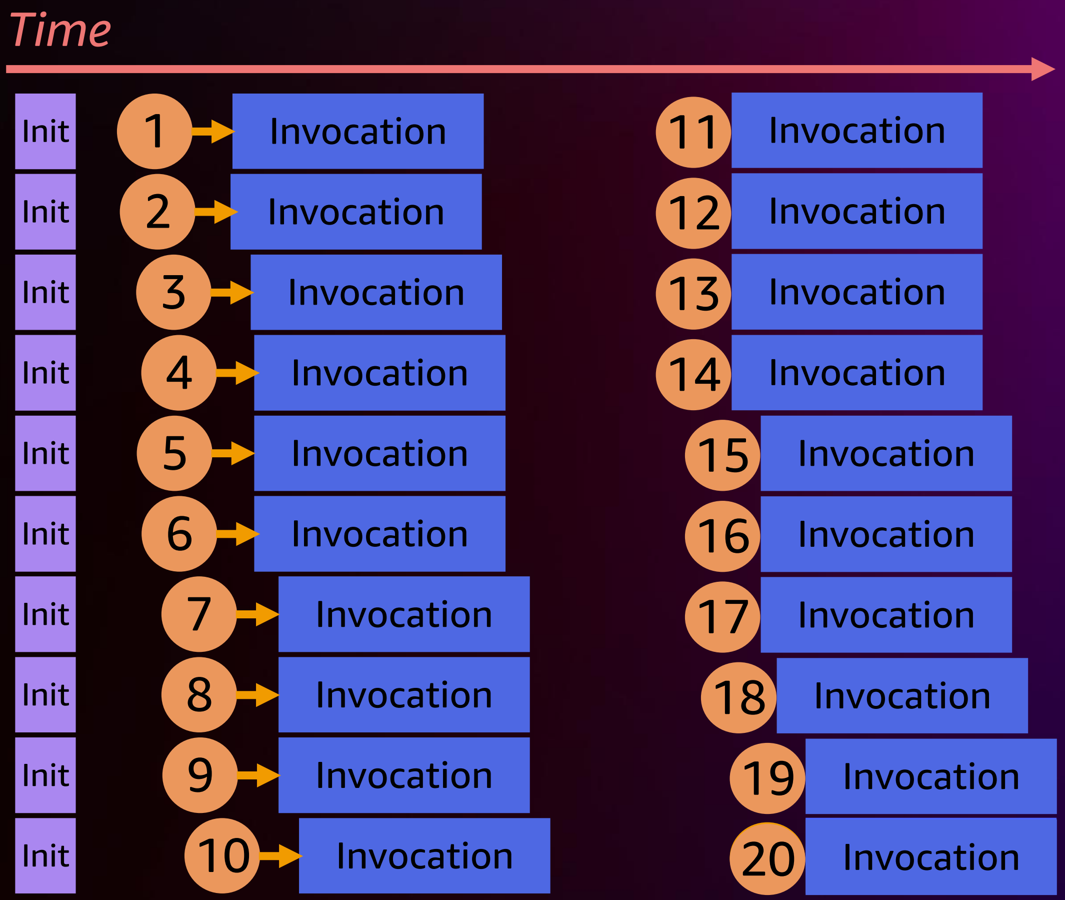
Reserved concurrency



- Maximum concurrency for a given function
- Also reserves that concurrency from the account’s quota
- Set function concurrency to protect downstream resources
- “Off switch” – set function concurrency to zero

Provisioned concurrency

- Sets minimum number of execution environments
- Pre-warm execution environments to reduce cold-start impact
- Burst to use standard concurrency if desired
- Can save costs in certain situations



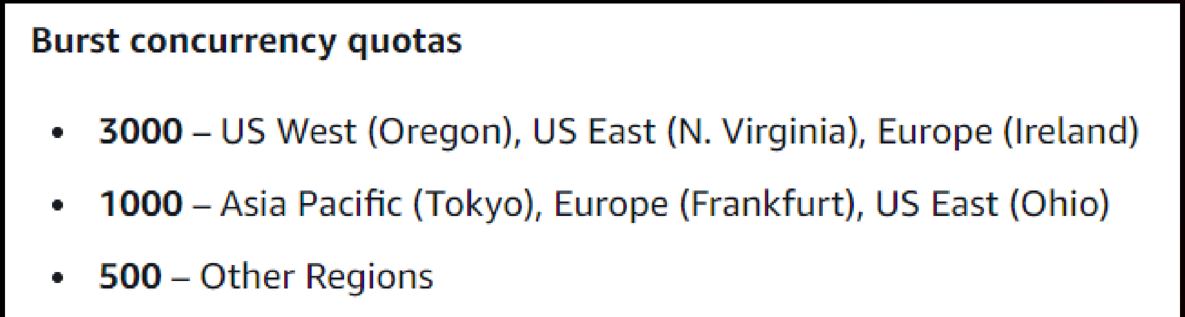
Provisioned concurrency = 10

Provisioned concurrency = 10

AWS Lambda function scaling quotas

Burst concurrency quota

Maximum increase in concurrency for an initial burst of traffic



Burst concurrency quotas

- **3000** – US West (Oregon), US East (N. Virginia), Europe (Ireland)
- **1000** – Asia Pacific (Tokyo), Europe (Frankfurt), US East (Ohio)
- **500** – Other Regions

After the initial burst, your function concurrency can scale by an additional 500 instances each minute

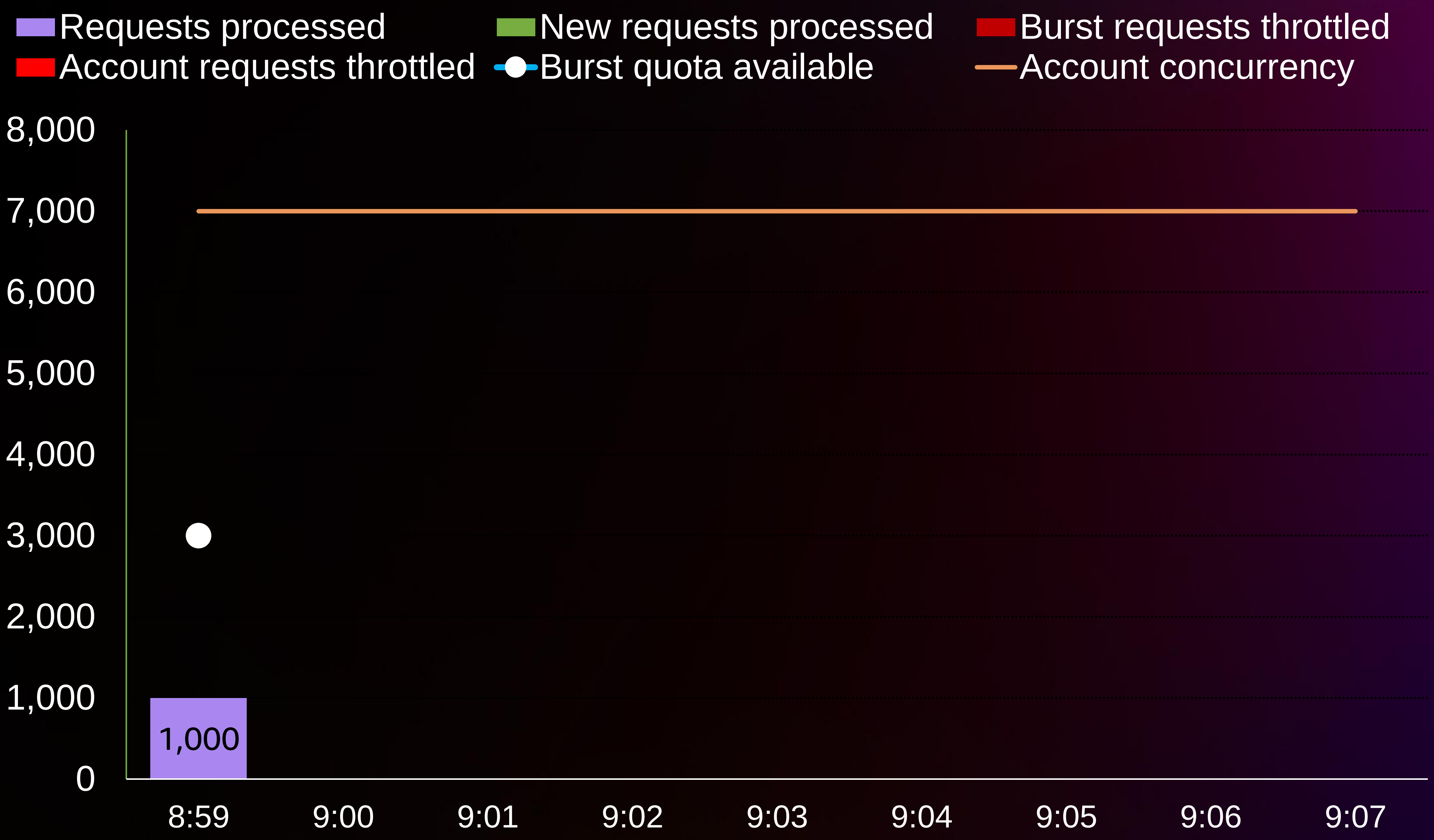
Account concurrency quota

Maximum concurrency in a given Region across all functions in an account

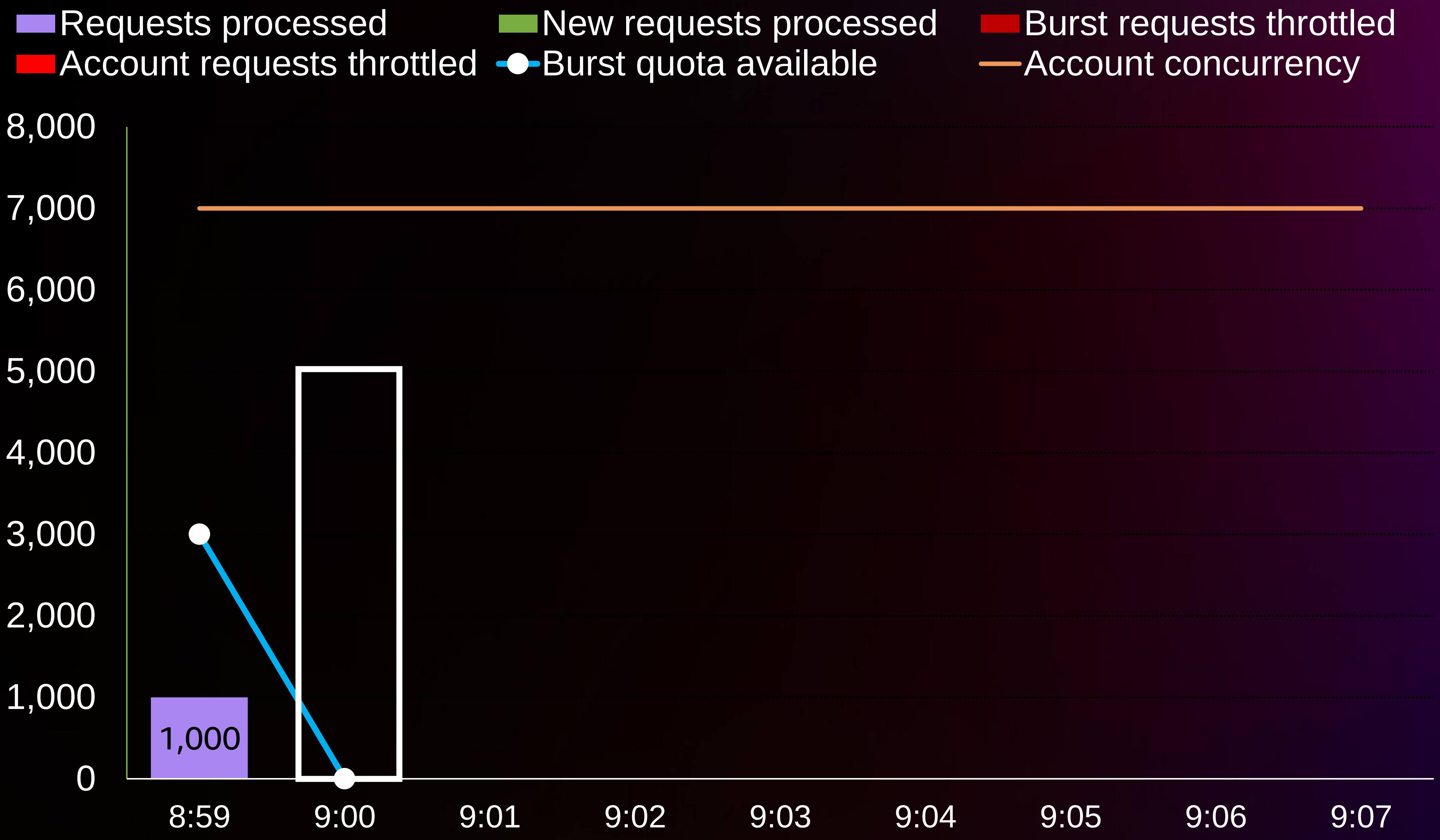
(default = 1,000 per Region)

This can be increased

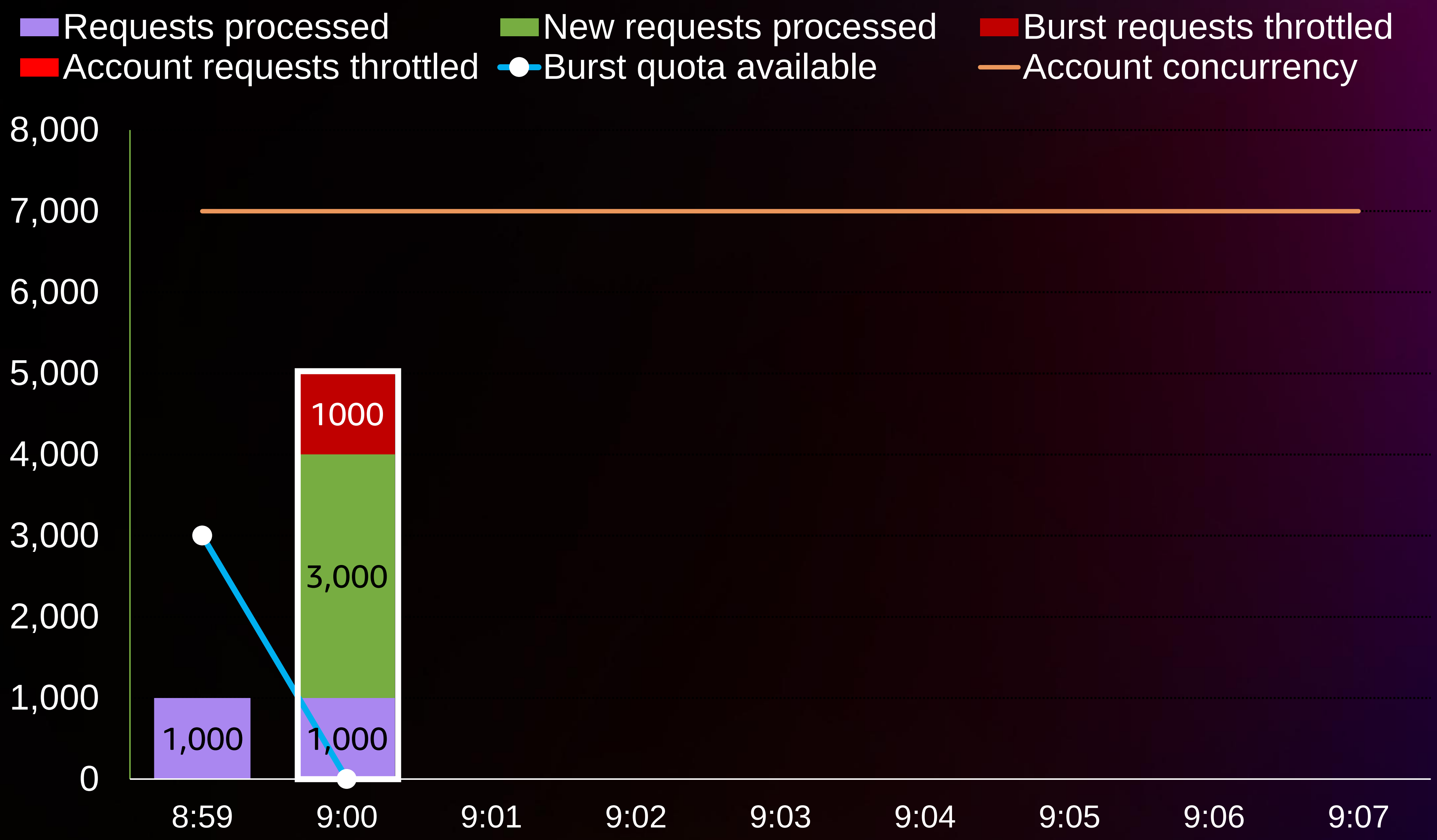
Scaling quotas



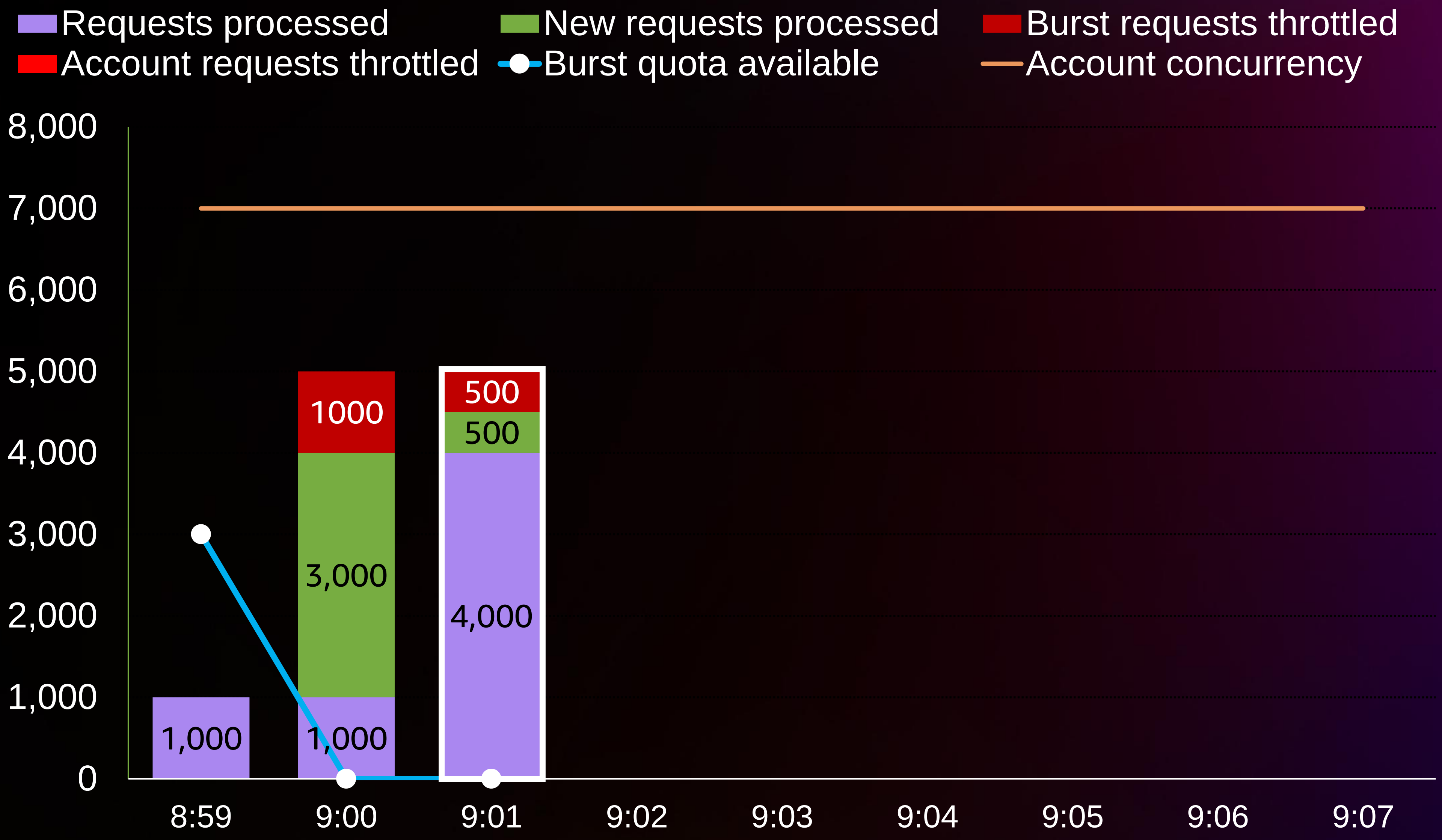
Scaling quotas



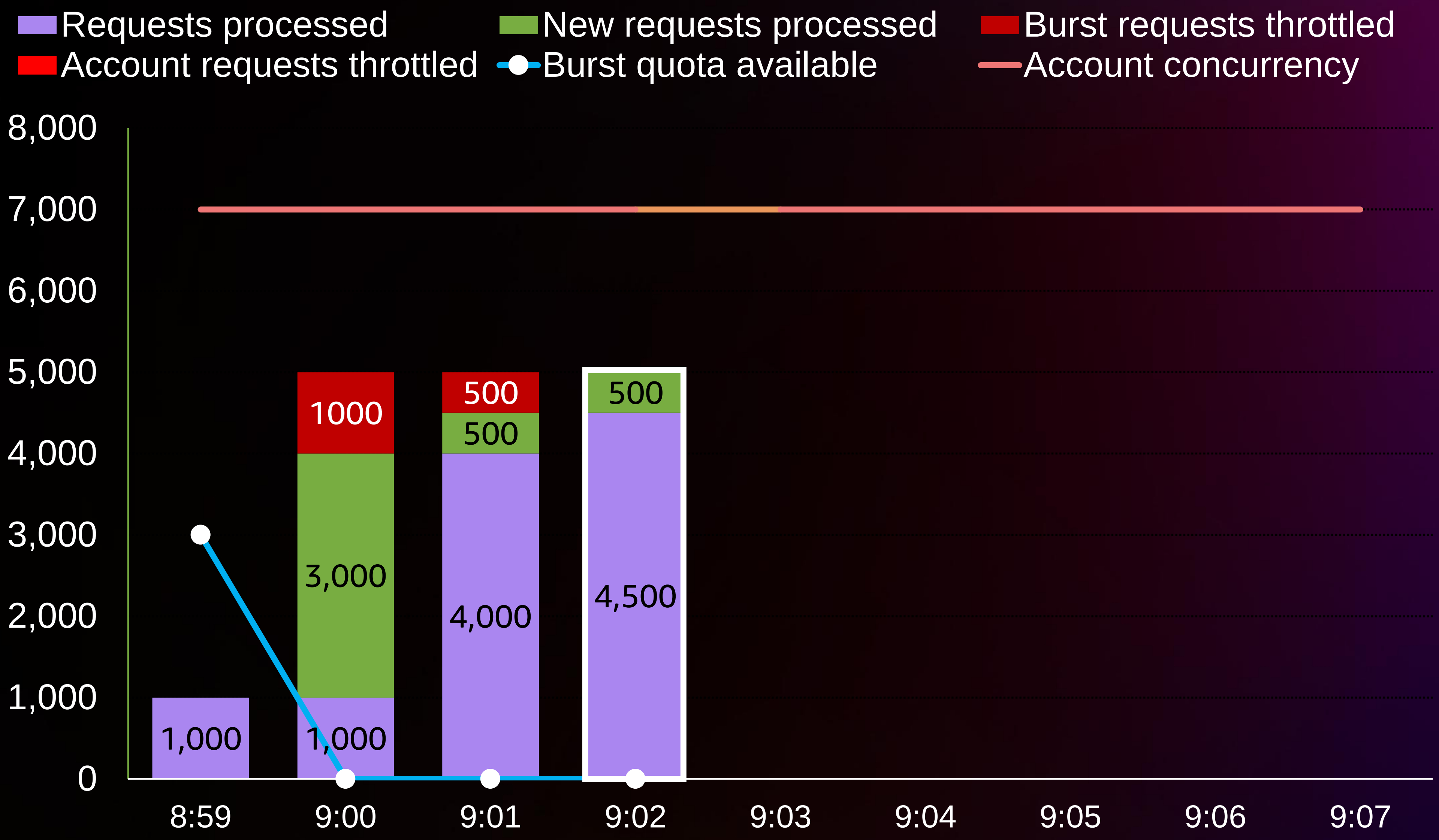
Scaling quotas



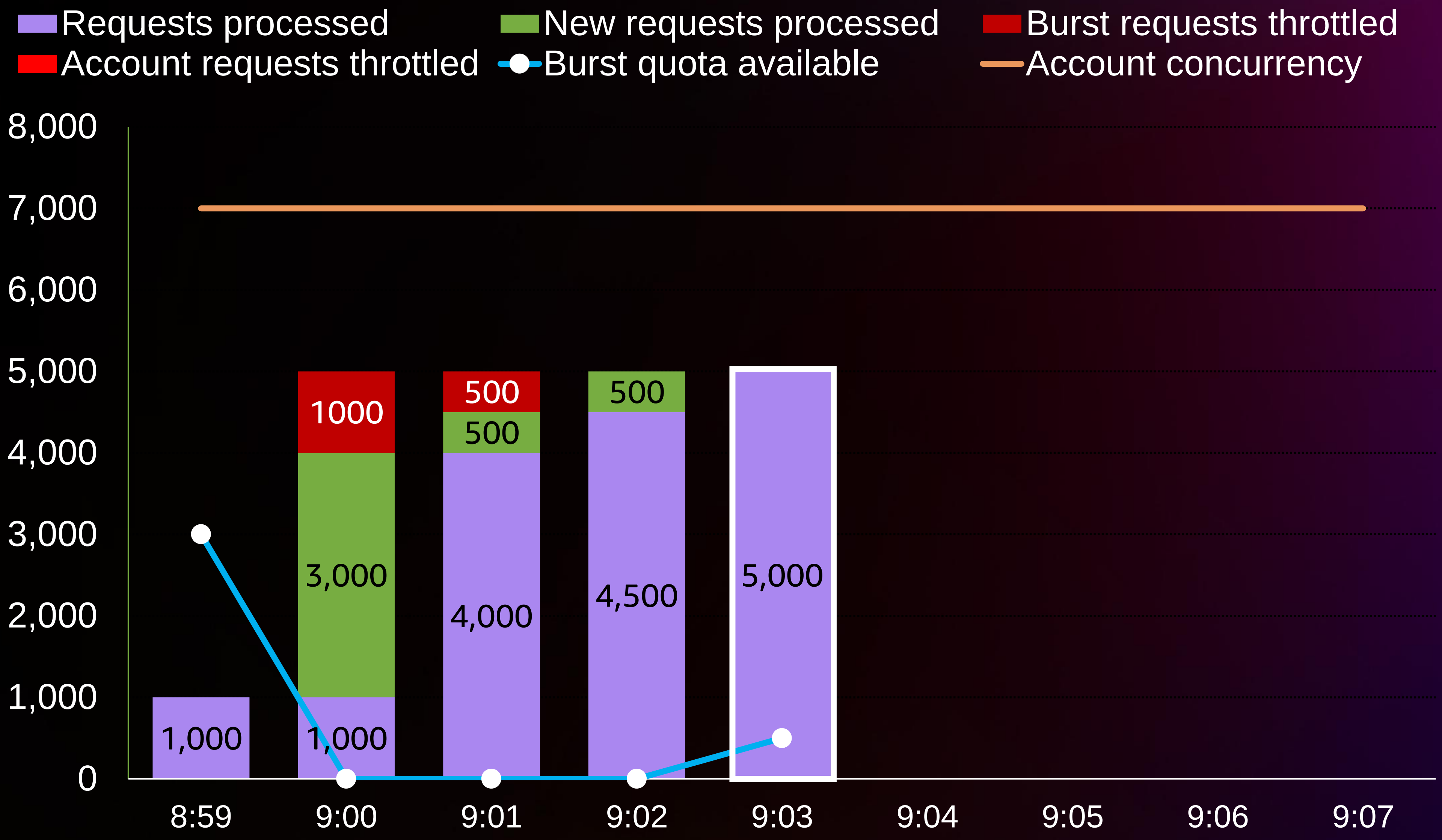
Scaling quotas



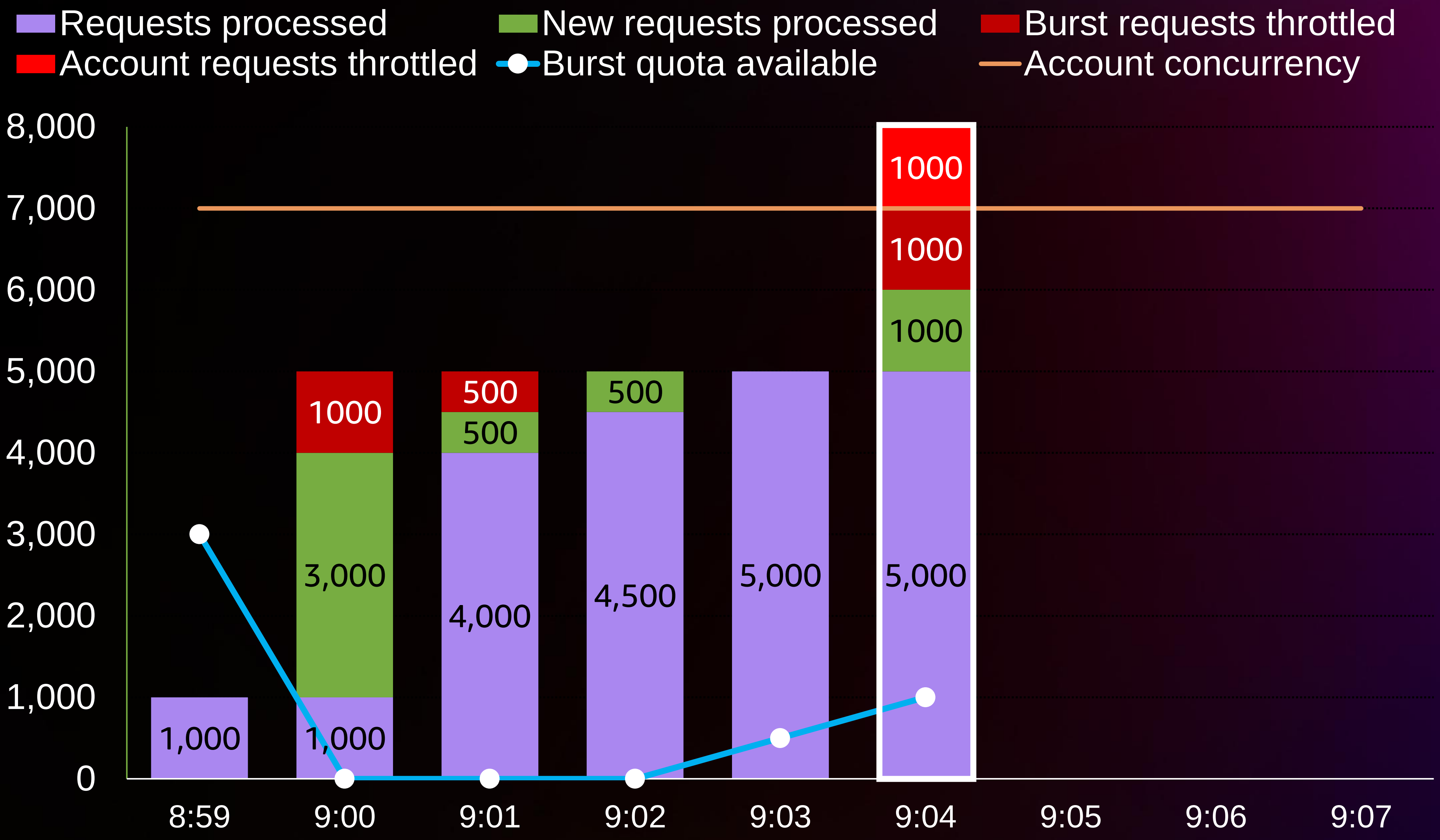
Scaling quotas



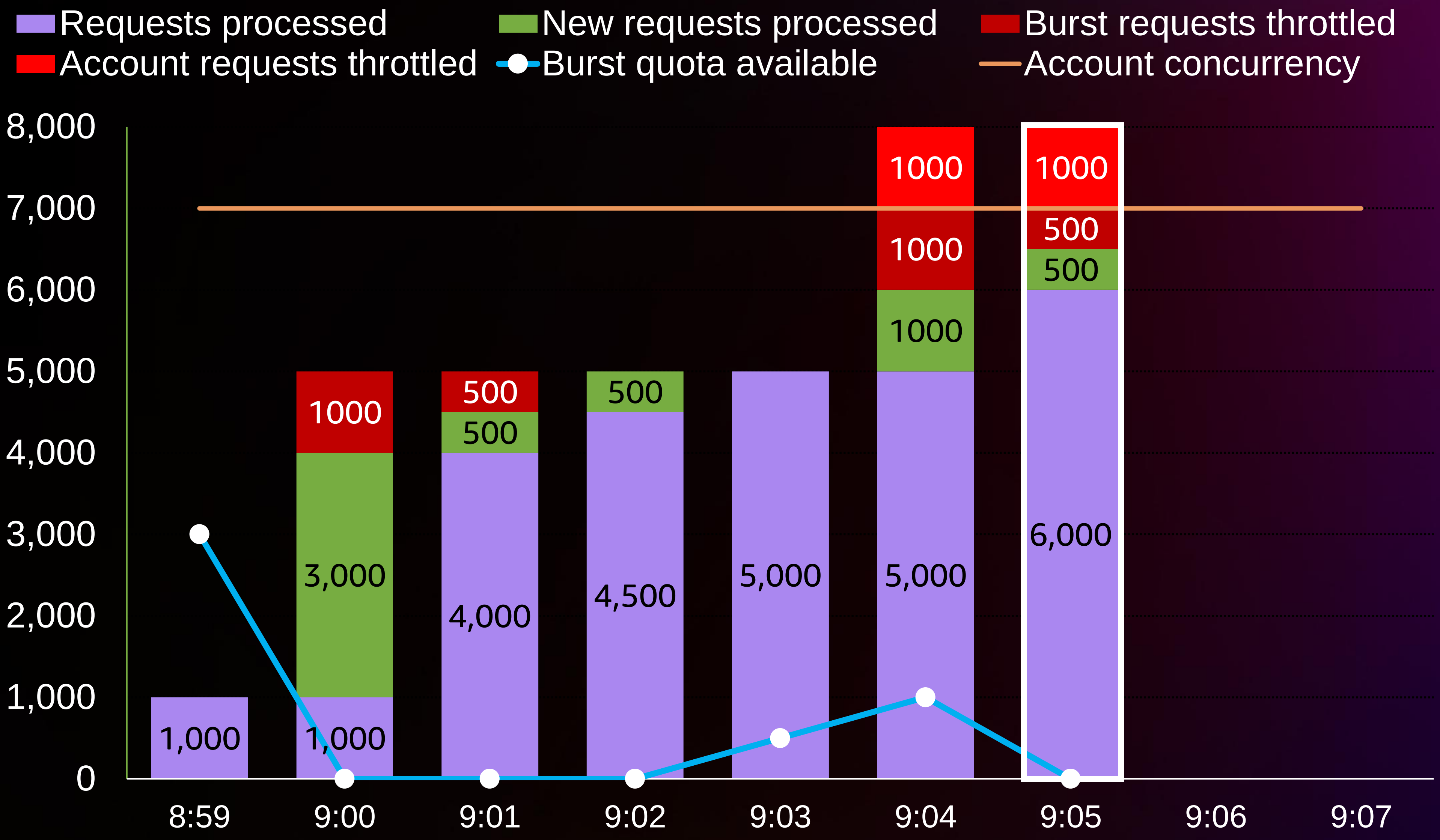
Scaling quotas



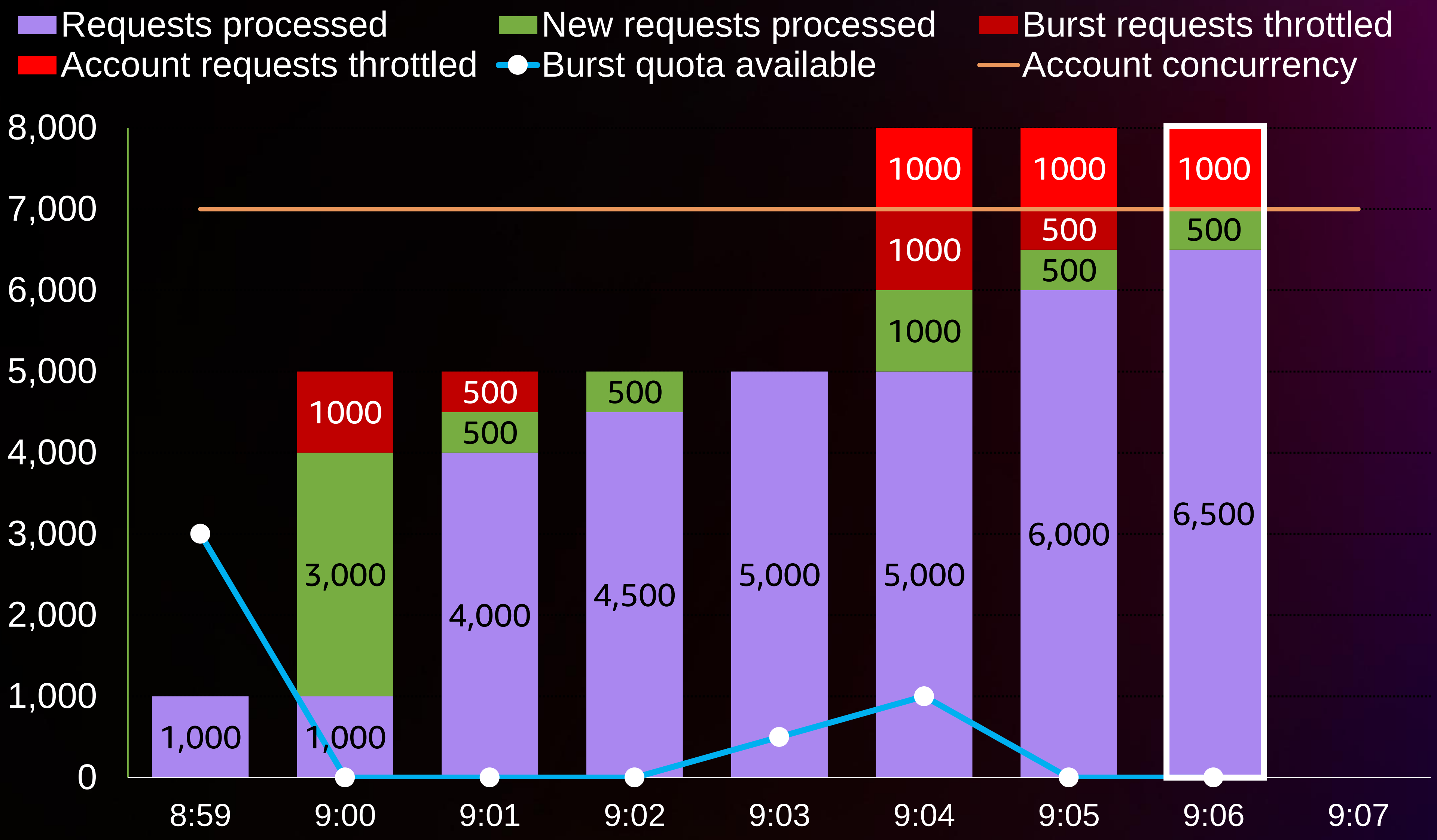
Scaling quotas



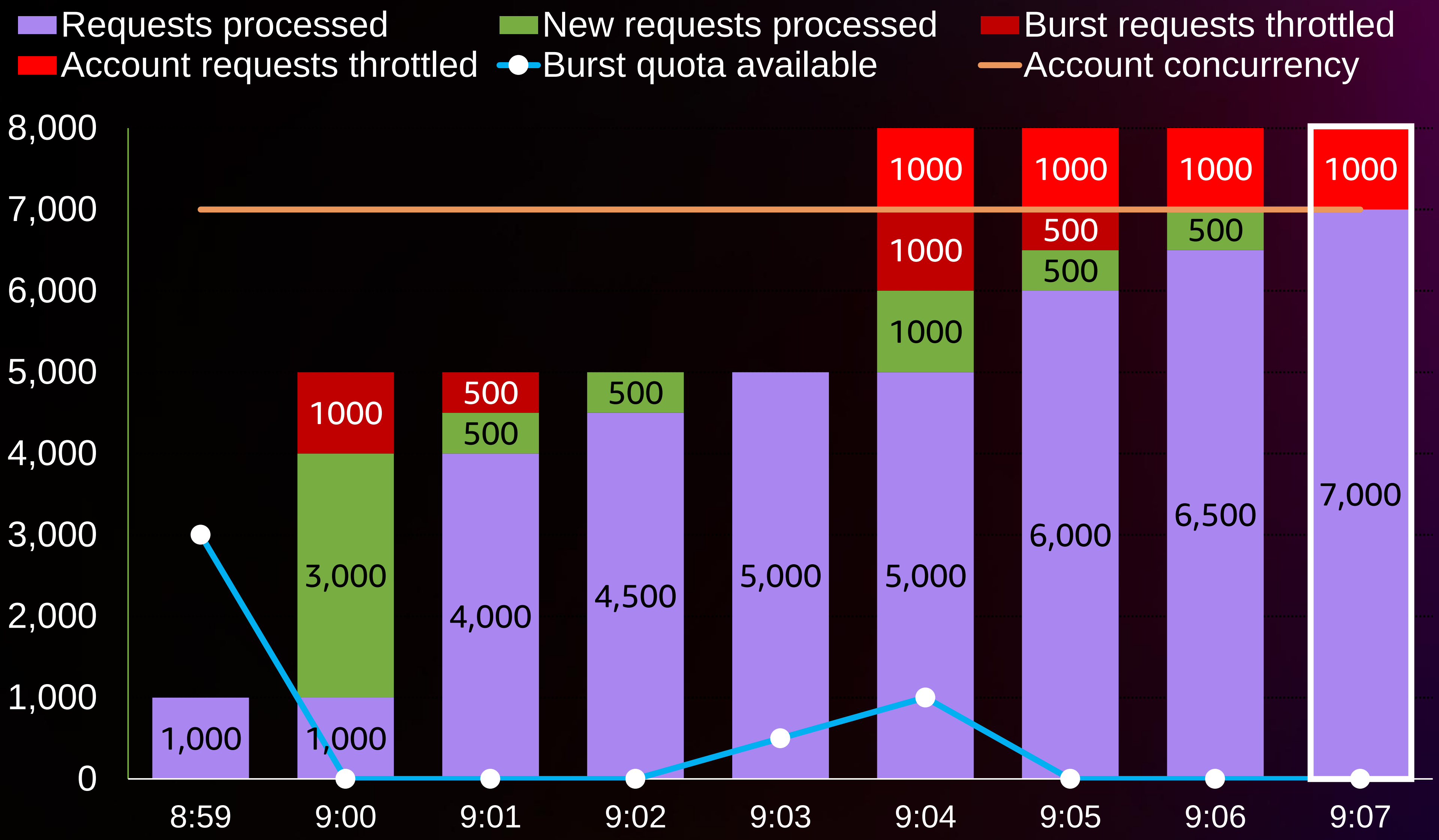
Scaling quotas



Scaling quotas



Scaling quotas



Fabulous functions: Best practices

- Use different invocation models
- Optimize cold starts
  - Don’t load it if you don’t need it, lazy initialize shared libraries
  - Establishing connections
  - State during environment reuse
- Try out ARM/AWS Graviton2
- More memory = proportionally more CPU and I/O
- Connect functions to a VPC only when you need to
- Understand concurrency and quotas, how to reserve and provision

Configuration as code

Infrastructure as code (IaC)

Build infrastructure using configuration files

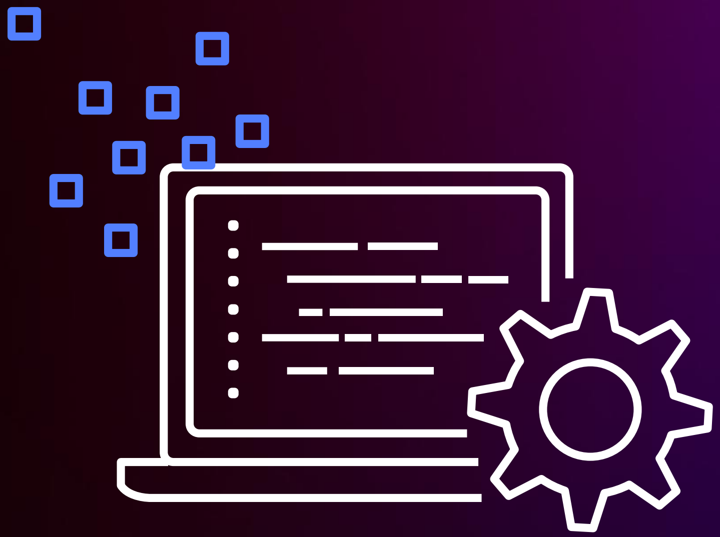
Treat configuration files as software code

With serverless, your infrastructure is your application

Automate the provisioning process

Reduce configuration drift

Deploy to multiple environments and accounts



IaC frameworks for serverless

AWS Serverless Application Model (AWS SAM)

AWS Cloud Development Kit (AWS CDK)

serverless

Architect

CHALICE

Pulumi

Terraform

Serverless Application Model

AWS SAM comes in two parts

AWS SAM transform

Shorthand syntax to express resources and event source mappings, it provides infrastructure as code (IaC) for serverless applications

AWS SAM CLI

Provides tooling for local and cloud development, debugging, build, packaging, pipeline creation, and deployment for serverless applications

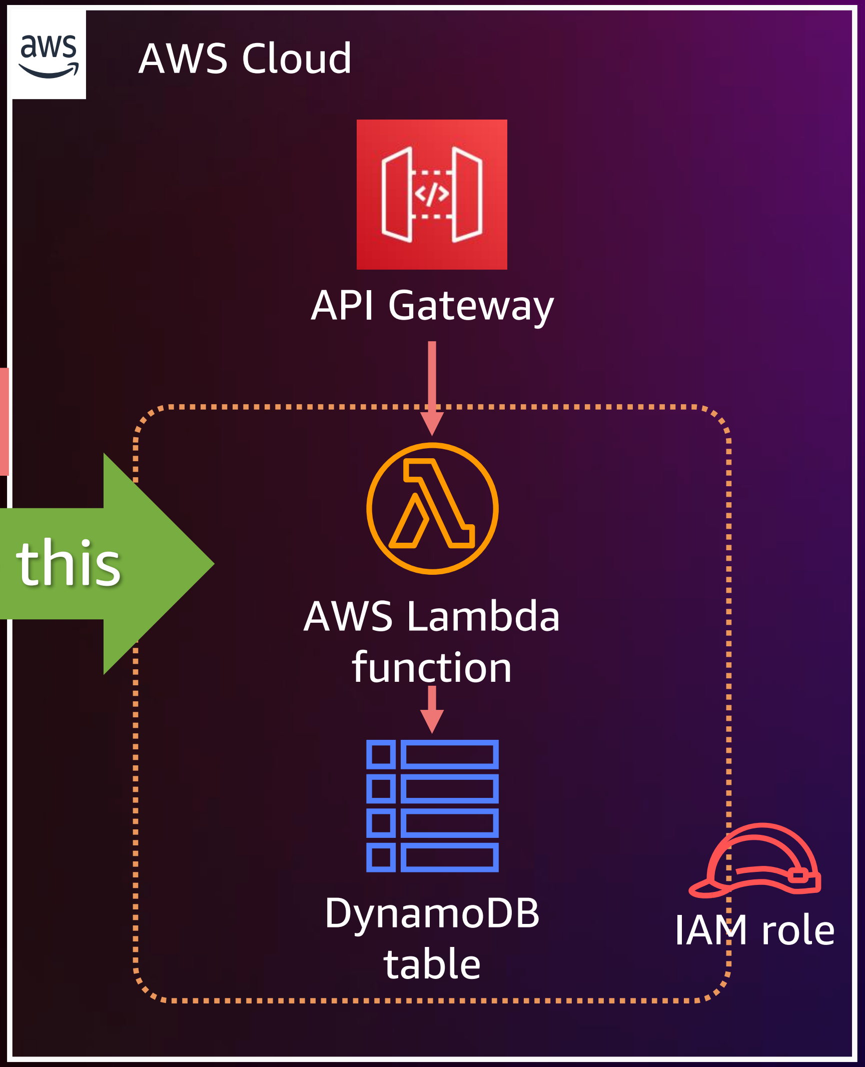
AWS SAM templates

Just 20 lines to create

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: src/
      Handler: app.handler
      Runtime: python3.9
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetProductsEvent:
          Type: Api
          Properties:
            Path: /products
            Method: get
    ProductTable:
      Type: AWS::Serverless::SimpleTable
```

Allowing this

To become this



Friends don’t let friends

"Action": "*"
"Action": "s3:*"
"Action": "dynamodb:*"
“Resource": "*"

Friends don’t let friends

```
"Action": "*"
"Action": "s3:*"
"Action": "dynamodb:*"
“Resource": "*"
```

IAM Access Analyzer

- Validate policies
- Check overly permissive policies

AWS SAM policy templates

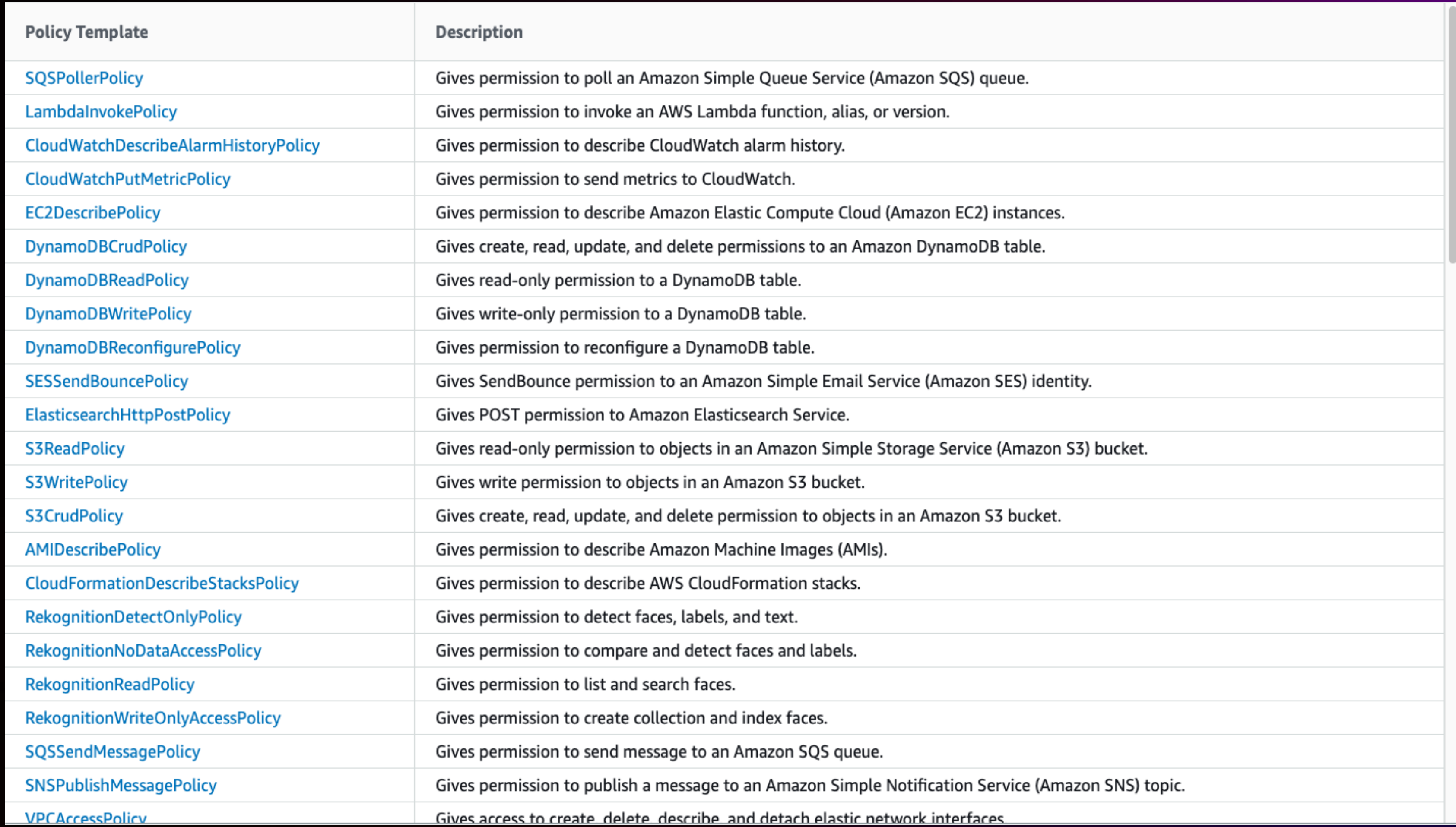
```
GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      ...
      Policies:
      - DynamoDBReadPolicy:
      TableName: !Ref ProductTable
      ...
ProductTable:
    Type: AWS::Serverless::SimpleTable
```

AWS SAM IAM policy templates

75+

Managed templates with more being added

Start here:
s12d.com/sam-policies



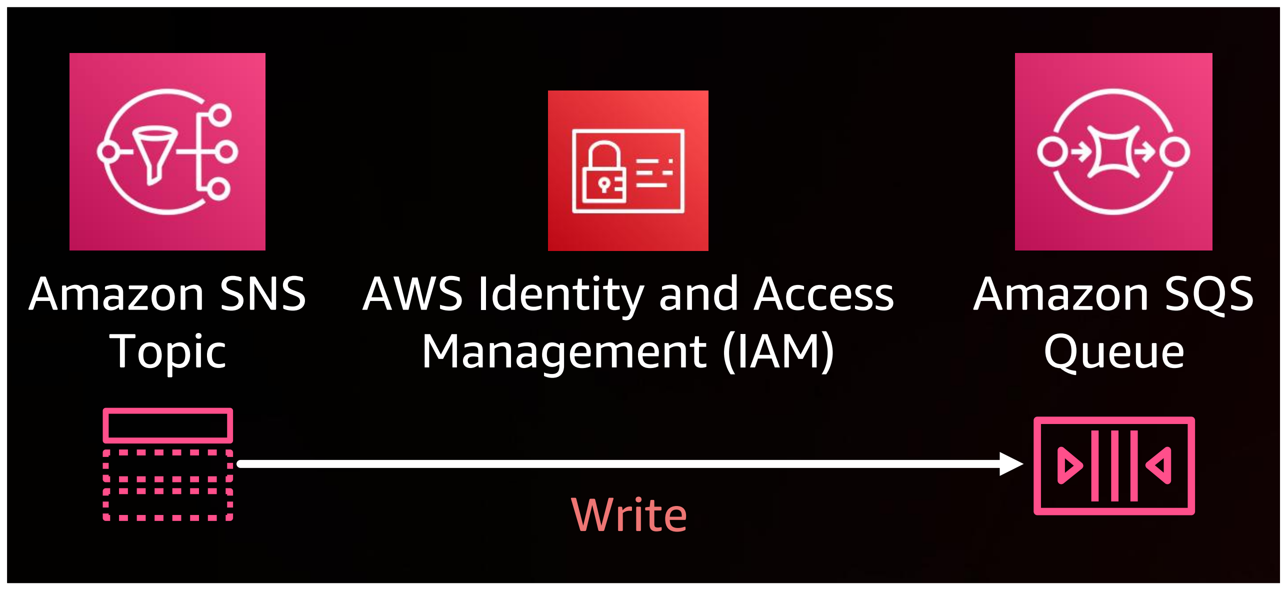
Policy Template	Description
SQSPollerPolicy	Gives permission to poll an Amazon Simple Queue Service (Amazon SQS) queue.
LambdaInvokePolicy	Gives permission to invoke an AWS Lambda function, alias, or version.
CloudWatchDescribeAlarmHistoryPolicy	Gives permission to describe CloudWatch alarm history.
CloudWatchPutMetricPolicy	Gives permission to send metrics to CloudWatch.
EC2DescribePolicy	Gives permission to describe Amazon Elastic Compute Cloud (Amazon EC2) instances.
DynamoDBCrudPolicy	Gives create, read, update, and delete permissions to an Amazon DynamoDB table.
DynamoDBReadPolicy	Gives read-only permission to a DynamoDB table.
DynamoDBWritePolicy	Gives write-only permission to a DynamoDB table.
DynamoDBReconfigurePolicy	Gives permission to reconfigure a DynamoDB table.
SESSendBouncePolicy	Gives SendBounce permission to an Amazon Simple Email Service (Amazon SES) identity.
ElasticsearchHttpPostPolicy	Gives POST permission to Amazon Elasticsearch Service.
S3ReadPolicy	Gives read-only permission to objects in an Amazon Simple Storage Service (Amazon S3) bucket.
S3WritePolicy	Gives write permission to objects in an Amazon S3 bucket.
S3CrudPolicy	Gives create, read, update, and delete permission to objects in an Amazon S3 bucket.
AMIDescribePolicy	Gives permission to describe Amazon Machine Images (AMIs).
CloudFormationDescribeStacksPolicy	Gives permission to describe AWS CloudFormation stacks.
RekognitionDetectOnlyPolicy	Gives permission to detect faces, labels, and text.
RekognitionNoDataAccessPolicy	Gives permission to compare and detect faces and labels.
RekognitionReadPolicy	Gives permission to list and search faces.
RekognitionWriteOnlyAccessPolicy	Gives permission to create collection and index faces.
SQSSendMessagePolicy	Gives permission to send message to an Amazon SQS queue.
SNSPublishMessagePolicy	Gives permission to publish a message to an Amazon Simple Notification Service (Amazon SNS) topic.
VPCAccessPolicy	Gives access to create, delete, describe, and detach elastic network interfaces.

AWS SAM serverless connectors

`AWS::Serverless::Connector`

Resource to describe how data and events flow between two resources and the level of permissions required

40+ supported sources and destinations



```
MyTableConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: AppSNSTopic
    Destination:
      Id: AppSQSQueue
    Permissions:
      - Write
```

s12d.com/serverless-connectors

AWS SAM extras

Don’t hard code names

Helps to avoid naming clashes

Use `sam delete` for cleanup

- Removes associated companion stacks
- Removes artifact ZIP files
- Removes ECR repository images and repos
- Removes the stack
- Prompts before deleting anything

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      ~~FunctionName: MyFunction~~
      CodeUri: hello-world/
      Handler: app.lambda_handler
      Runtime: python3.8
      Events:
        HelloWorld
          Type: Api
          Properties:
            Path: /hello
            Method: get
```

AWS SAM local support for HashiCorp Terraform

Preview

```
resource "aws_lambda_function" "test_lambda" {
  filename = "lambda_function_payload.zip”
  function_name = "lambda_function_name”
  role = aws_iam_role.iam_for_lambda.arn
  handler = "index.test”
  source_code_hash = filebase64sha256("payload.zip")
  runtime = "nodejs16.x”
  environment {
    variables = { foo = "bar" }
  }
}
```

Commands

```
sam local invoke
sam local start-lambda
```

The Complete AWS SAM Workshop

Learn many of the major AWS SAM features

2–4 hours self-paced

Module 1: Create a SAM app

Module 2: Run locally

Module 3: Deploy manually

Module 4: CI/CD

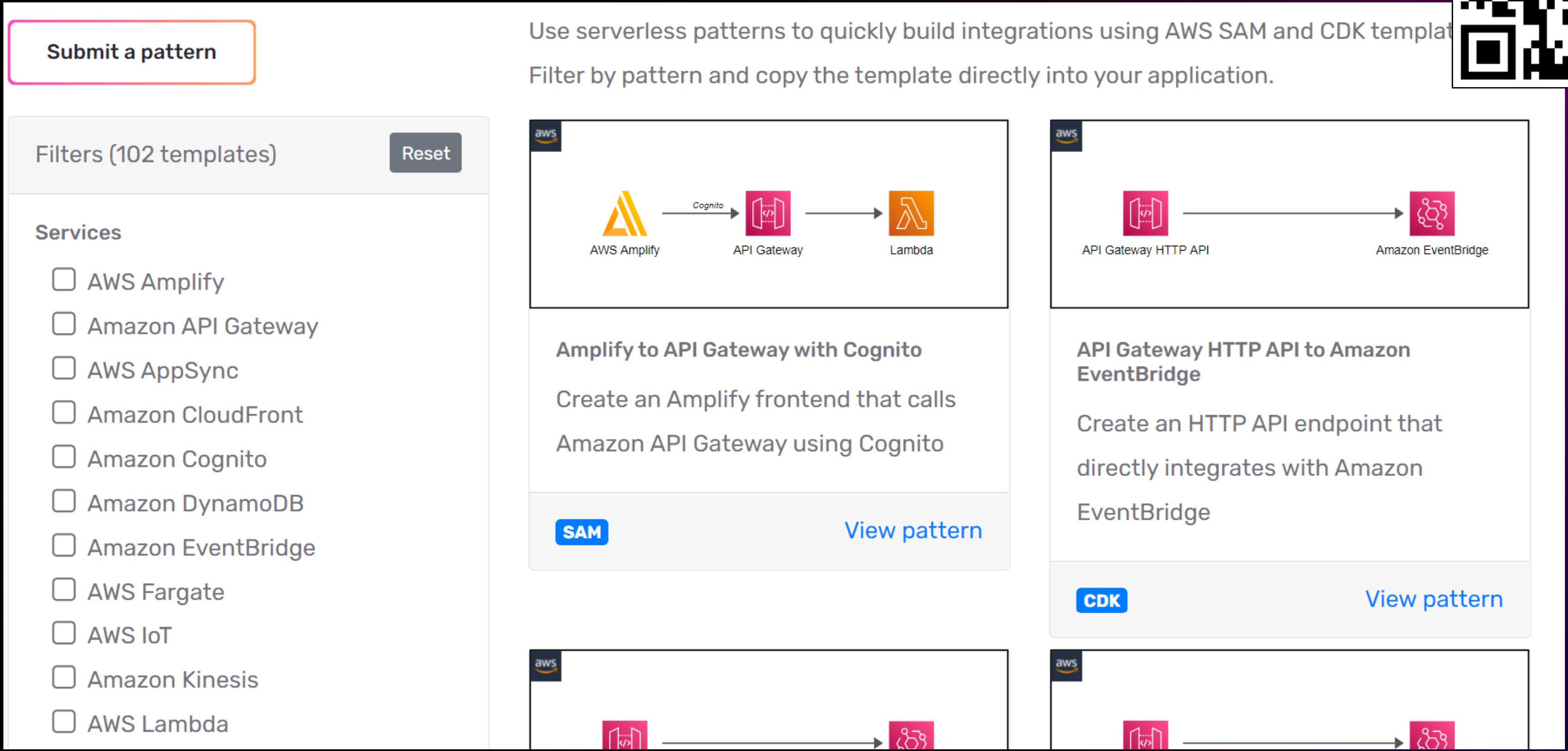
Module 5: Canary deployments

Module 6: SAM accelerate

s12d.com/sam-workshop

Serverless patterns collection

s12d.com/patterns



Submit a pattern

Use serverless patterns to quickly build integrations using AWS SAM and CDK templat

Filter by pattern and copy the template directly into your application.

Filters (102 templates) Reset

Services

- ☐ AWS Amplify
- ☐ Amazon API Gateway
- ☐ AWS AppSync
- ☐ Amazon CloudFront
- ☐ Amazon Cognito
- ☐ Amazon DynamoDB
- ☐ Amazon EventBridge
- ☐ AWS Fargate
- ☐ AWS IoT
- ☐ Amazon Kinesis
- ☐ AWS Lambda

Amplify to API Gateway with Cognito

Create an Amplify frontend that calls Amazon API Gateway using Cognito

SAM View pattern

API Gateway HTTP API to Amazon EventBridge

Create an HTTP API endpoint that directly integrates with Amazon EventBridge

CDK View pattern

Configuration as code: Best practices

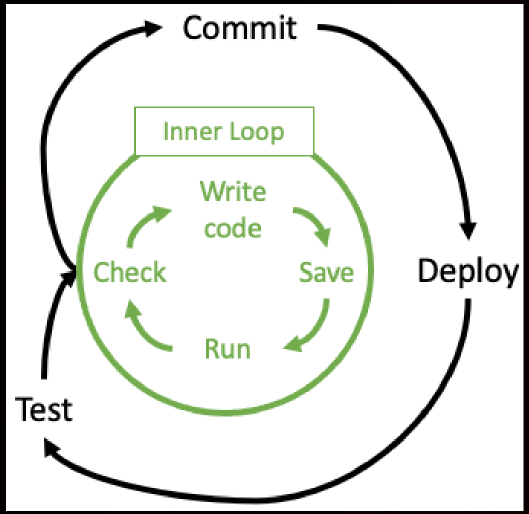
- Use a framework!
- Friends don’t let friends “Action”: “*”
- Use policy templates
- Discover the serverless patterns collection
- Split stacks: stateful/long-lived vs. more stateless resources

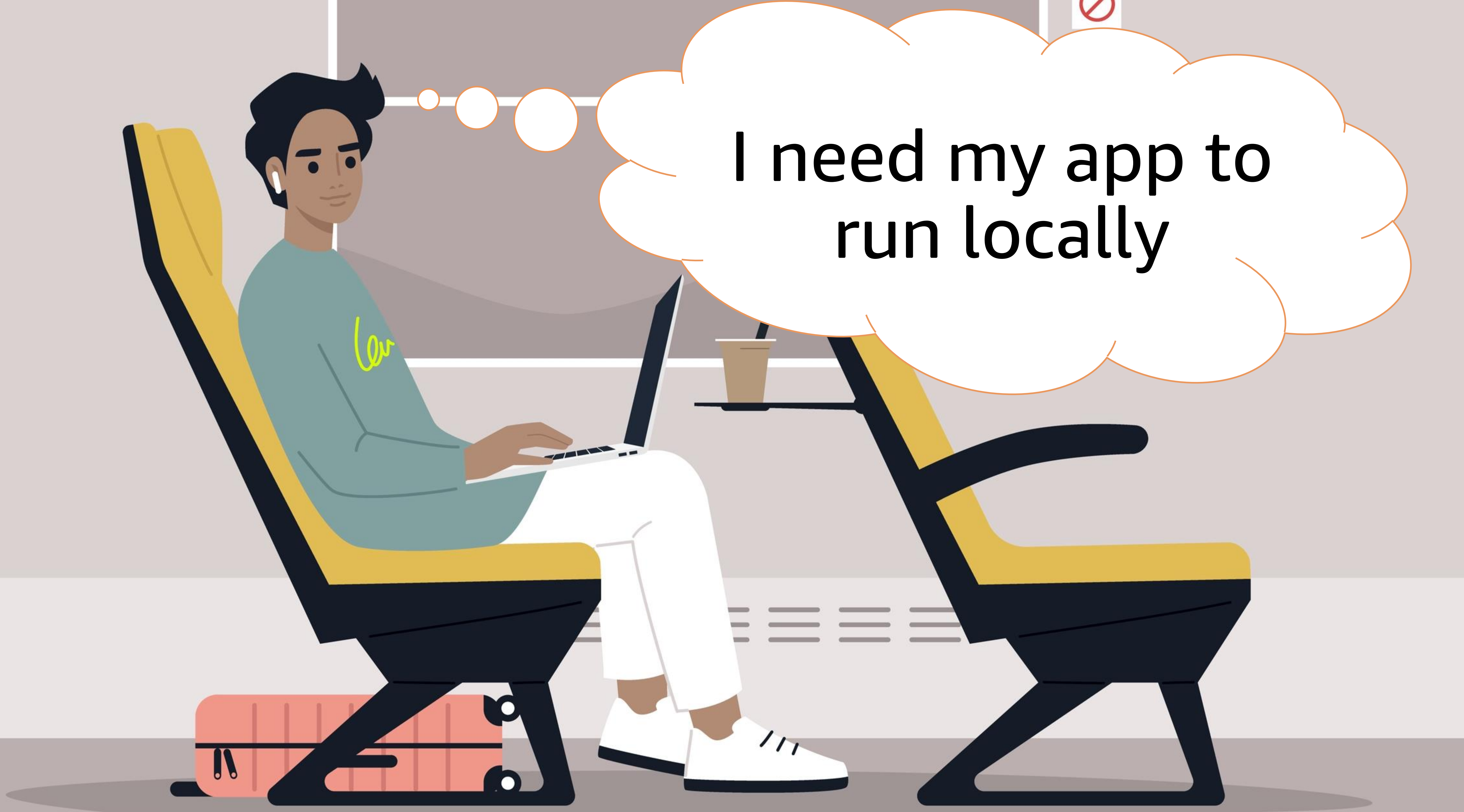
From prototype to production

What developers are used to…

Developers typically use the following workflow cycle before committing code to the main branch

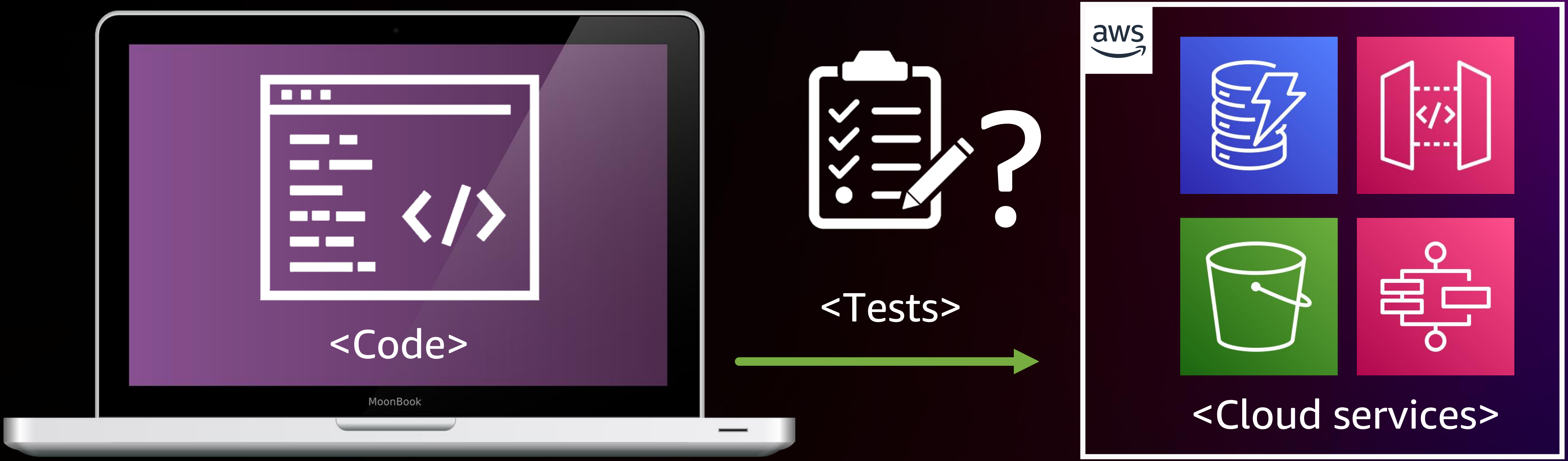
1. Write code
2. Save code
3. Run code
4. Check results





I need my app to run locally

Cloud-native applications are different



<Code>

<Tests>

<Cloud services>

I need my app to run locally

What’s actually more important, testing locally, or testing quickly?

Service emulation

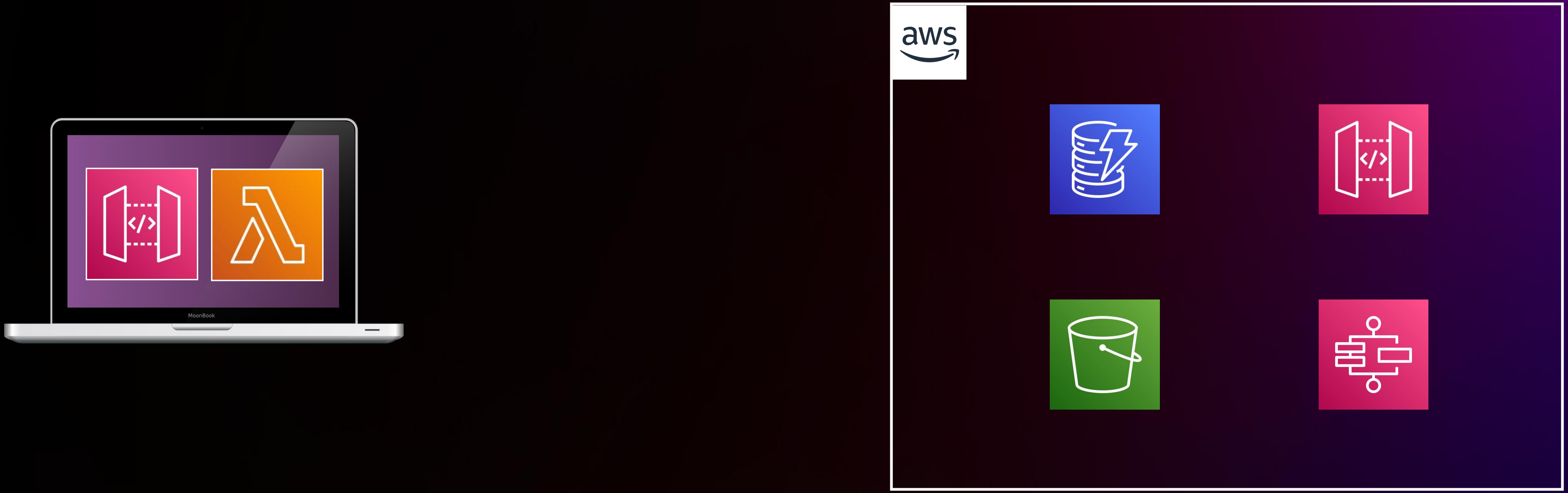
In serverless applications you configure the interfaces between services

Avoid emulating these services on your local machine

Use mock frameworks sparingly for unit testing complex internal business logic

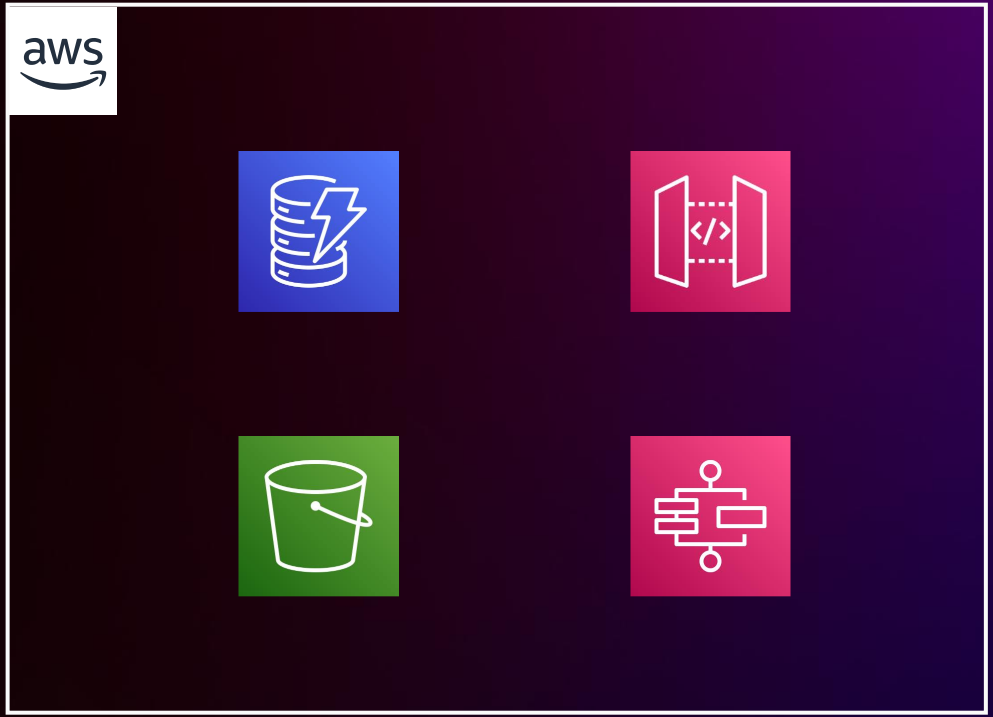
Image by Pixel_perfect from Pixabay

Minimal service emulation



Minimal service emulation

Run your code in an actual cloud environment as early as possible



- Local iteration of AWS Lambda function code
- Communicate with actual services
- Run integration tests in the cloud
- Increase infrastructure accuracy for testing

AWS SAM accelerate

s12d.com/accelerate

ITERATE AGAINST THE CLOUD WITH THE SPEED OF LOCAL DEVELOPMENT

`sam build`

Build only the parts of your code that have changed

- Parallel builds
- Incremental builds
- Temporary layers

`sam sync`

Quickly sync code/API/Workflow changes with the cloud

- Watch for changed files

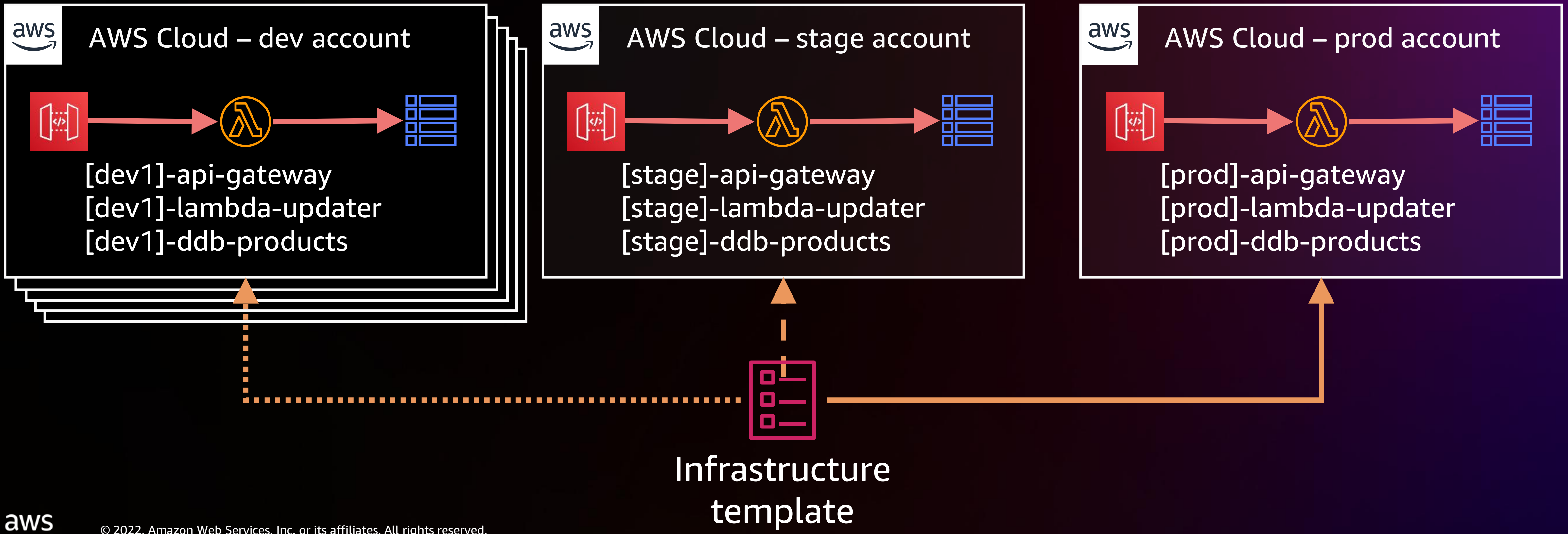
`sam logs`

Tail aggregated application logs in local terminal

- Filter logs
- Include traces

Pressing Ctrl+S in your IDE synchronizes your local stack to a cloud stack . . . in seconds

Reusable templates



AWS Cloud – dev account

[dev1]-api-gateway
[dev1]-lambda-updater
[dev1]-ddb-products

AWS Cloud – stage account

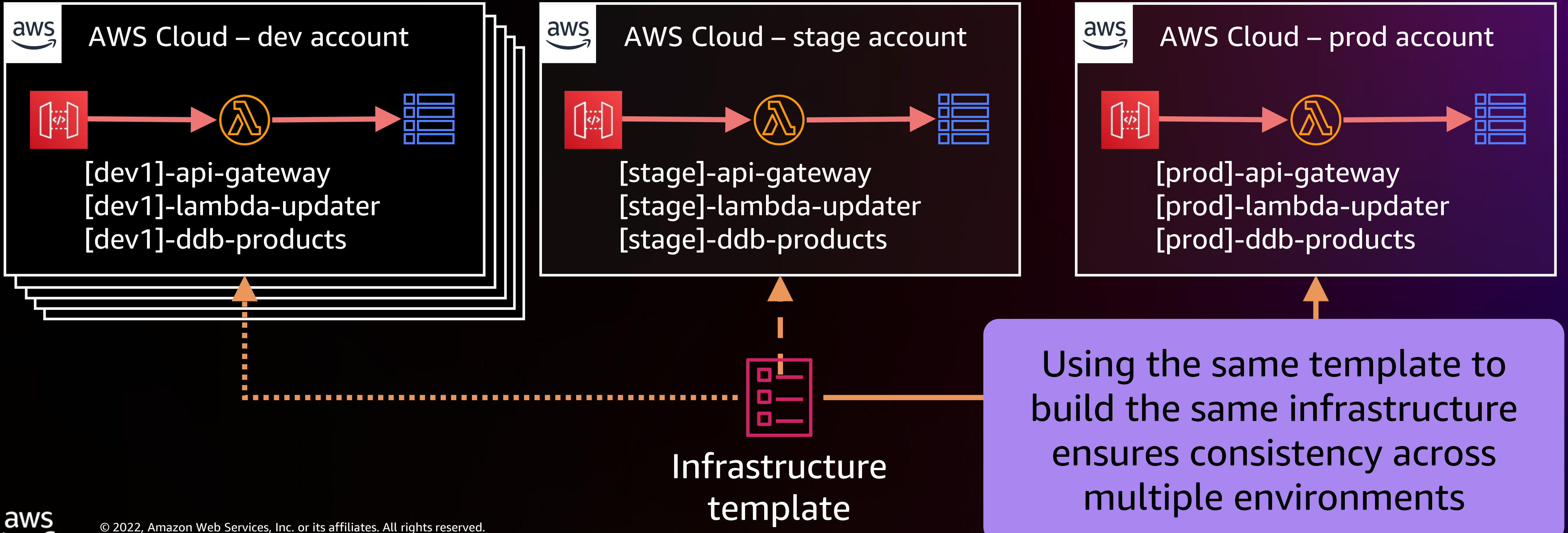
[stage]-api-gateway
[stage]-lambda-updater
[stage]-ddb-products

AWS Cloud – prod account

[prod]-api-gateway
[prod]-lambda-updater
[prod]-ddb-products

Infrastructure template

Reusable templates

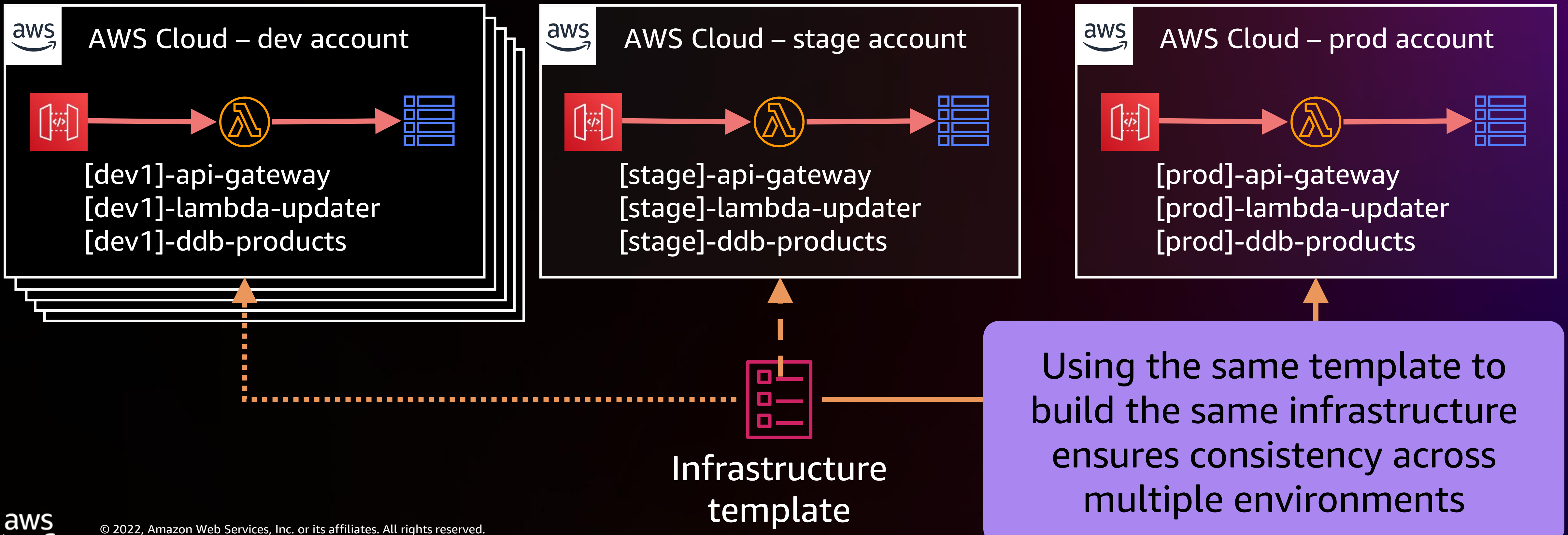


Using the same template to build the same infrastructure ensures consistency across multiple environments

Reusable templates

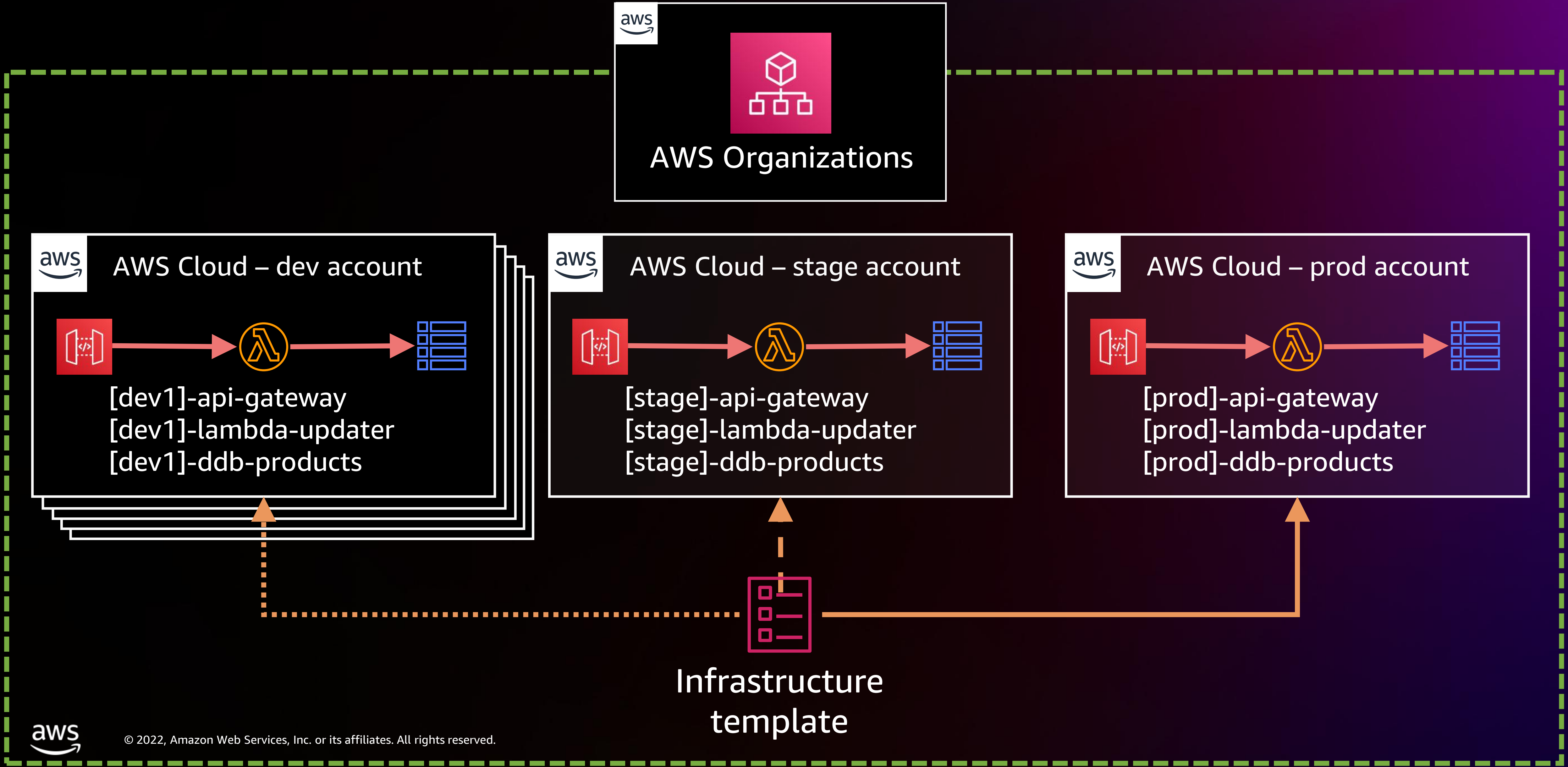
Can store environment configurations in Systems Manager Parameter Store

AWS Systems Manager Parameter Store



Using the same template to build the same infrastructure ensures consistency across multiple environments

AWS account per environment



AWS Organizations

AWS Cloud – dev account

[dev1]-api-gateway
[dev1]-lambda-updater
[dev1]-ddb-products

AWS Cloud – stage account

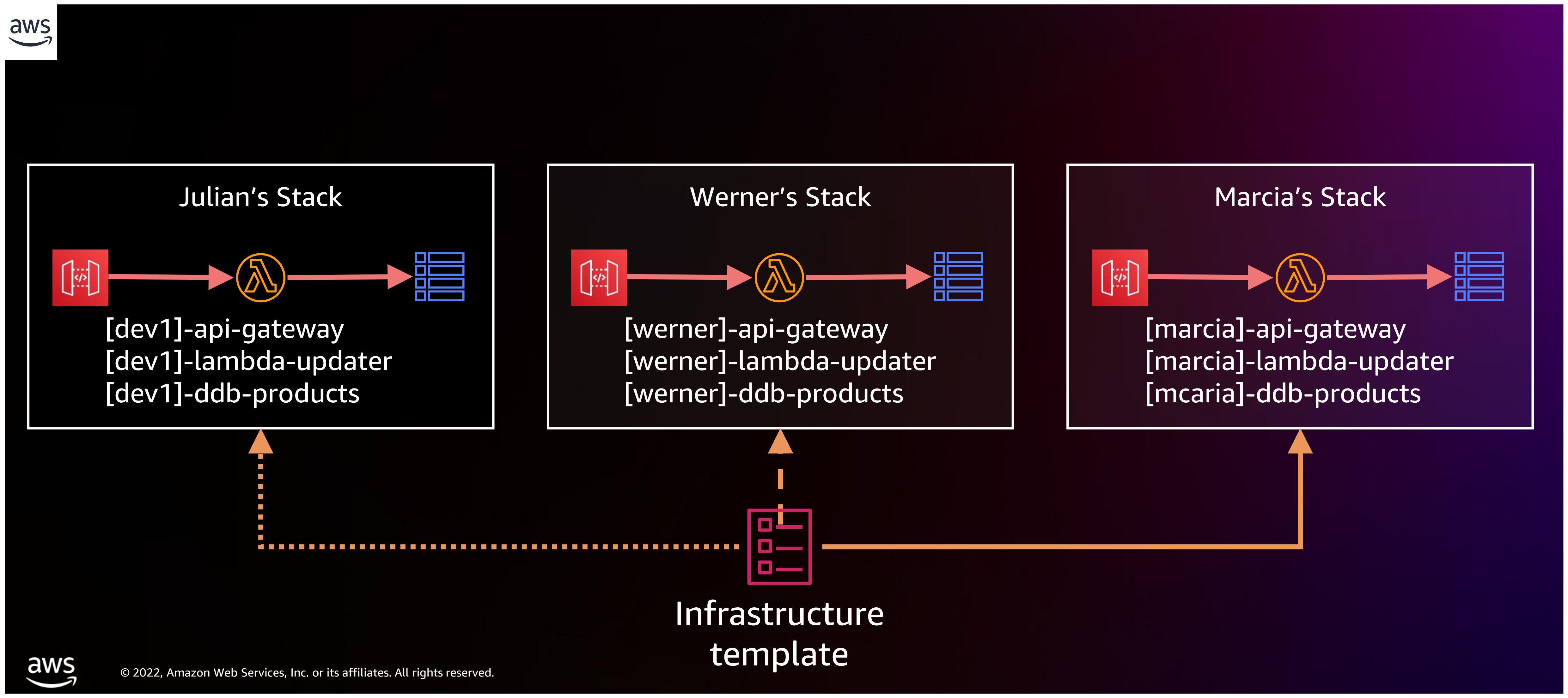
[stage]-api-gateway
[stage]-lambda-updater
[stage]-ddb-products

AWS Cloud – prod account

[prod]-api-gateway
[prod]-lambda-updater
[prod]-ddb-products

Infrastructure template

Shared accounts with prefixing



Julian's Stack

[dev1]-api-gateway
[dev1]-lambda-updater
[dev1]-ddb-products

Werner's Stack

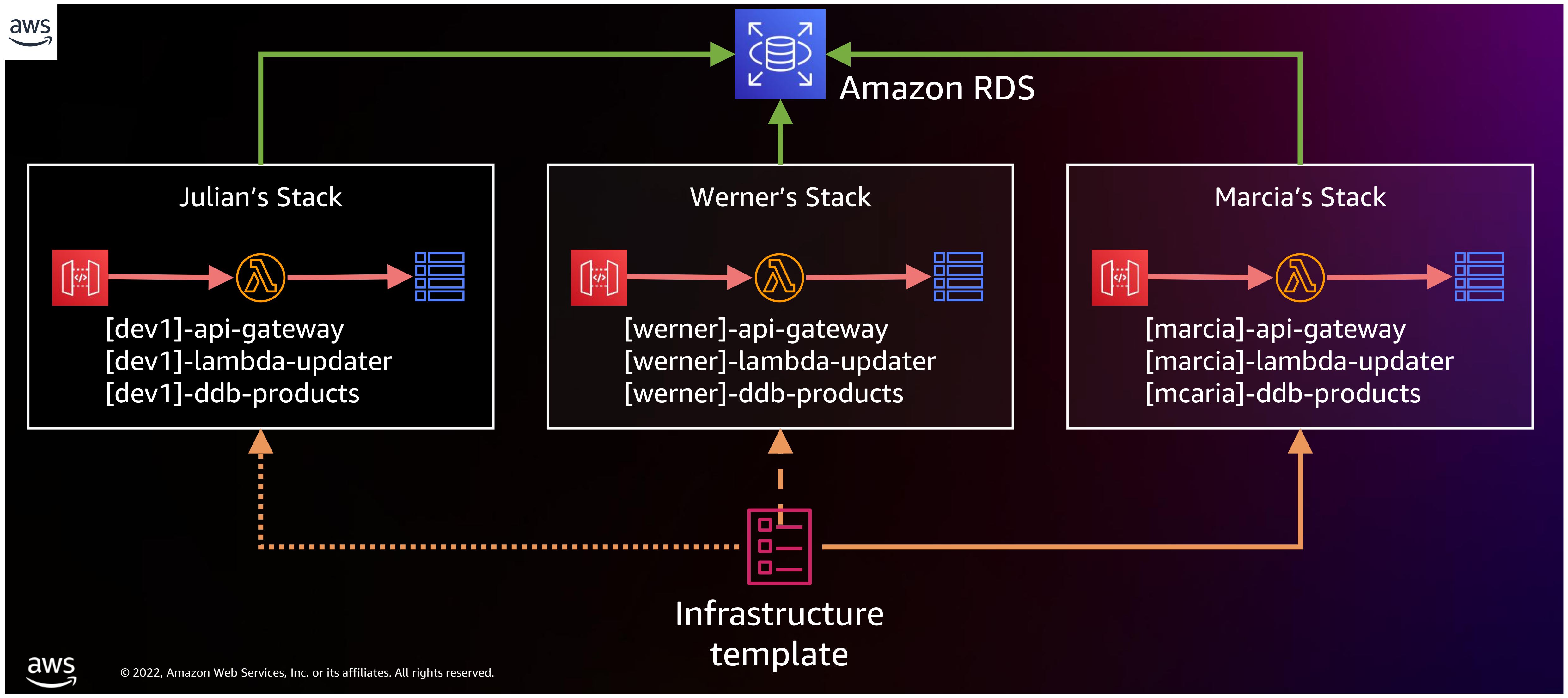
[werner]-api-gateway
[werner]-lambda-updater
[werner]-ddb-products

Marcia's Stack

[marcia]-api-gateway
[marcia]-lambda-updater
[mcaria]-ddb-products

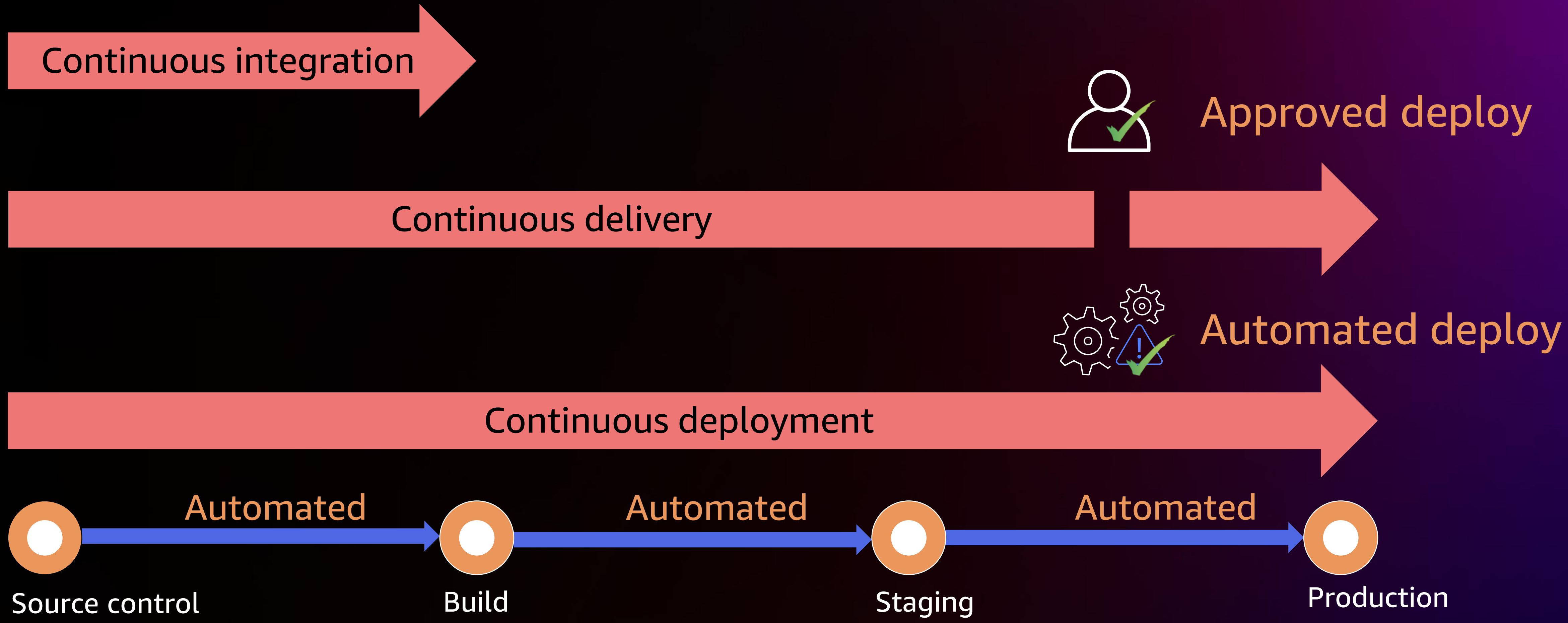
Infrastructure template

Share expensive/non scale-to-zero resources



The CI/CD pipeline

CONTINUOUS INTEGRATION, DELIVERY, AND DEPLOYMENT PIPELINE



Continuous integration

Approved deploy

Continuous delivery

Automated deploy

Continuous deployment

Automated

Automated

Automated

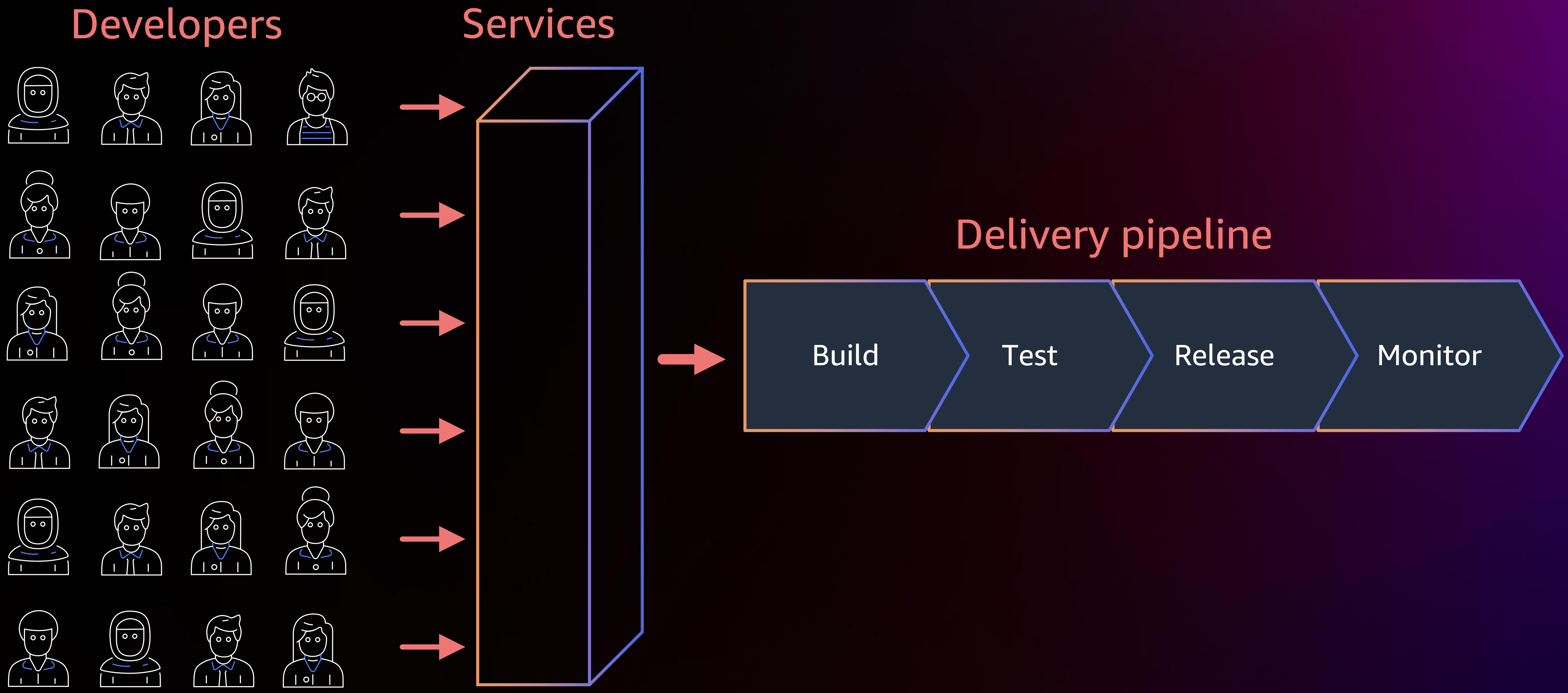
Source control

Build

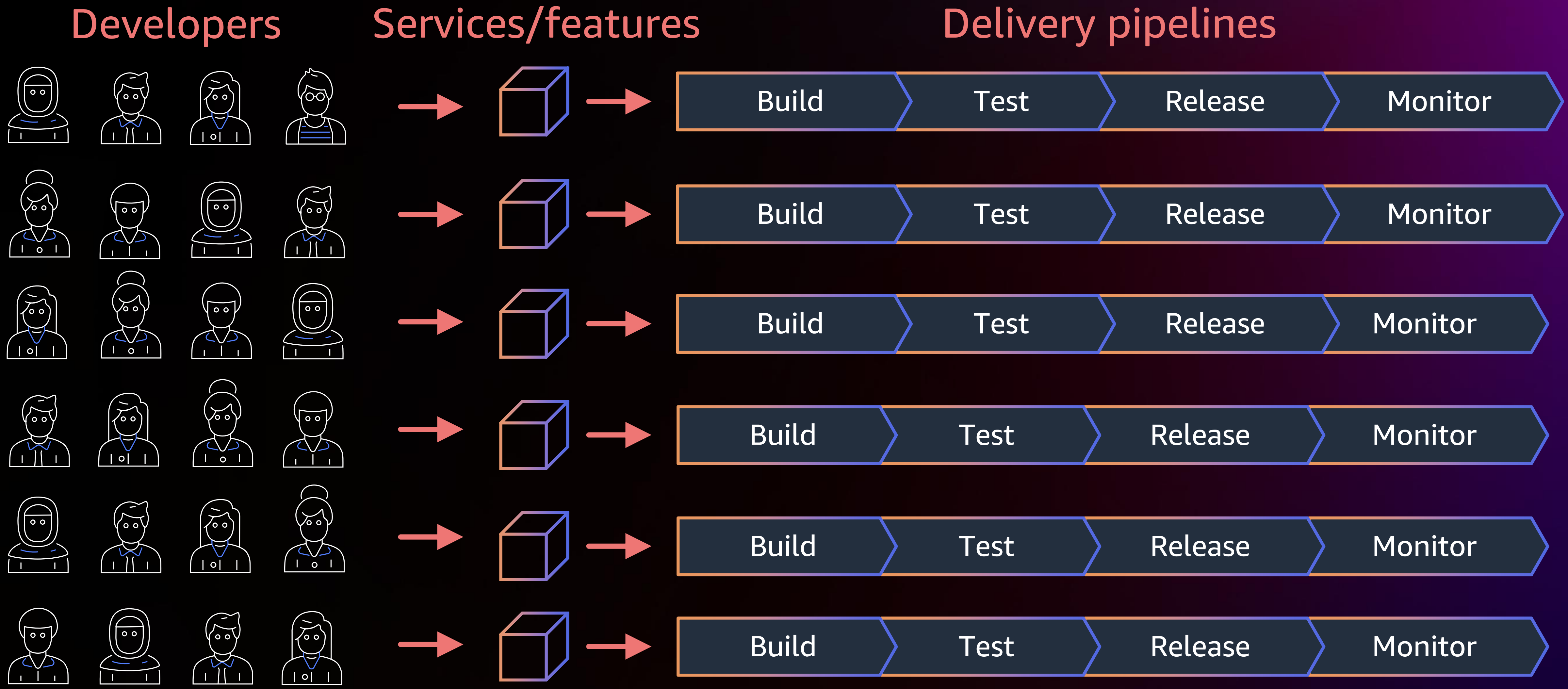
Staging

Production

Single development pipeline



Multiple development pipelines



Developers

Services/features

Delivery pipelines

Build → Test → Release → Monitor

Build → Test → Release → Monitor

Build → Test → Release → Monitor

Build → Test → Release → Monitor

Build → Test → Release → Monitor

Build → Test → Release → Monitor

AWS SAM pipelines

`sam pipeline init --bootstrap`

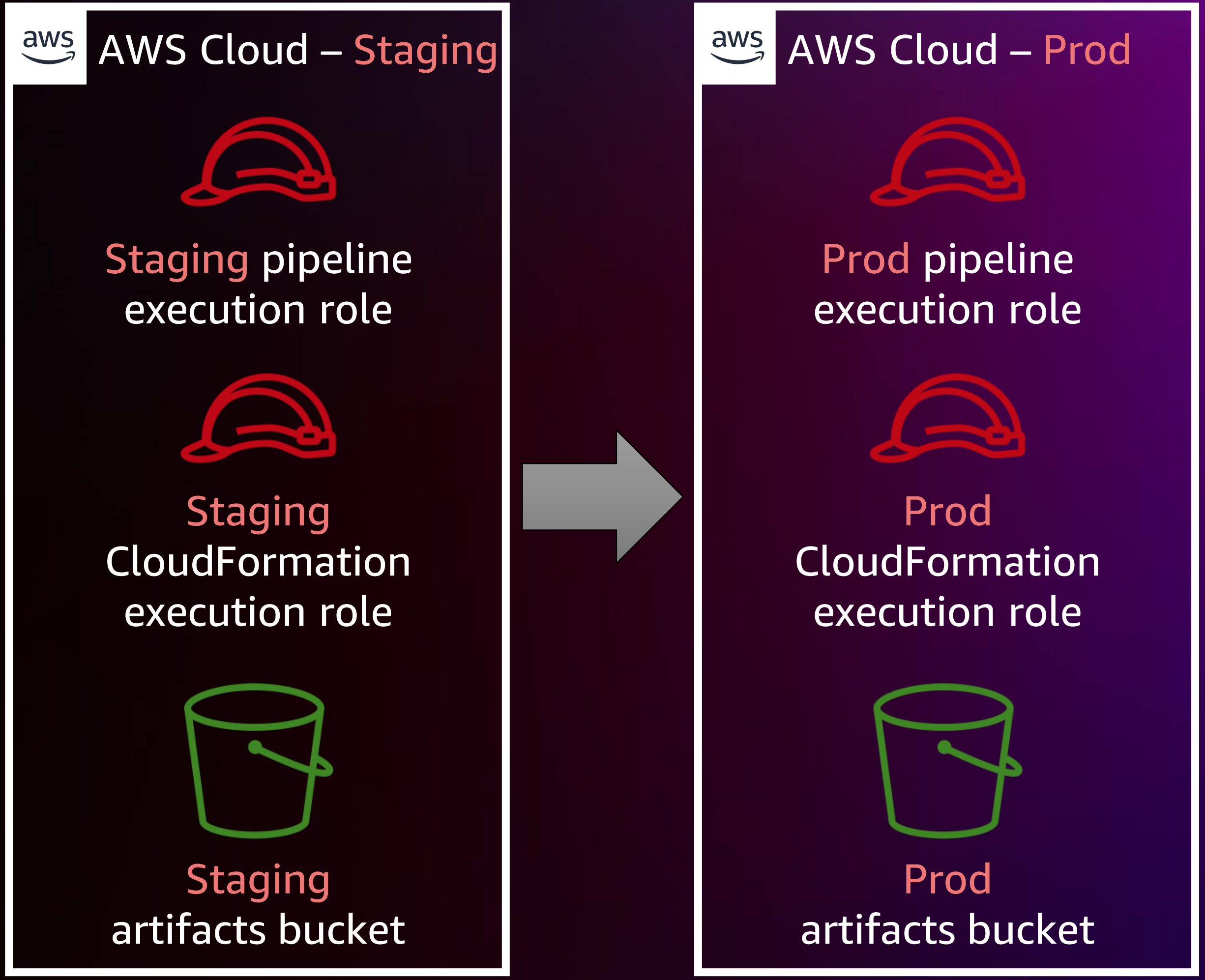
Creates the AWS resources and permissions required to deploy application artifacts from your code repository into your AWS environments

IAM or OIDC authorizer

s12d.com/sam-pipelines

GitHub

GitLab



AWS Cloud – Staging

Staging pipeline execution role

Staging CloudFormation execution role

Staging artifacts bucket

AWS Cloud – Prod

Prod pipeline execution role

Prod CloudFormation execution role

Prod artifacts bucket

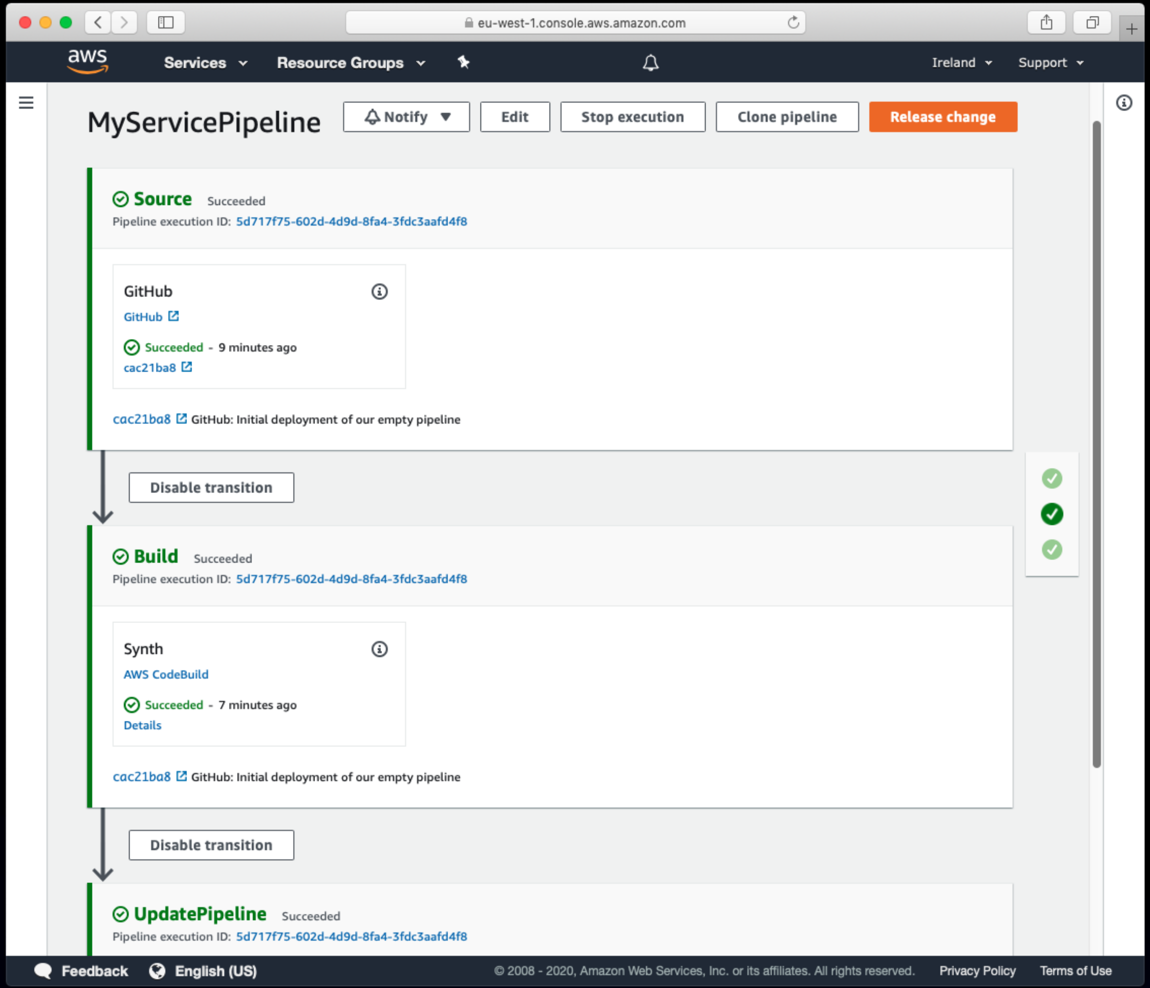
AWS CodePipeline

Jenkins

AWS CDK Pipelines

AWS CDK Pipelines is a high-level construct library that makes it easy to set up a continuous deployment pipeline for your CDK applications, powered by AWS CodePipeline

s12d.com/cdk-pipelines



MyServicePipeline

Source Succeeded
Pipeline execution ID: 5d717f75-602d-4d9d-8fa4-3fdc3aafd4f8

GitHub
GitHub
Succeeded - 9 minutes ago
cac21ba8

cac21ba8 GitHub: Initial deployment of our empty pipeline

Build Succeeded
Pipeline execution ID: 5d717f75-602d-4d9d-8fa4-3fdc3aafd4f8

Synth
AWS CodeBuild
Succeeded - 7 minutes ago
Details

cac21ba8 GitHub: Initial deployment of our empty pipeline

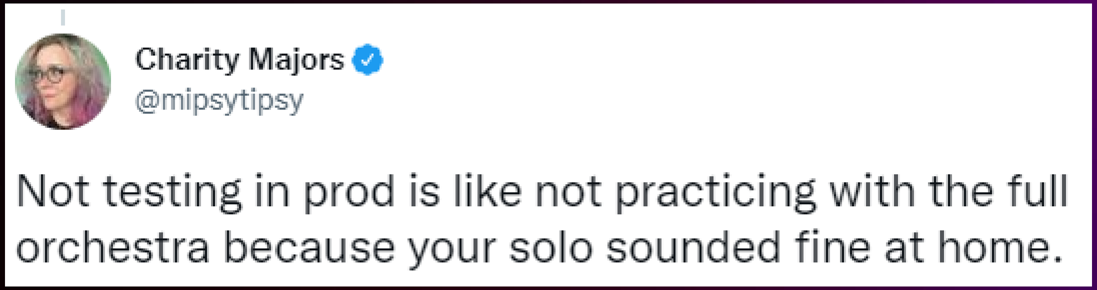
UpdatePipeline Succeeded
Pipeline execution ID: 5d717f75-602d-4d9d-8fa4-3fdc3aafd4f8

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Confident production

- Testing in production
  - Deploy to a subset of traffic
  - See how it compares to the previous version
  - Discover the “unknown unknowns”

- Observability-driven development
  - Include observability during development process
  - Is the system behaving as expected?
  - How is the system used?
  - What is the business impact?

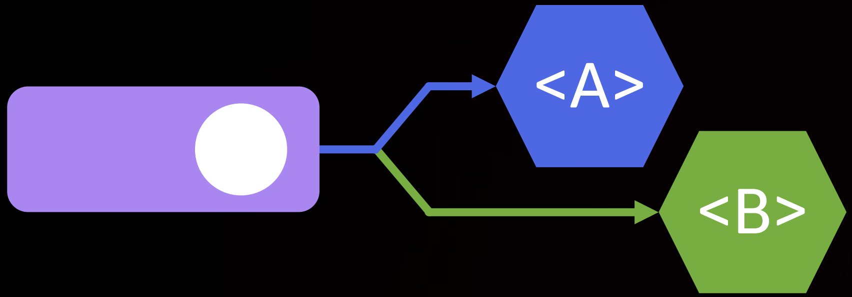


Charity Majors
@mipsytipsy

Not testing in prod is like not practicing with the full orchestra because your solo sounded fine at home.

Feature flags/toggles

```
function getPaymentOptions(){
  if(FLAG_ADD_AMEX){
    return [“Visa”, ”Mastercard”, “Amex”];
  }else{
    return [“Visa”, ”Mastercard”];
  }
}
```



AWS AppConfig Feature Flags

Amazon CloudWatch Evidently

split

LaunchDarkly

Optimizely

Canaries

SCRIPTS THAT EXTERNALLY MONITOR YOUR ENDPOINTS AND APIS

s12d.com/cw-synthetics-canaries

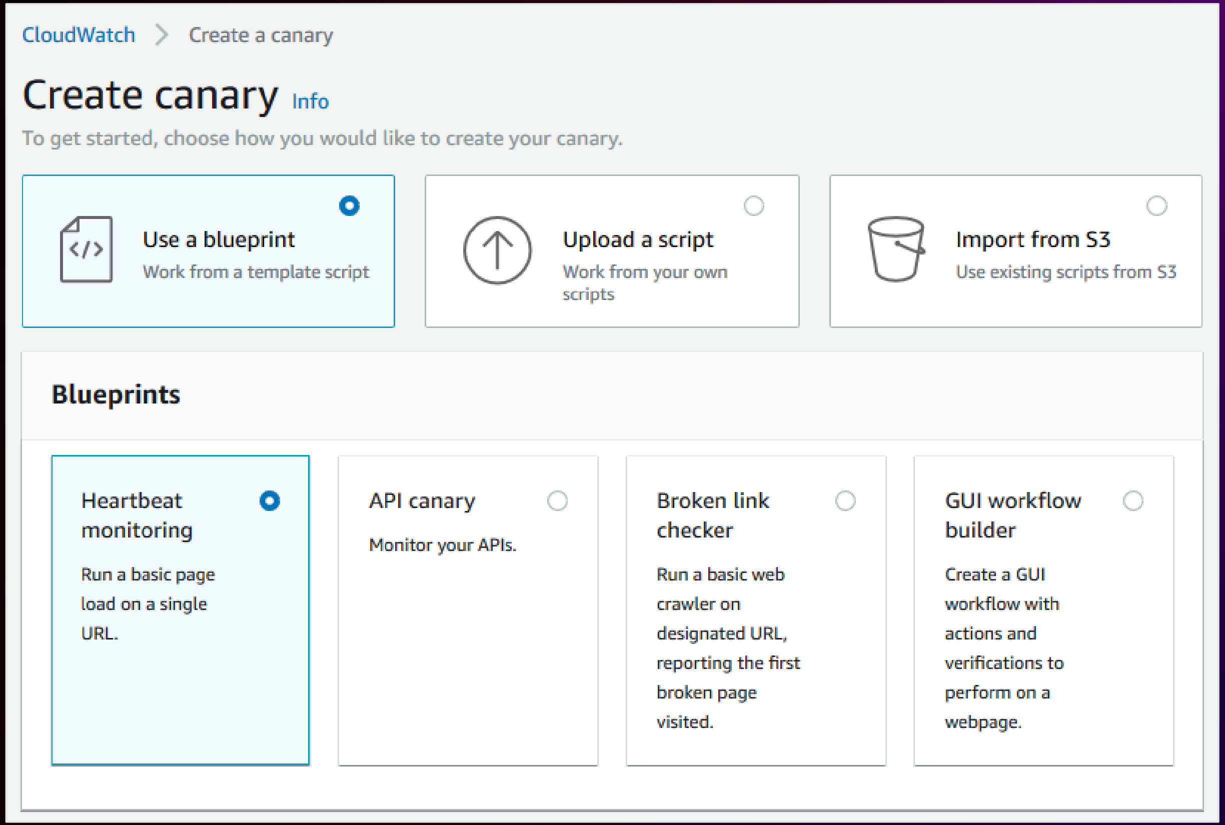
Amazon CloudWatch Synthetics

Monitor web applications using modular, lightweight canary tests

Help you check the availability and latency of your web services

Troubleshoot anomalies by investigating load time data, screenshots of the UI, logs, and metrics

Run continuously or just once



CloudWatch > Create a canary

Create canary Info

To get started, choose how you would like to create your canary.

● Use a blueprint
Work from a template script

○ Upload a script
Work from your own scripts

○ Import from S3
Use existing scripts from S3

Blueprints

● Heartbeat monitoring
Run a basic page load on a single URL.

○ API canary
Monitor your APIs.

○ Broken link checker
Run a basic web crawler on designated URL, reporting the first broken page visited.

○ GUI workflow builder
Create a GUI workflow with actions and verifications to perform on a webpage.

Amazon CloudWatch embedded metric format

AUTOMATICALLY CREATE METRICS FROM LOG ENTRIES

Event payload

```
message = {
  "PriceInCart": 100,
  "QuantityInCart": 2,
  "ProductId": "a23390f3",
  "CategoryId": "bca4cec1",
  "Environment": "prod",
  "UserId": "31ba3930",
  "CartId": "58dd189f",
"LogLevel": "INFO",
  "Timestamp": "2019-12-11
12:44:40.300473",
  "Message": "Added 2 items
'a23390f3' to cart '58dd189f'"
}
```

Log entry

```
[...]
  "_aws": {
  "functionVersion": "$LATEST",
    "Timestamp": 1576064416496,
    "CloudWatchMetrics": [{
      "Namespace": "ecommerce-cart",
      "Dimensions": [
        ["Environment", "CategoryId"]
      ],
      "Metrics": [
        {"Name": "PriceInCart", "Unit": "None"},
        {"Name": "QuantityInCart", "Unit": "None"}
      ]}]},
 "Environment": "prod",
 "CategoryId": "bca4cec1"
 "PriceInCart": 100,
 "QuantityInCart": "2"
}
```

aws

Amazon CloudWatch embedded metric format

AUTOMATICALLY CREATE METRICS FROM LOG ENTRIES

Event payload

Log entry

Open-source client libraries available for

- Node.js
- Python
- Java
- C#

s12d.com/cwl-emf-client

```
message = {
  "PriceInCart": 100,
  "QuantityIn
  "ProductId": "a23390f3",
  "CategoryId": "   4cec1",
  "Environment": "prod",
  "UserId": "   3930",
  "CartId": "58dd189f",
"LogLevel": "INFO",
  "Timestamp": "2019-12-11
12:44:40.300473",
  "Message": "Added 2 items
'a23390f3' to cart '58dd189f'"
}
```

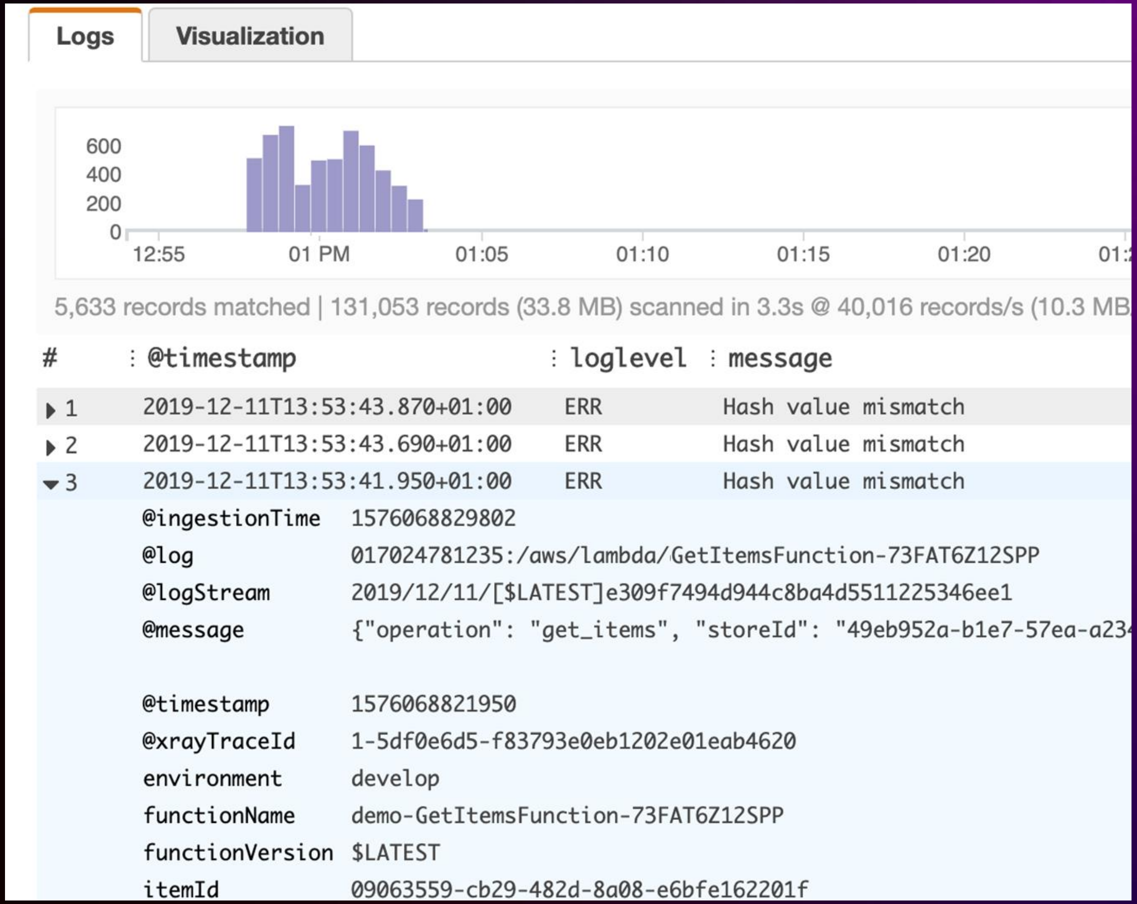
```
"aws": {
           $LATEST",
           64416496,
           ": [{
           ommerce-cart",
           , "CategoryId"]

           eInCart", "Unit": "None"},
           tityInCart", "Unit": "None"}

 Environment": "prod",
"CategoryId": "bca4cec1"
"PriceInCart": 100,
"QuantityInCart": "2"
}
```

Querying logs

Amazon CloudWatch Logs Insights

- Interactively search and analyze your log data in CloudWatch Logs
- Processes structured log data
- Flexible purpose-built query language
- Query up to 20 log groups
- Save queries

```
fields Timestamp, LogLevel, Message
| filter LogLevel == "ERR"
| sort @timestamp desc
| limit 10
```



5,633 records matched | 131,053 records (33.8 MB) scanned in 3.3s @ 40,016 records/s (10.3 MB

```
#    : @timestamp                      : loglevel  : message
▸ 1    2019-12-11T13:53:43.870+01:00     ERR         Hash value mismatch
▸ 2    2019-12-11T13:53:43.690+01:00     ERR         Hash value mismatch
▾ 3    2019-12-11T13:53:41.950+01:00     ERR         Hash value mismatch
       @ingestionTime   1576068829802
       @log             017024781235:/aws/lambda/GetItemsFunction-73FAT6Z12SPP
       @logStream       2019/12/11/[$LATEST]e309f7494d944c8ba4d5511225346ee1
       @message         {"operation": "get_items", "storeId": "49eb952a-b1e7-57ea-a23

       @timestamp       1576068821950
       @xrayTraceId     1-5df0e6d5-f83793e0eb1202e01eab4620
       environment      develop
       functionName     demo-GetItemsFunction-73FAT6Z12SPP
       functionVersion  $LATEST
       itemId           09063559-cb29-482d-8a08-e6bfe162201f
```

aws

s12d.com/snippets-cwli

AWS Lambda Powertools

A SUITE OF UTILITIES FOR LAMBDA FUNCTIONS

Logging:	Output as structured JSON
Tracing:	Send traces to AWS X-Ray
Metrics:	Custom metrics with embedded metric format
Utilities:	Parameters, idempotency, SQS processing, and more
	(language dependent)

Python	Java	Typescript/JavaScript	.NET (preview)
s12d.com/powertools-python	s12d.com/powertools-java	s12d.com/powertools-typescript	s12d.com/powertools-dotnet

Serverless observability learning path

MASTERING SERVERLESS APPLICATION OBSERVABILITY

8 Episode Series
with Julian Wood, Senior Developer Advocate

s12d.com/mastering-observability

One observability workshop

Hands-on experience to set up monitoring and observability for your applications

3–4 hours self-paced

- CloudWatch ServiceLens Map
- AWS X-Ray
- Contributor Insights
- CloudWatch Synthetics
- CloudWatch RUM
- CloudWatch Evidently
- Container Insights
- Logs Insights
- Lambda Insights
- Metrics
- Dashboards
- Anomaly detection
- Embedded Metric Format
- Alarms
- Amazon Managed Service for Prometheus
- Amazon Managed Grafana
- AWS Observability Accelerator
- AWS Distro for OpenTelemetry
- Load test & troubleshoot

s12d.com/observability-workshop

From prototype to production: Best practices

- Avoid emulating services locally
- Test business logic locally, the rest in the cloud
- Try AWS SAM accelerate
- Build reusable templates from a single codebase in version control
- Create CI/CD superpowers and build/deploy/test on each commit
- Use embedded metrics format: metrics from logs
- Explore feature flags/canaries
- Use Lambda Powertools

Summary

Serverless is?

Event state

Service-full serverless

Fabulous functions

Configuration as code

From prototype to production

Resources

s12d.com/svs401-22

Continue your AWS Serverless learning

Learn at your own pace

Expand your serverless skills with our Learning Plan on **AWS Skill Builder**

Increase your knowledge

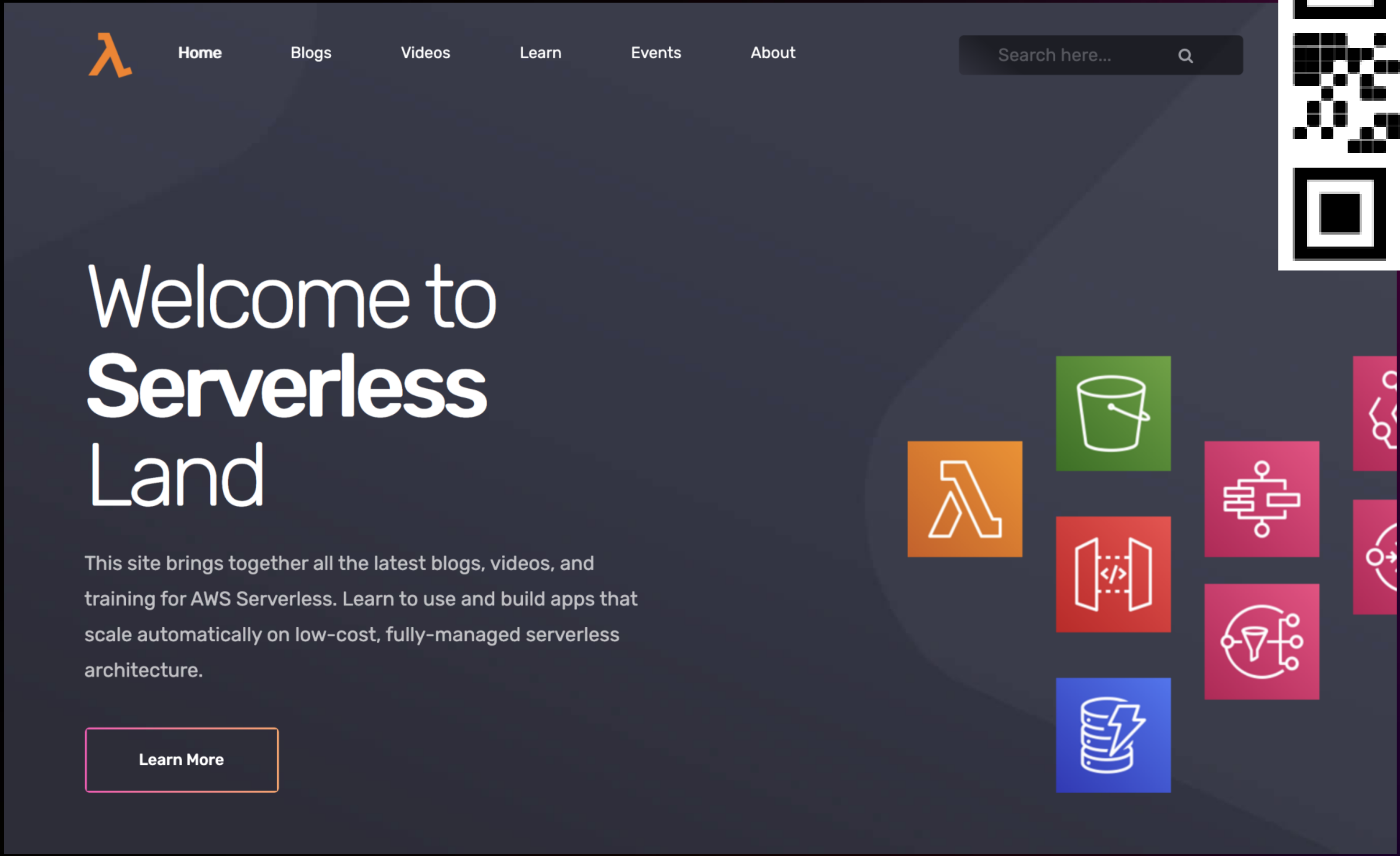
Use our **Ramp-Up Guides** to build your serverless knowledge

Earn AWS Serverless badge

Demonstrate your Knowledge by achieving **digital badges**

s12d.com/serverless-learning

Serverlessland.com



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.

Thank you!

Julian Wood

@julian_wood

Email: jrwood@amazon.com

Please complete the session survey in the **mobile app**