AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

SVS402-R1

Best practices of advanced serverless developers

Julian Wood
Senior Developer Advocate, Serverless
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What are we talking about today?

Thinking in events

Service-full serverless

Fabulous functions

Healthy serverless

Development workflow

Resources



s12d.com/svs402-ri21

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking in events

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The start of “serverless”?

November 13, 2014

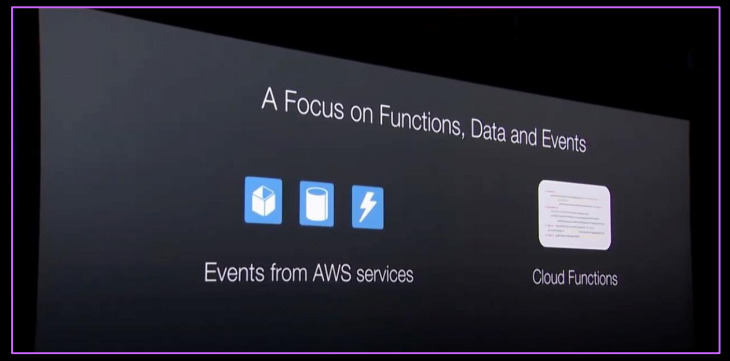
AWS introduces the preview of AWS Lambda, an event-driven computing service for dynamic applications

April 9, 2015

AWS Lambda becomes generally available for production use

No mention of “serverless”!









© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

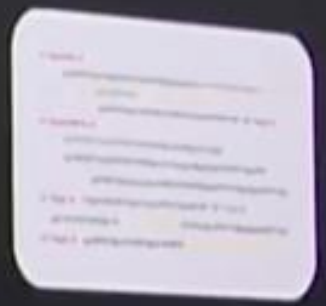


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

A Focus on Functions, Data and Events

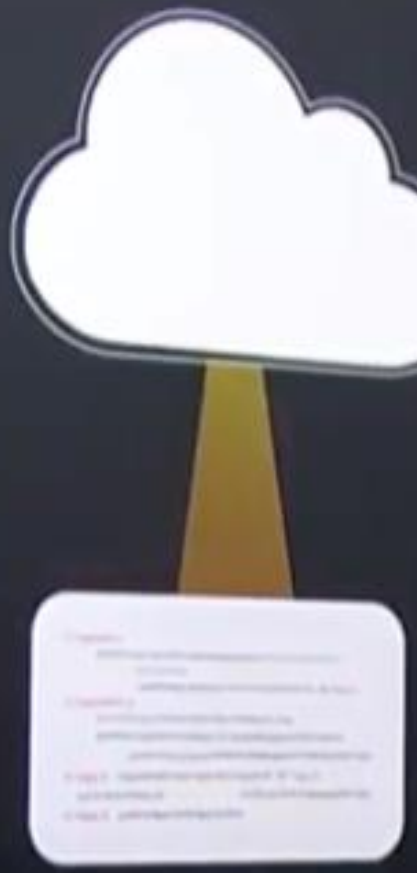


Events from AWS services



Cloud Functions

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Easy to Use, and Low Maintenance

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



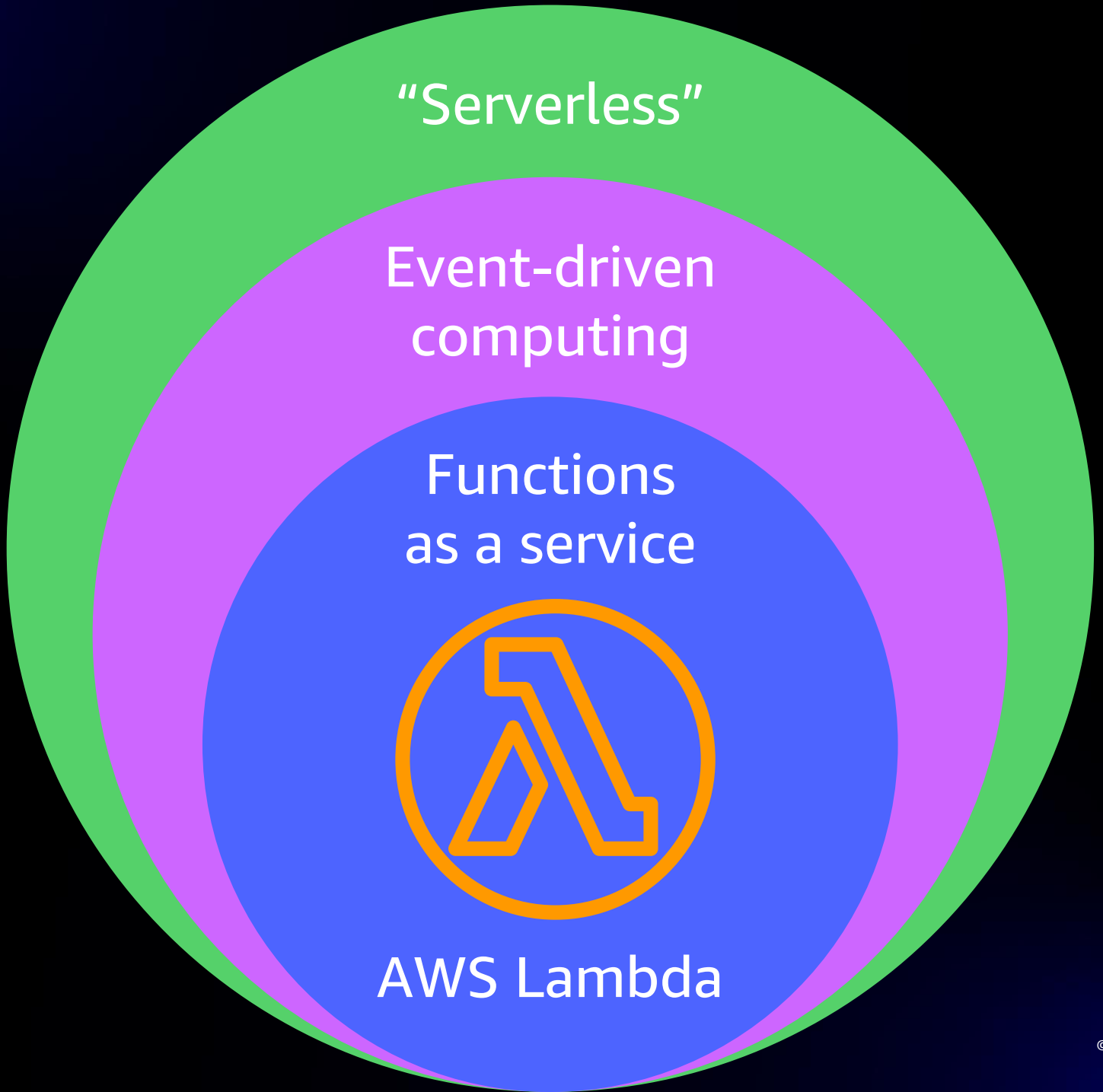
Cost Effective and Efficient

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Very Rapid Responses to Events



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What are serverless services?



Amazon EventBridge



Amazon SNS



Amazon SQS



Amazon API Gateway



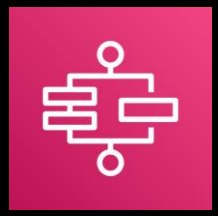
Amazon DynamoDB



AWS Lambda



Amazon S3



AWS Step Functions



Amazon Kinesis



AWS IoT Core

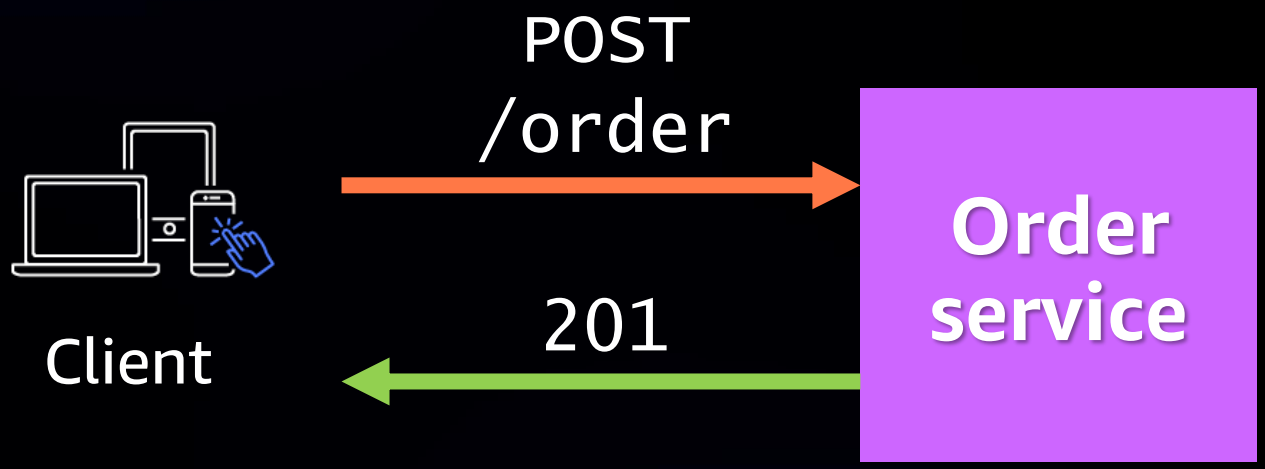


Amazon Elastic Transcoder

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

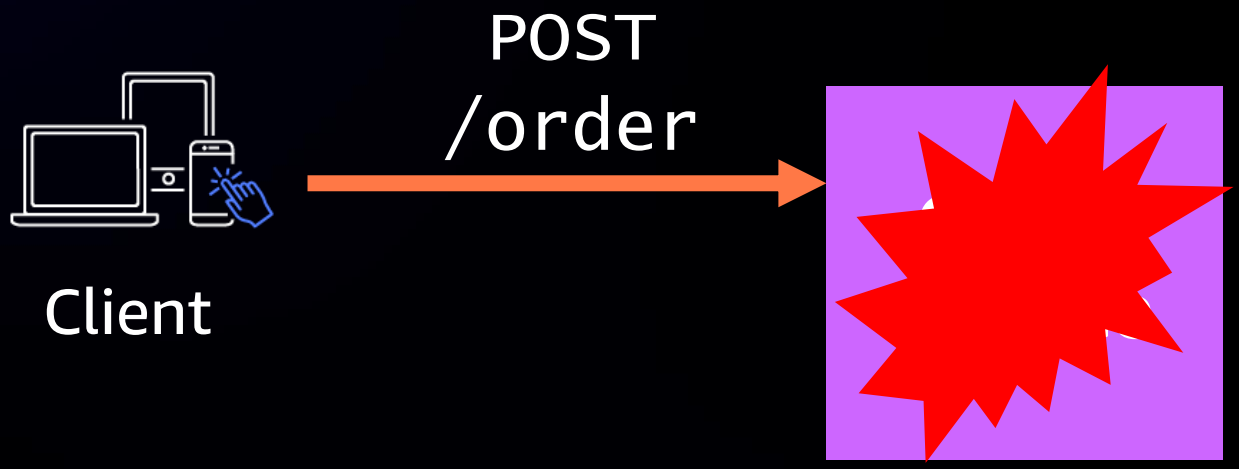
Synchronous APIs



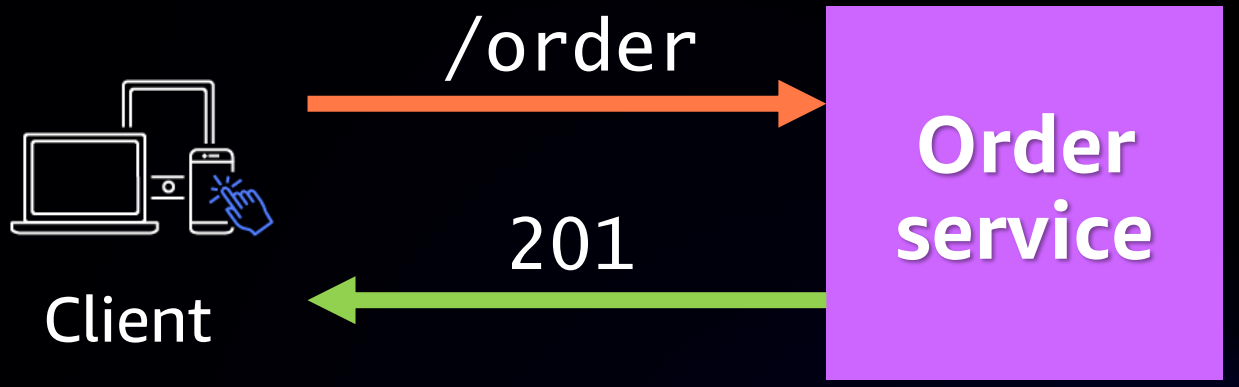
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous APIs



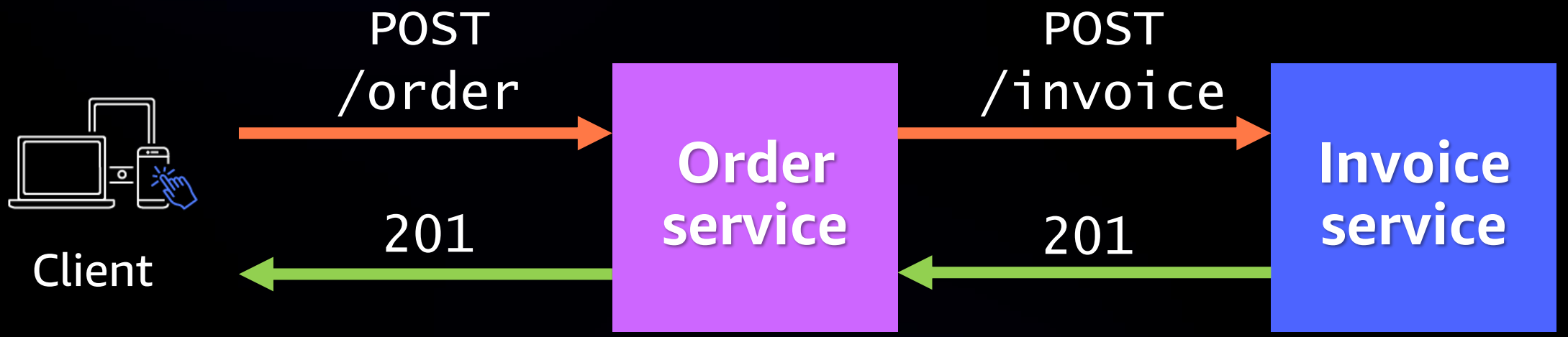
retry after failure



aws

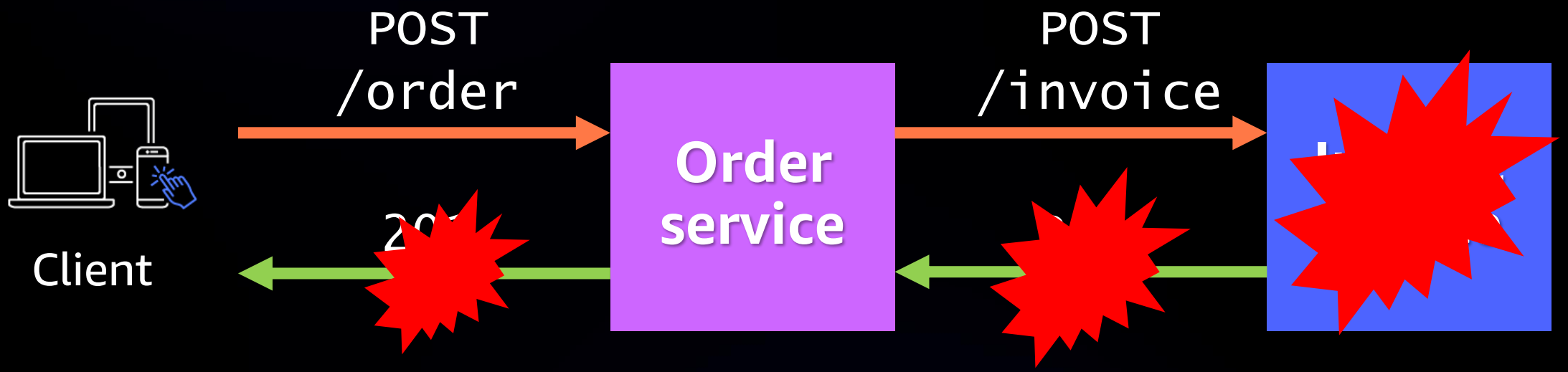
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous APIs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous APIs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous APIs



Who owns the retry? For how long? Does the client ever know? And so on.

This effectively creates a “tight coupling” where failures become harder to recover from

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous APIs



Client

POST
/order

201

Order service

POST
/invoice

Invoice service



Client

GET
/invoice

200

Invoice service

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Async friends: Queues, topics, buses, streams



Amazon SQS

Queues
Fully managed

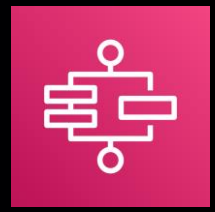
Message queuing service to decouple and scale distributed systems



Amazon SNS

Pub/sub topics
Fully managed

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



EventBridge

Event bus
Fully managed

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



Kinesis

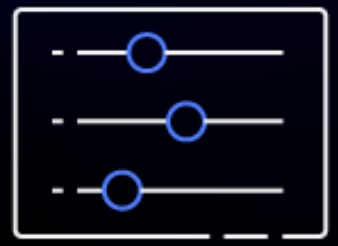
Streams
Fully managed

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

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ways to compare



Scale/concurrency controls



Durability



Persistence



Consumption models



Retries



Pricing

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ways to compare

Choosing Events, Queues, Topics, and Streams in Your Serverless Application

AWS Online Tech Talks



s12d.com/events-queues-topics-streams

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

event

[i-’vent] **noun**

A signal that a system’s state has changed.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

event

[i-’vent] **noun**

A signal that a system’s state has changed.

- Facts
- Temporal
- Immutable
- Observable
- JSON

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Commands/APIs	Events
Synchronous	Asynchronous
Has an intent Directed to a target	Is a fact Others can observe
“DoSomething” “CreateAccount” “AddProduct” “CreateFile”	“SomethingDone” “AccountCreated” “ProductAdded” “FileCreated” “PaymentFailed”

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Event storming

Event discovery

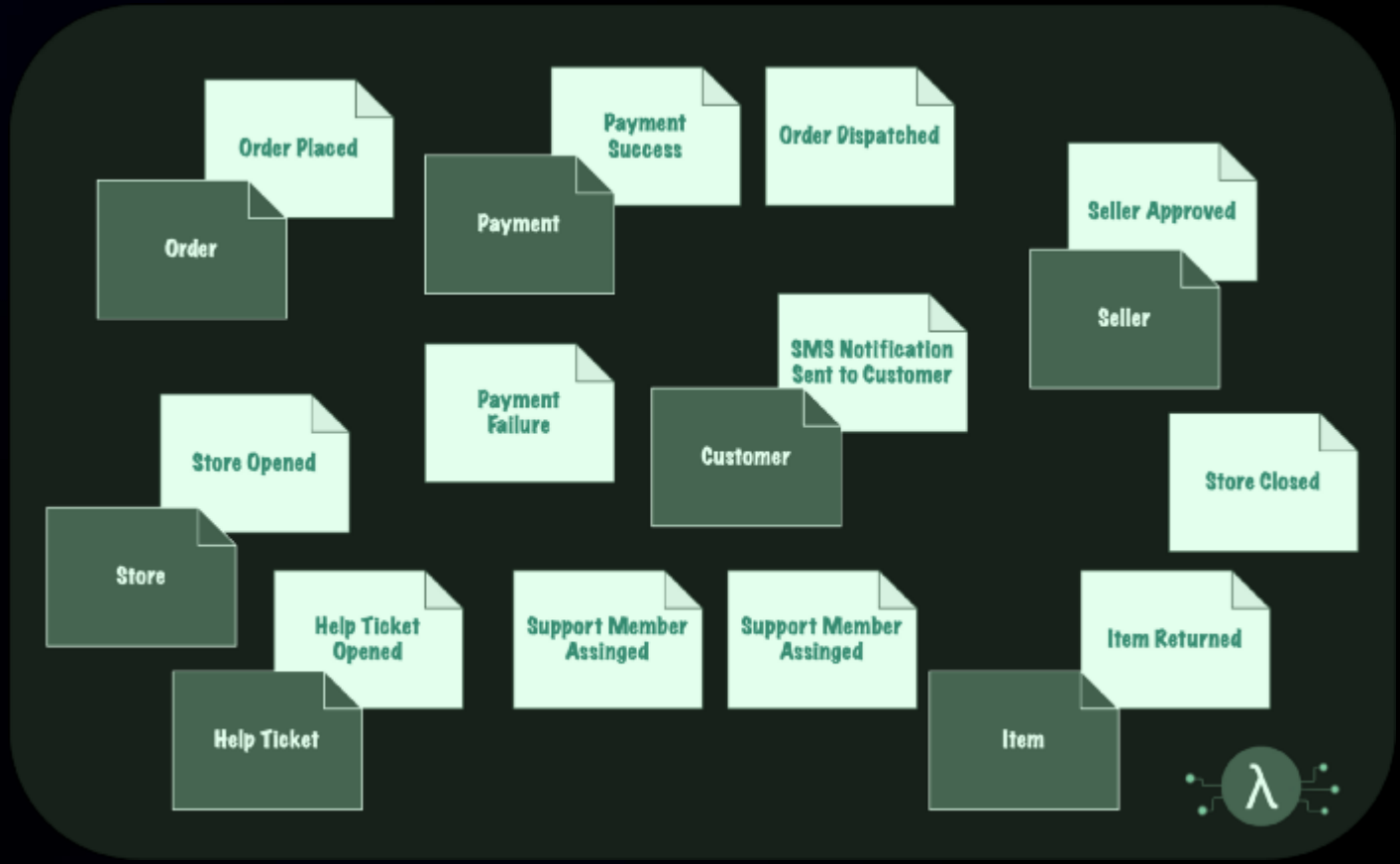
Temporal sequencing

Trigger detection

Categorization

Naming

Schema

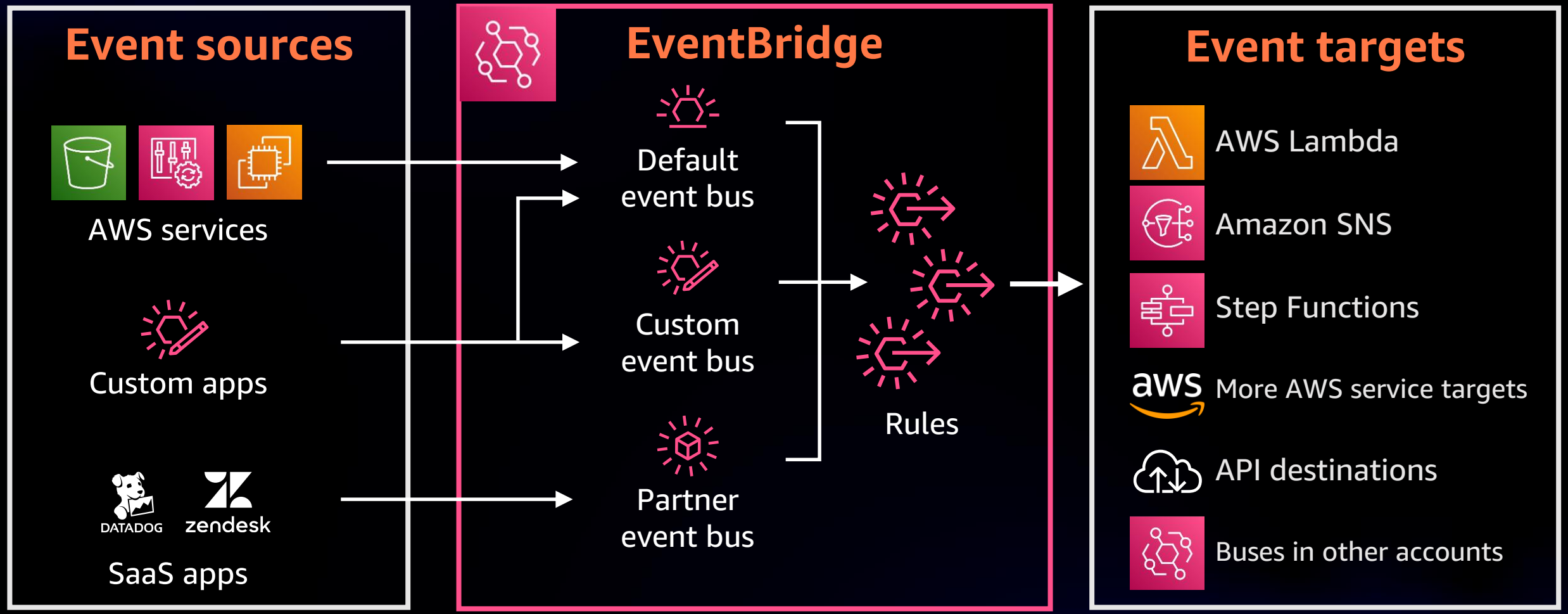


Ben Ellerby | AWS Serverless Hero

s12d.com/eventbridge-storming

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon EventBridge sources and targets



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deciding on event content

Send all information

(Fat event)

For example, for a change event:

- Object ID
- Previous content
- New content
- All fields
- Metadata on which fields have changed

Send only metadata

(Thin event)

For example, for a change event:

- Object ID

Consumers then make an API call to retrieve data

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deciding on event content

Send all information

(Fat event)

Sending too much information increases traffic and complexity

Send only metadata

(Thin event)

Having to contact an API introduces a runtime dependency

Ideally: Send events without considering the subscribers
Reality: Need to consider data, bounded contexts, services

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Enriching events

ADD EVENT ENVELOPE, METADATA, AND DATA DETAIL

Event envelope

```
{
  "version": "0",
  "id": "10ec61ab-d758-61f7-96a0-592d003f6b0b",
  "source": "MyCompany.MyServerlessApp",
  "detail-type": "Product.ProductInfoUpdated",
  "account": "111122223333",
  "time": "2021-09-15T19:47:52Z",
  "region": "us-west-2",
  "resources": [
        "MyServerlessApp-Product-
ProductInfoUpdatedFunction-VAv4YNEz6ojM"
  ],
```

Event detail

```
"detail": {
    "metadata": {
      "domain": "MyServerlessApp",
      "service": "Product",
      "environment": "prod",
      "category": "ProductInfoUpdated",
      "type": "PriceUpdated",
      "fields": ["salesPrice"]
    },
    "data": {
      "productId": "7ba35e6b",
      "salesPrice": 39.95,
      “priceChange": -0.95,
      "costPrice": 18.76
    }
  }
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Think in events: Best practices

- Plan events as part of your architecture design
- Embrace asynchronous and eventual consistency
- Use one or multiple messaging services
- Pass state as events
- Enrich events with content and metadata

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Service-full serverless
(configuration over code)

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

A serverless application



AWS Lambda

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

A serverless application

Function



Node.js
Python
Java
C#
Go
Ruby
Runtime API

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

A serverless application

Event source



Function



Changes in data state



Requests to endpoints



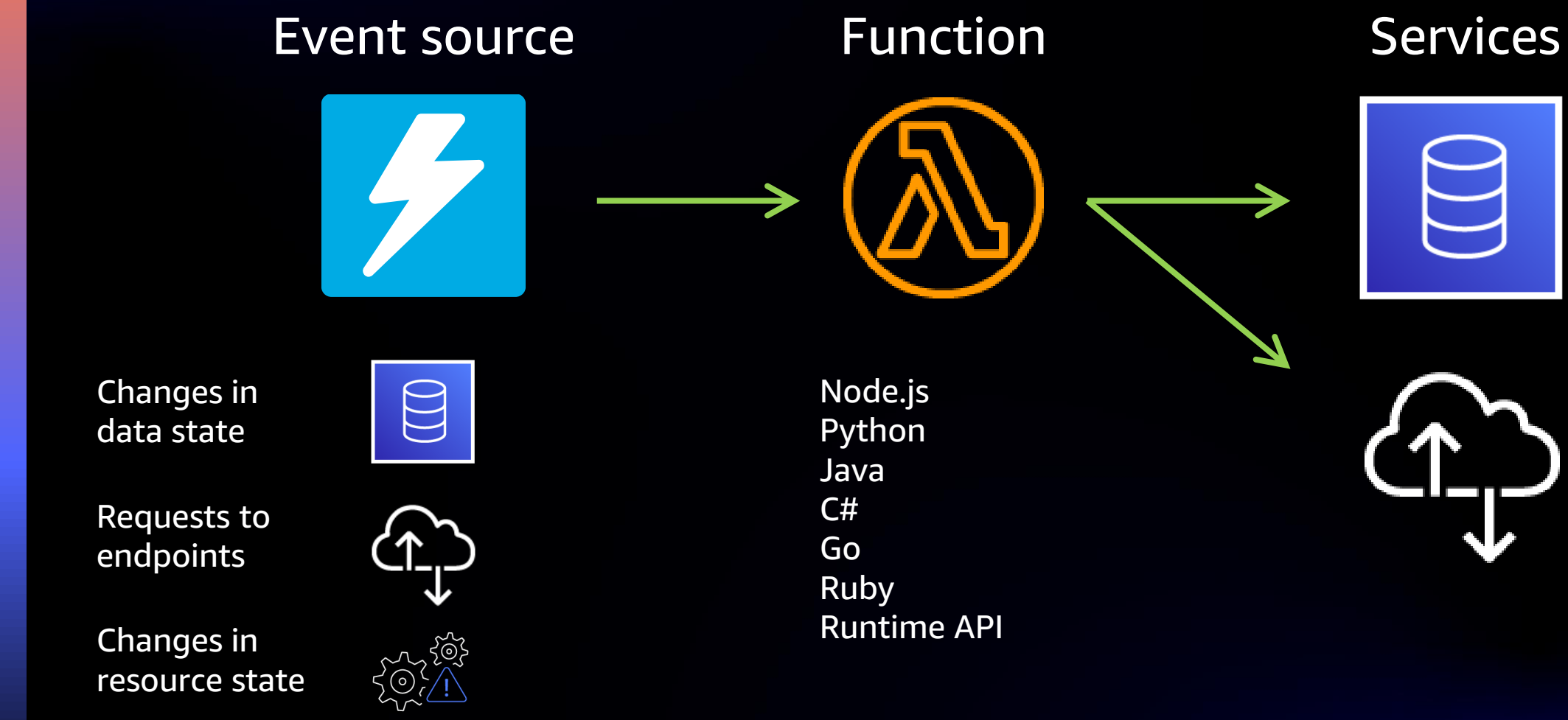
Changes in resource state



Node.js
Python
Java
C#
Go
Ruby
Runtime API

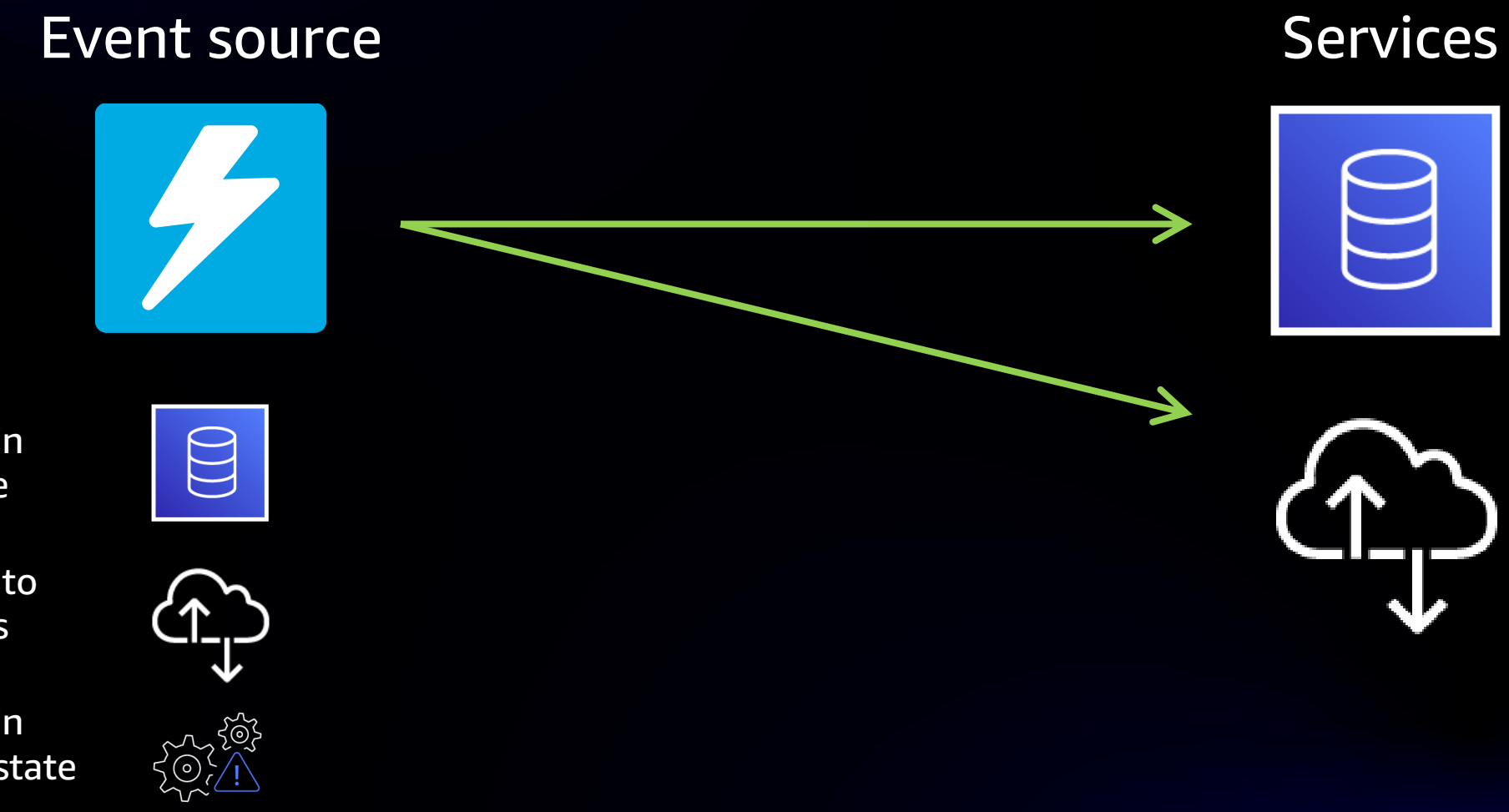
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

A serverless application



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

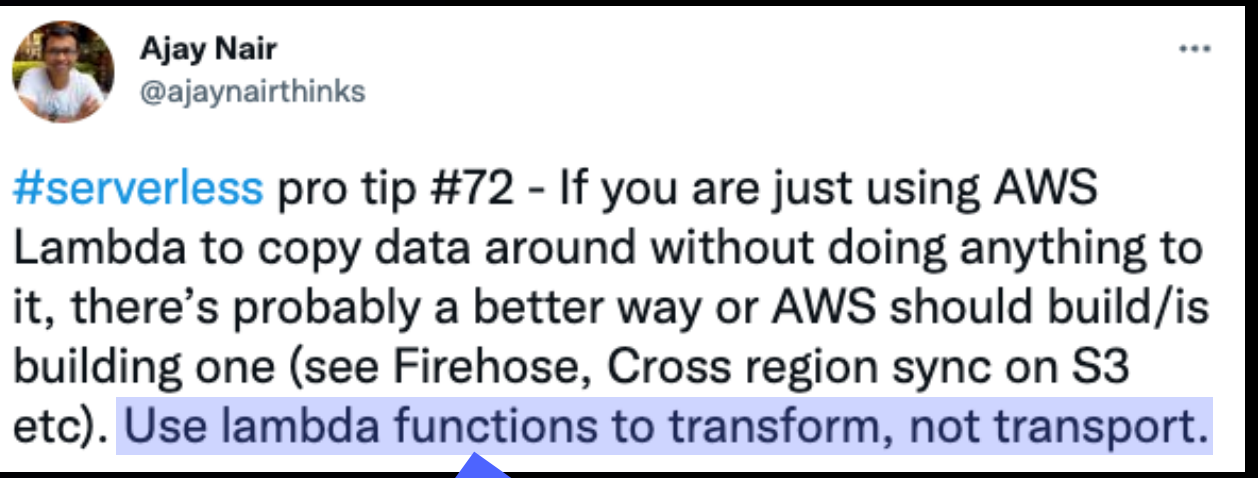
A serverless application



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Is an AWS Lambda function even needed?





Use AWS Lambda functions to transform, not transport

Image by Clker-Free-Vector-images from Pixabay

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How much

LOGIC



are you squeezing in to your code?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How little



are you actually invoking an AWS Lambda function for?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Consider a “traditional” app



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The lift and shift



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The migration

Offload processes to managed services



Front end

Auth
Caching logic



Throttling logic
Routing

Backend

Error handling



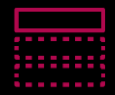
Bus



Queue



Workflow



Pub/sub

Retry logic











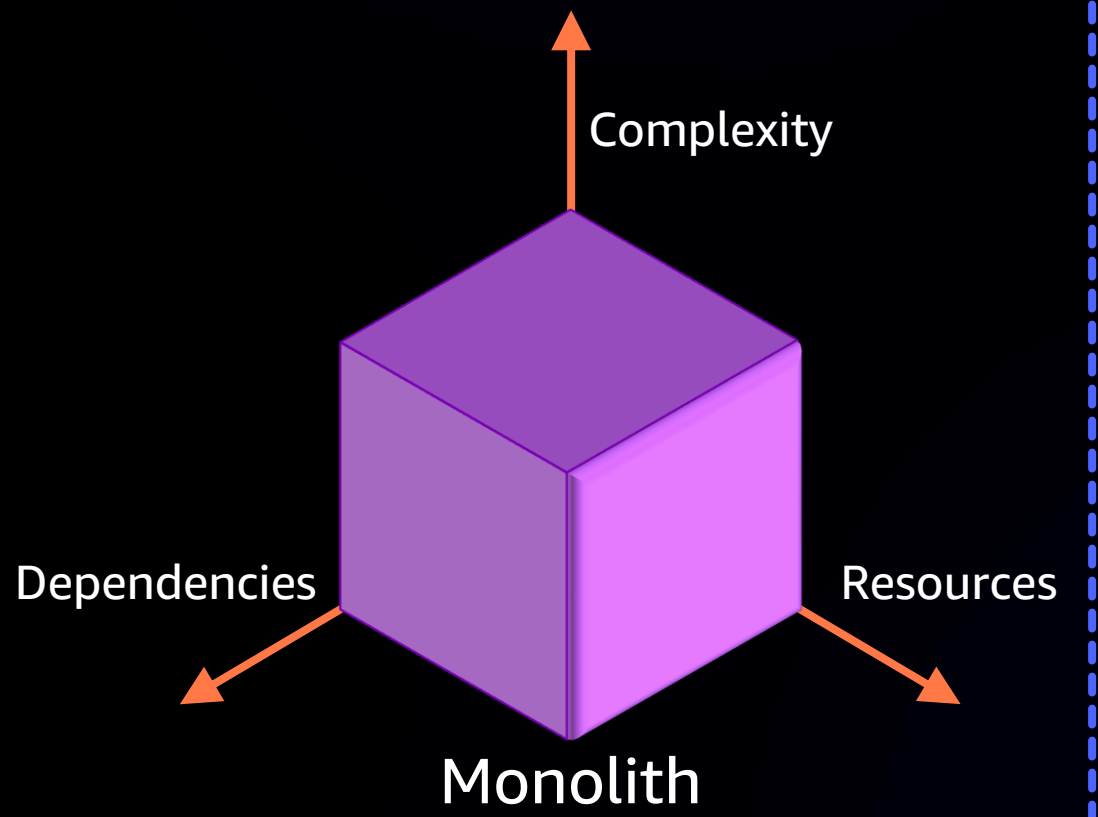
Storage



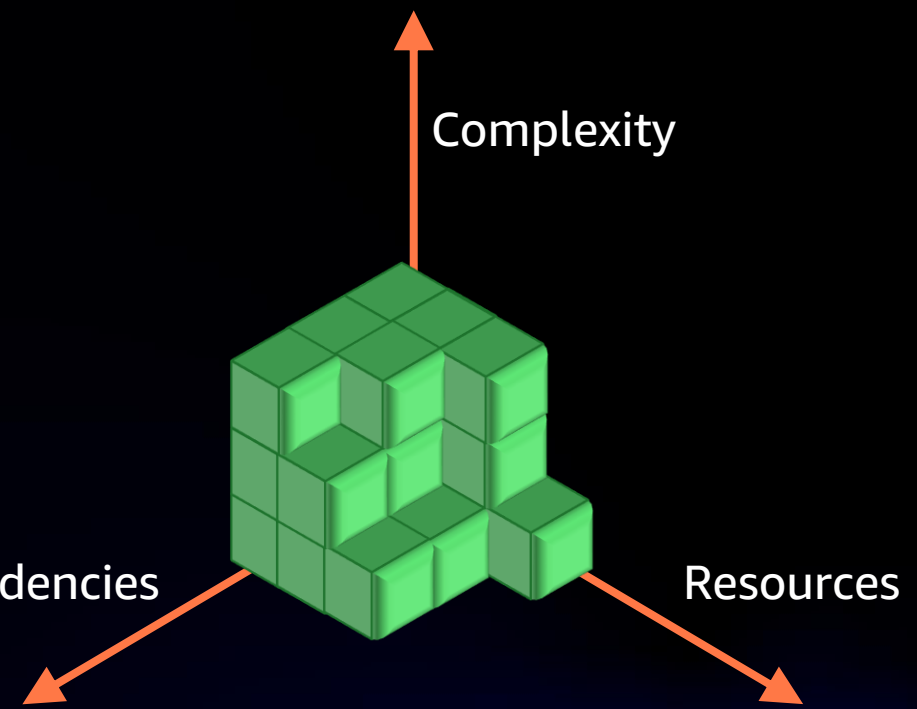
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Finer-grained control

Single responsibility
Increased durability
Reduced risk
Improved security



Monolith



Serverless application

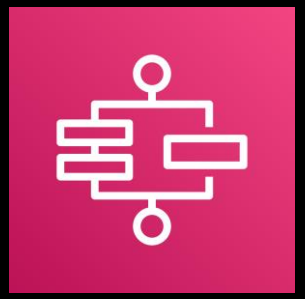
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Orchestration/choreography as configuration

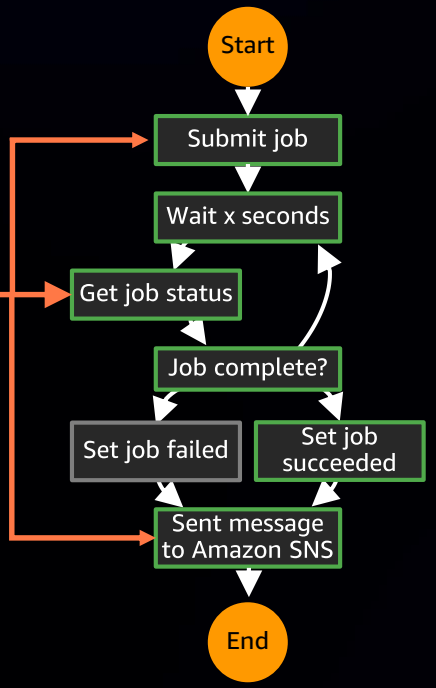
AS YOU GROW AND EXPAND, COORDINATION IS MORE CHALLENGING

Orchestration

Step Functions



AWS Lambda functions

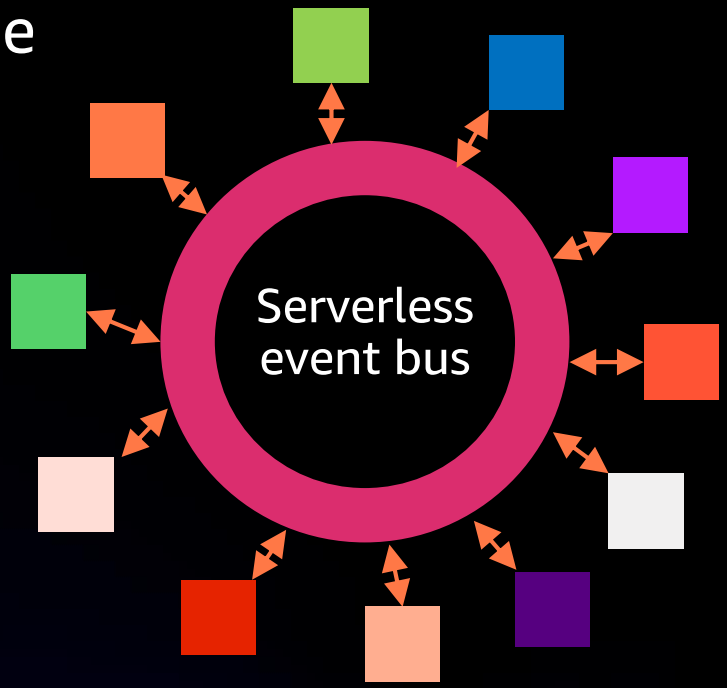


Coordinate the components of distributed applications and microservices using visual workflows

Choreography

EventBridge





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

aws

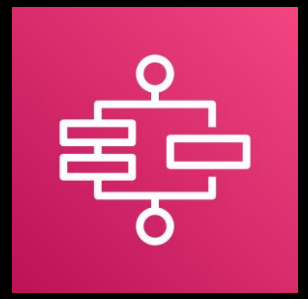
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Orchestration/choreography as configuration

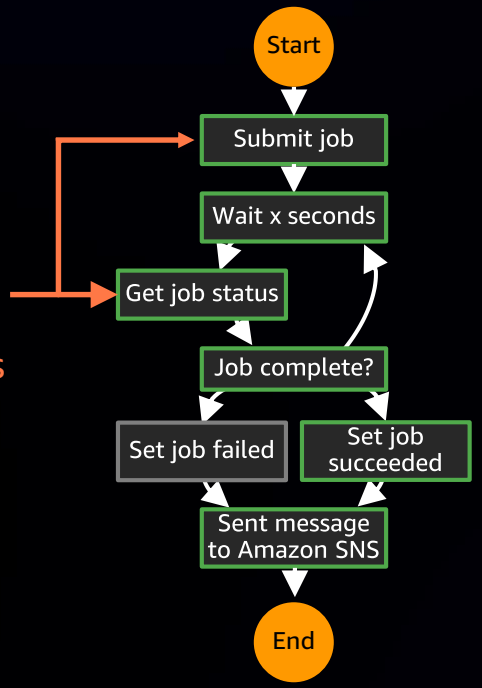
AS YOU GROW AND EXPAND, COORDINATION IS MORE CHALLENGING

Orchestration

Step Functions



AWS Lambda functions

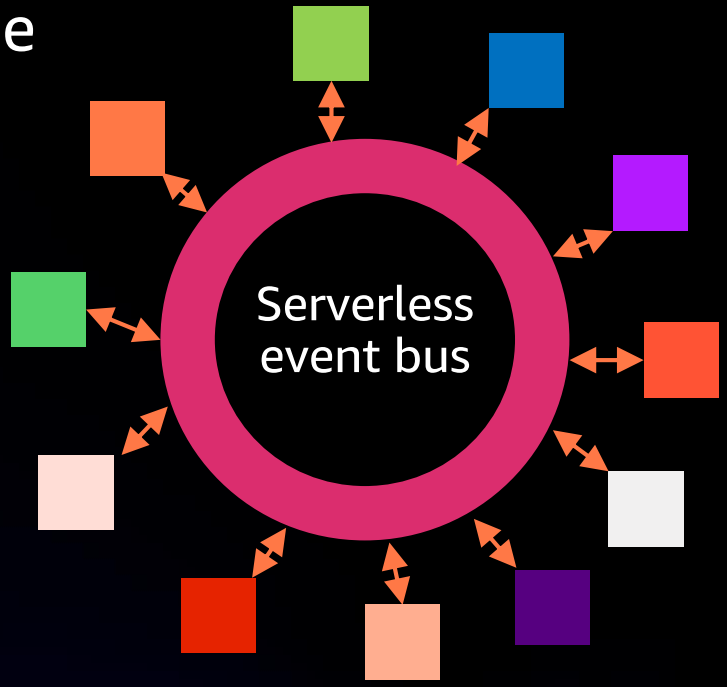


AWS SDK integration

Choreography

EventBridge





API destinations

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Service-full serverless: Best practices

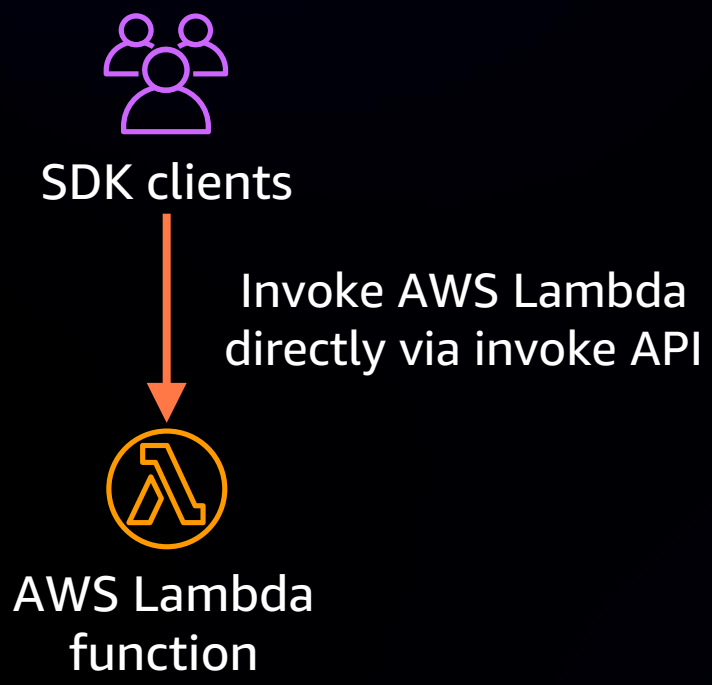
- Use service integrations where possible
- Code is a liability = prefer configuration over code
- Use AWS Lambda to transform, not transport
- Use messaging services
- Avoid monolithic services and functions
- Orchestrate workflows with Step Functions
- Choreograph events with EventBridge

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fabulous functions

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda API



API provided by the AWS Lambda service

Used by all other services that invoke AWS Lambda across all models

Supports sync and async

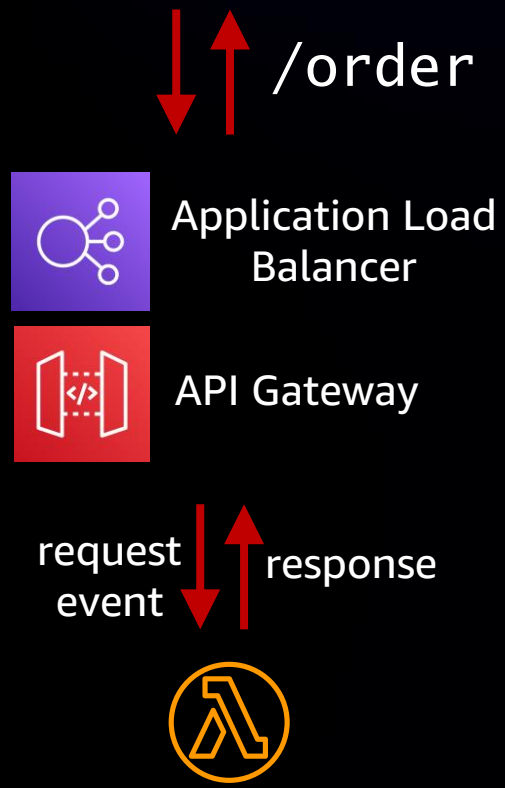
Can pass any event payload structure you want

Client included in every AWS SDK

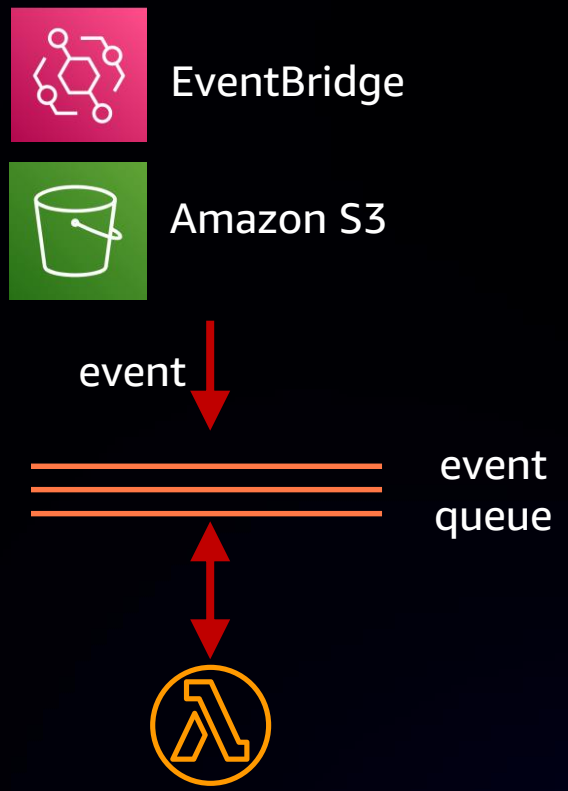
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The AWS Lambda invocation model

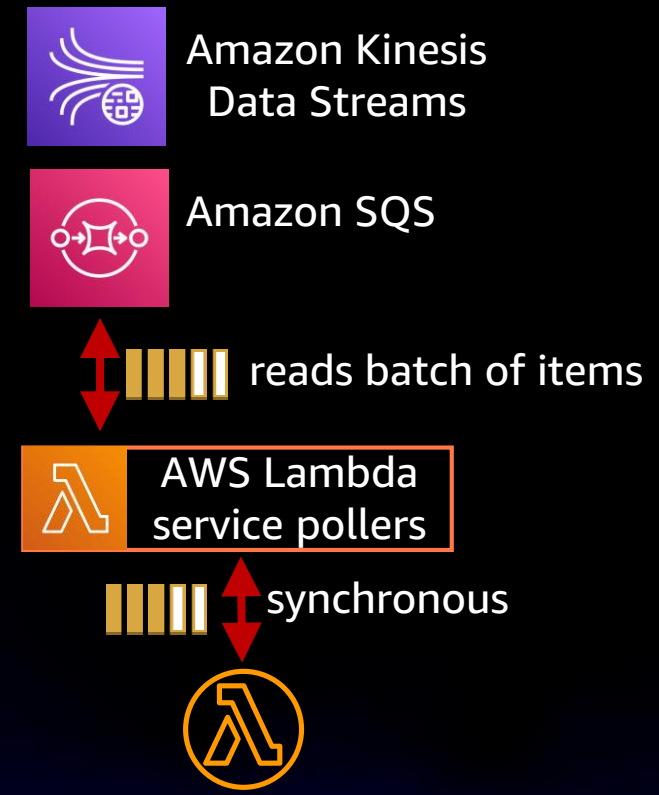
Synchronous
request/response



Asynchronous
event

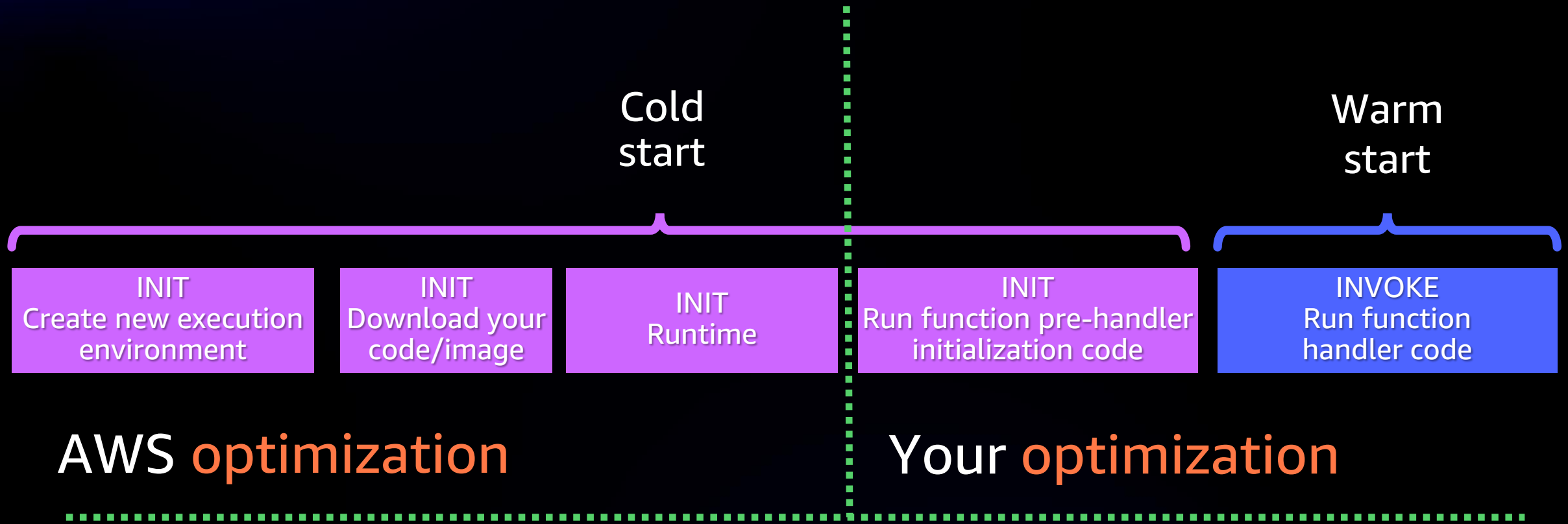


Event source mapping
polls stream/queue



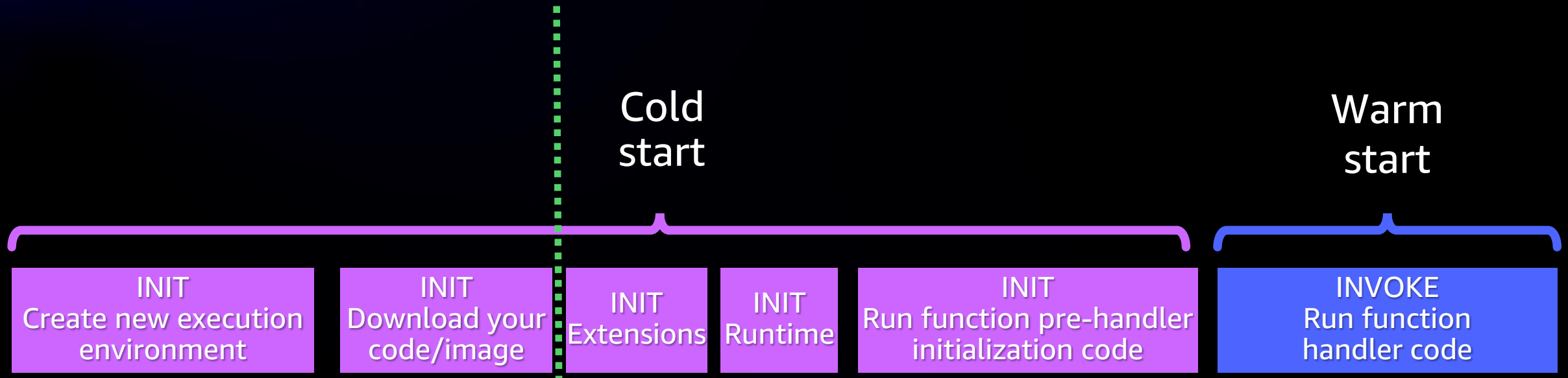
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda execution environment lifecycle



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda execution environment lifecycle



AWS optimization

Your optimization

When using extensions/runtime modifications

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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) {
....
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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
```

Consider using the AWS SDK for JavaScript v3: ~3 MB package rather than 8 MB for v2

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

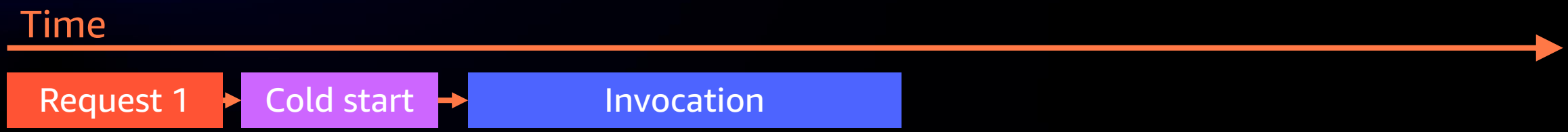
- Regardless of event source or invocation model
- Batches pulled from Kinesis Data Streams, Amazon SQS, or Amazon DynamoDB Streams count as a single event

Concurrent requests require new execution environments to be created

- Limited in concurrency by burst rate per account per Region

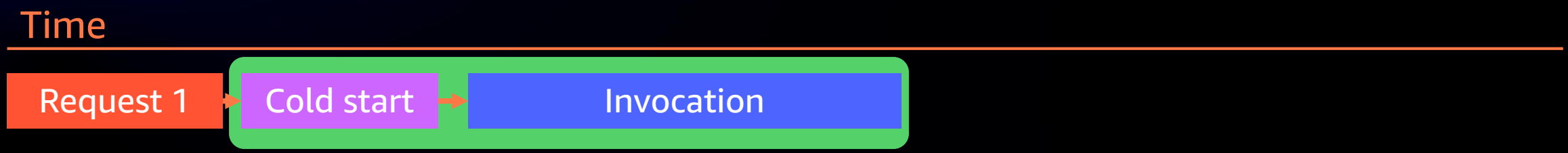
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Understanding concurrency



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Understanding concurrency

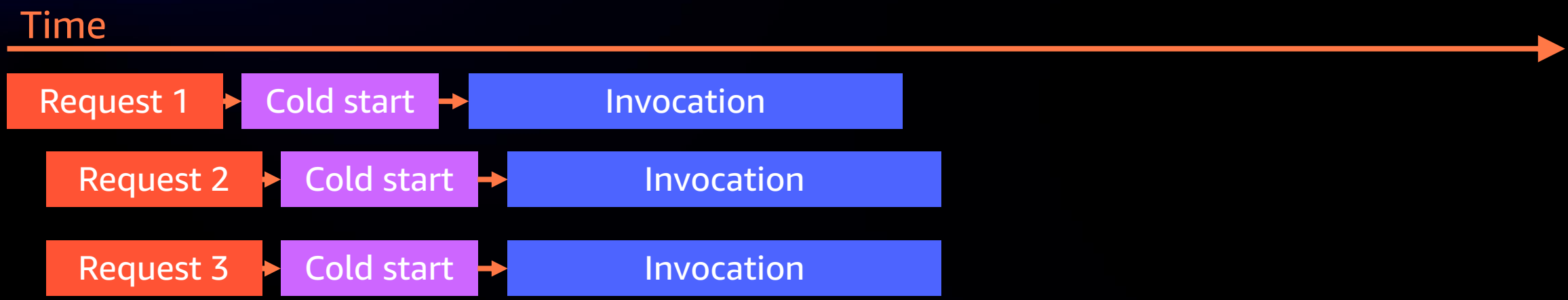


This execution environment is blocked for this entire time

aws

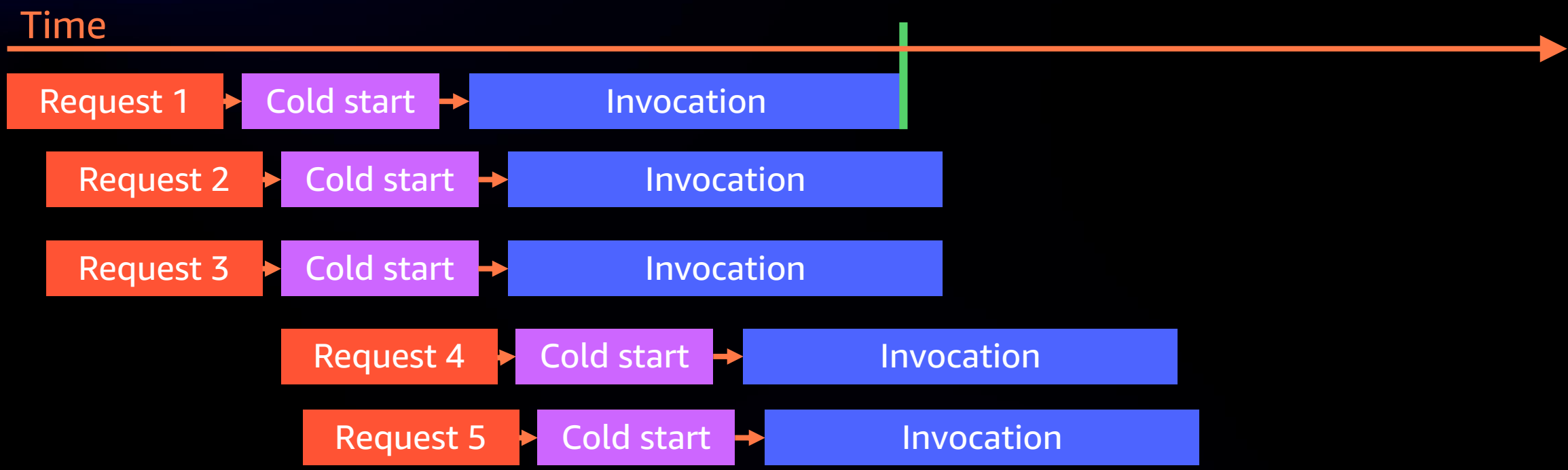
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Understanding concurrency



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

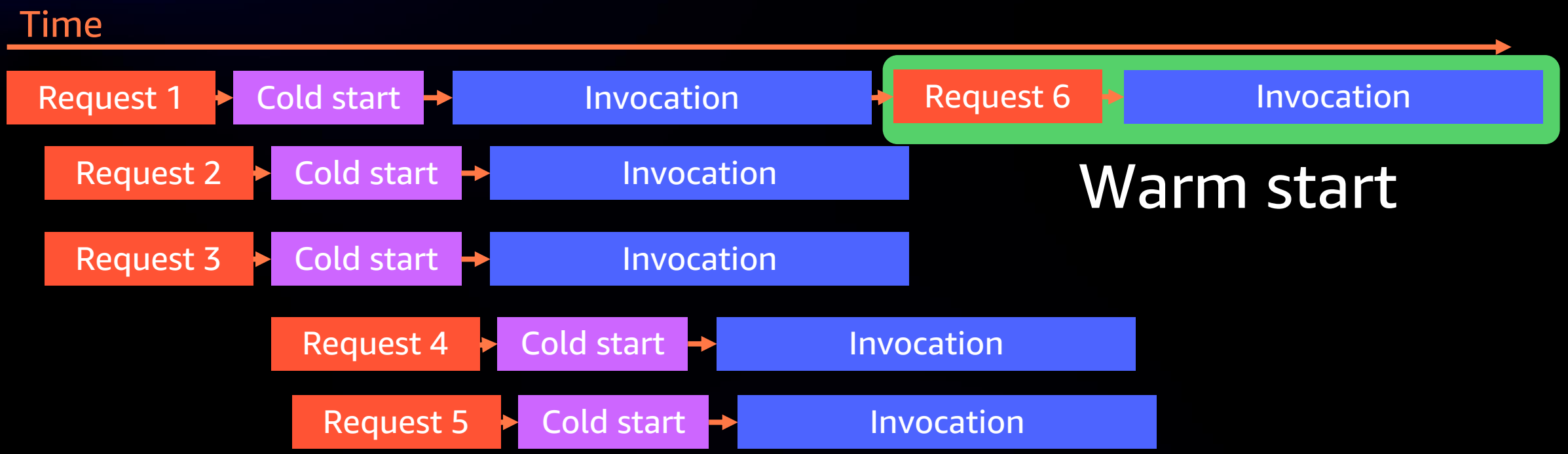
Understanding concurrency



The first environment is now free and warm

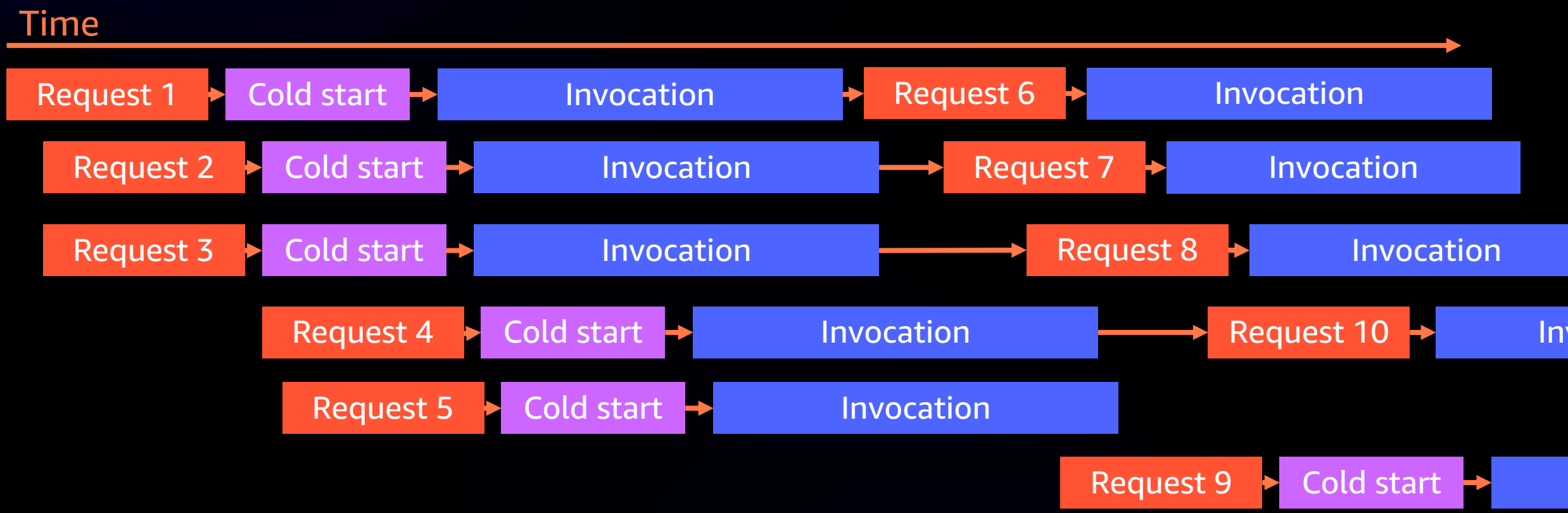
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Understanding concurrency



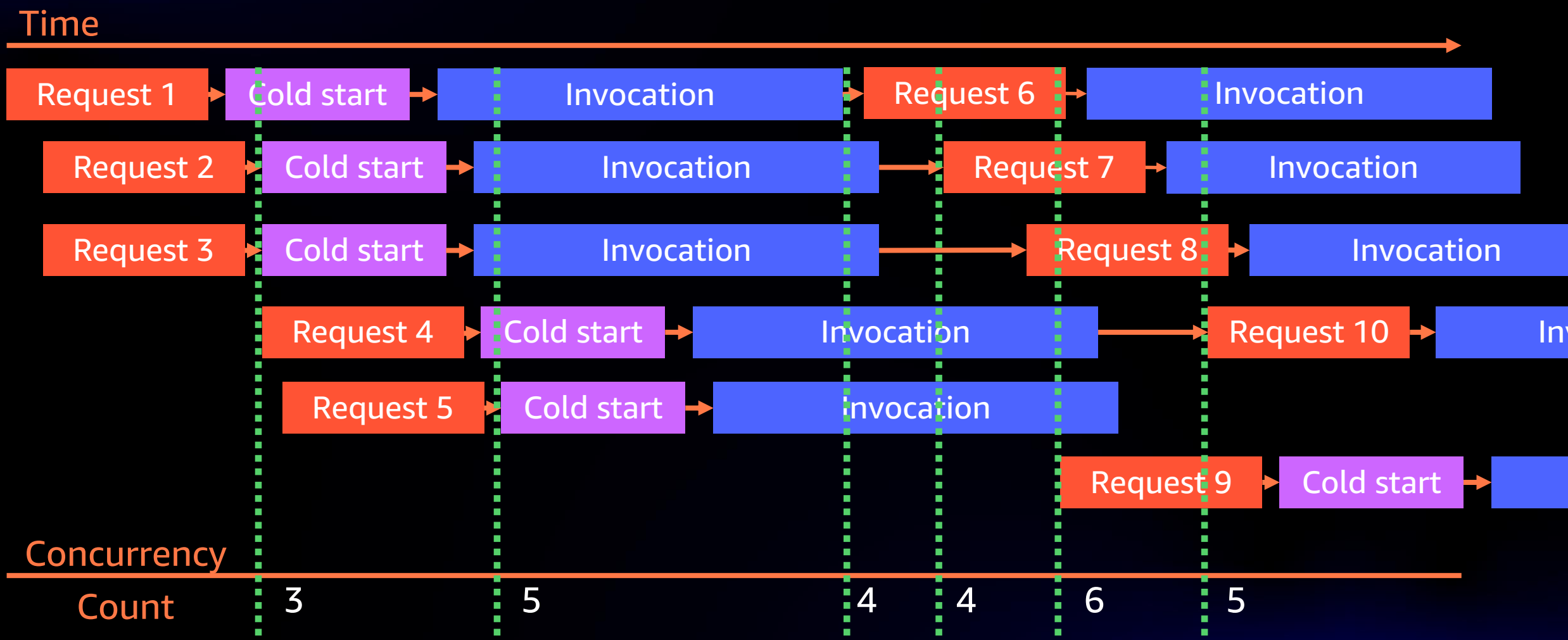
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Understanding concurrency



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Understanding concurrency



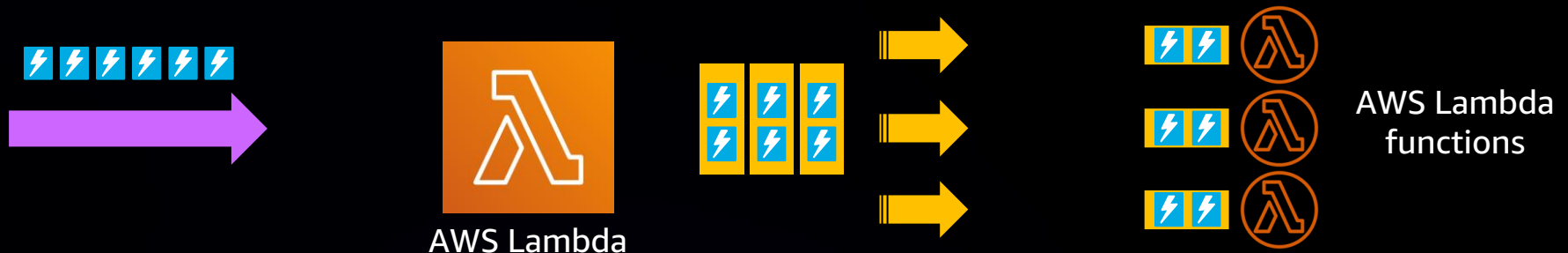
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Concurrency across invocation models

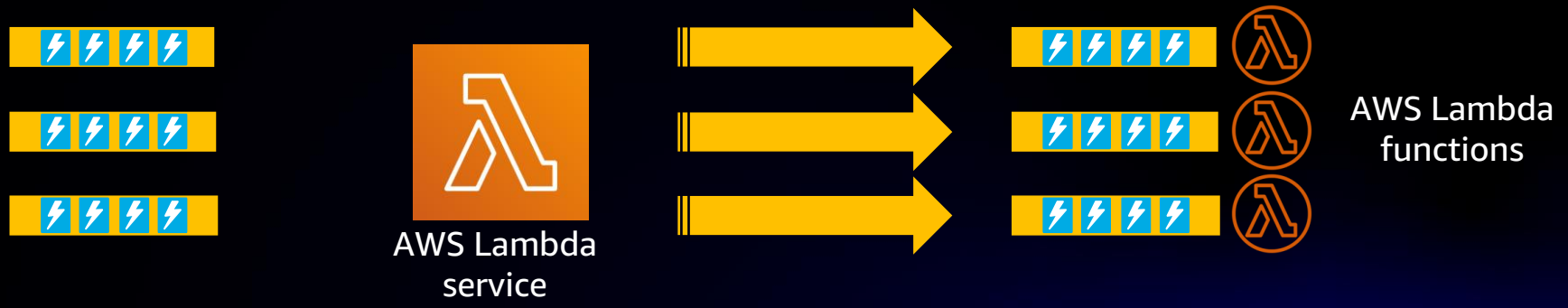
Synchronous
Asynchronous



Event source mapping
Queues



Event source mapping
Streams

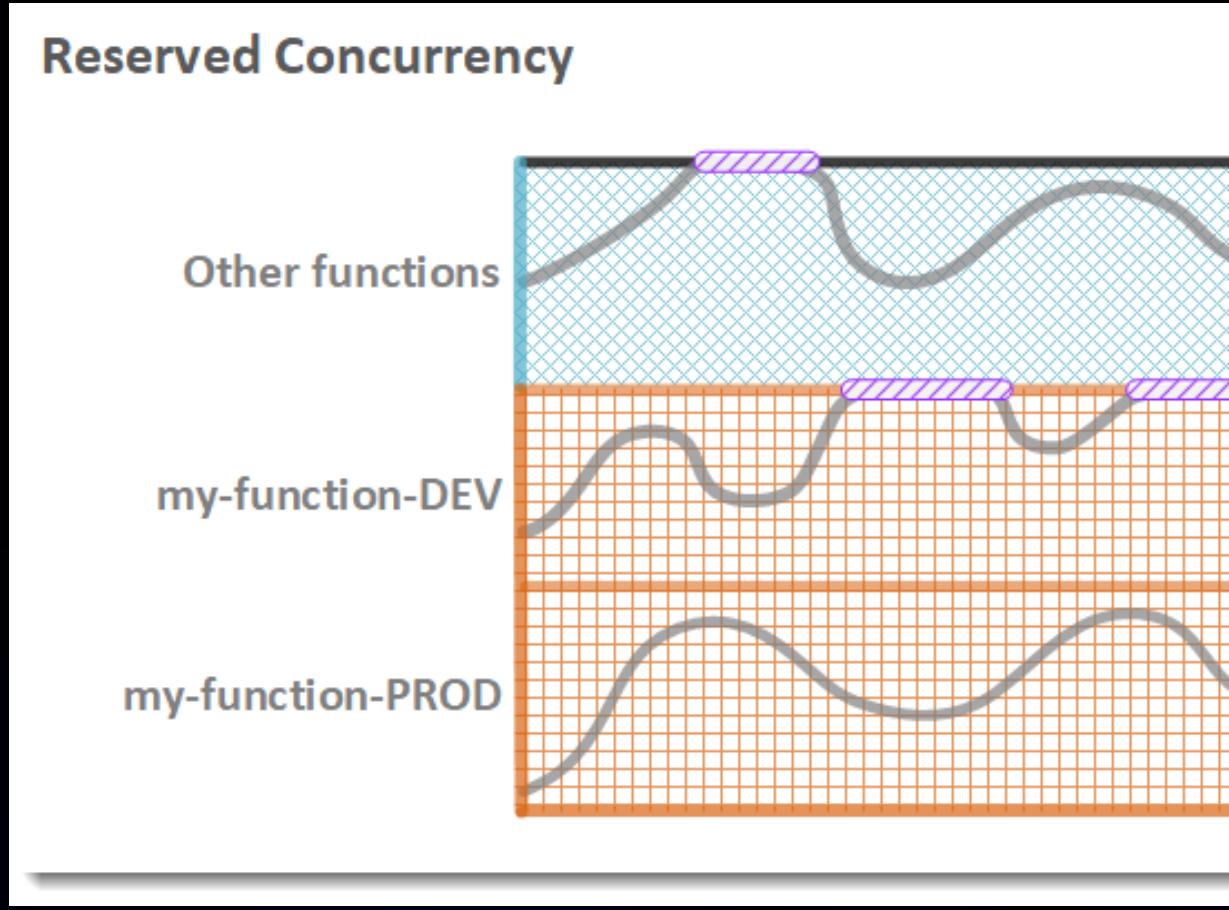


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda concurrency controls

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

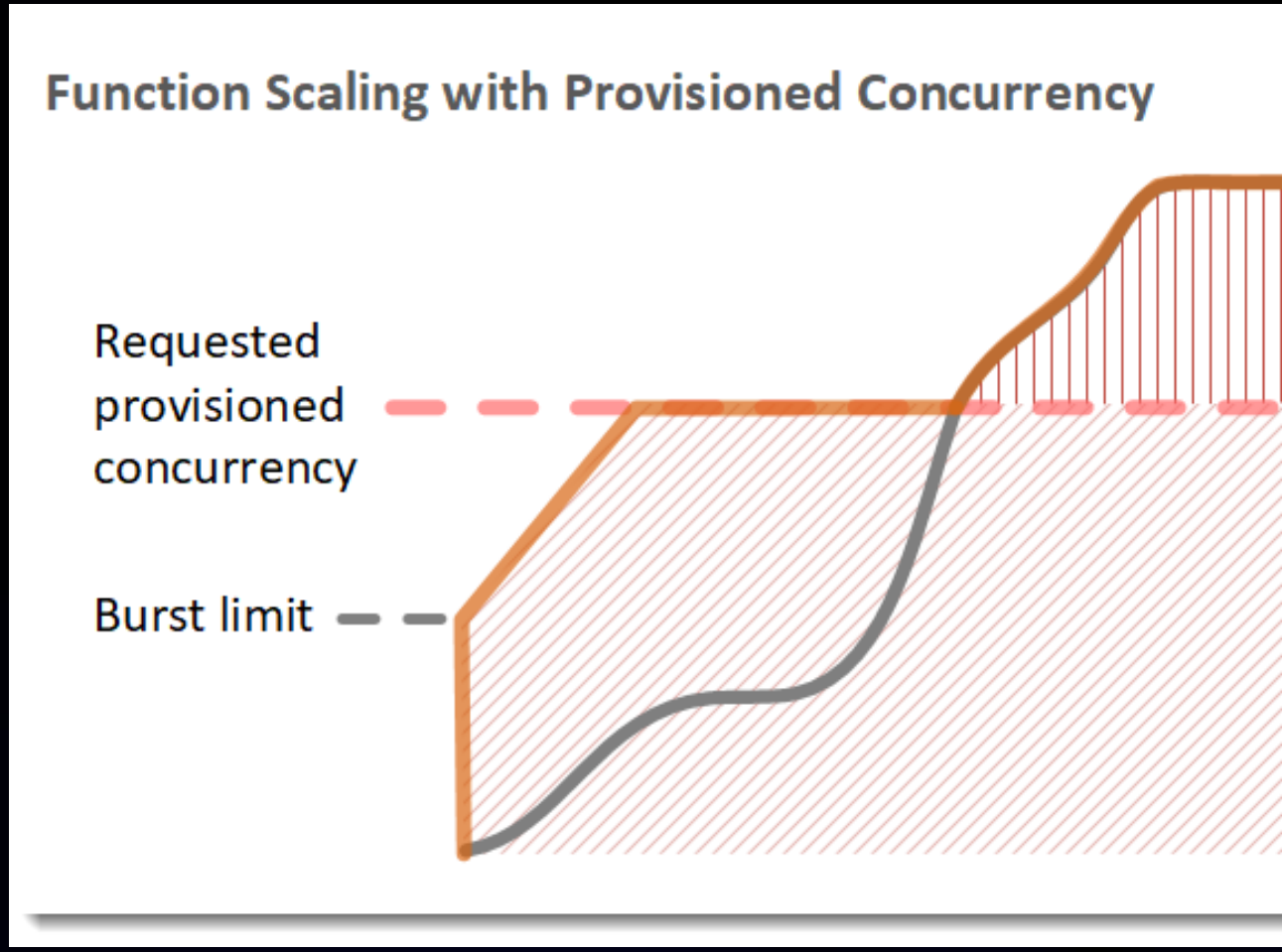


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda concurrency controls

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



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda functions on AWS Graviton2

ARM-based AWS Graviton2 processors =
up to **34% better price-performance** over x86-based AWS Lambda



Highest performance
vs. x86



20% lower cost
vs. same-sized
AWS Lambda functions



Up to 34% better price-performance

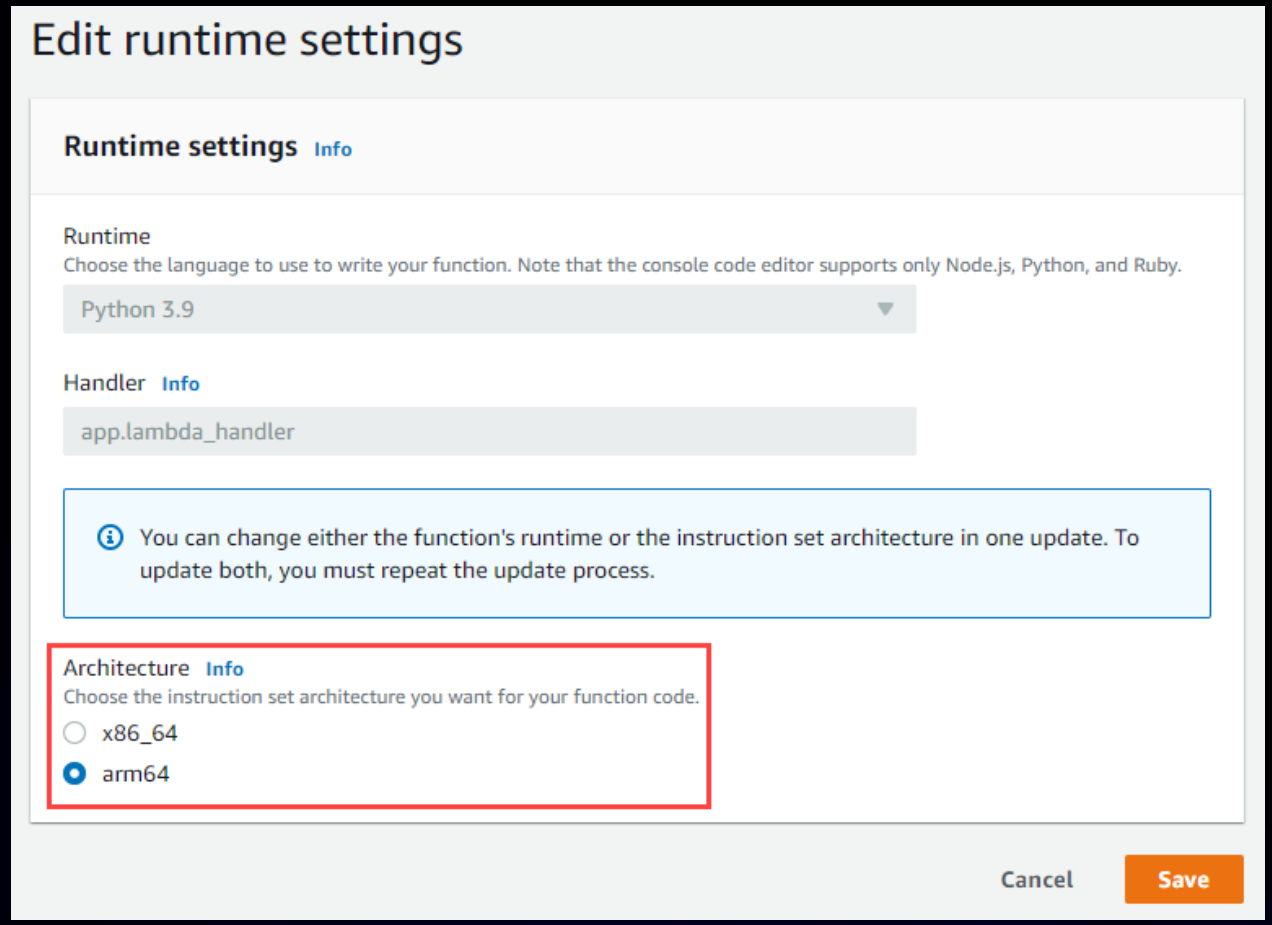
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda functions on arm64

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



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

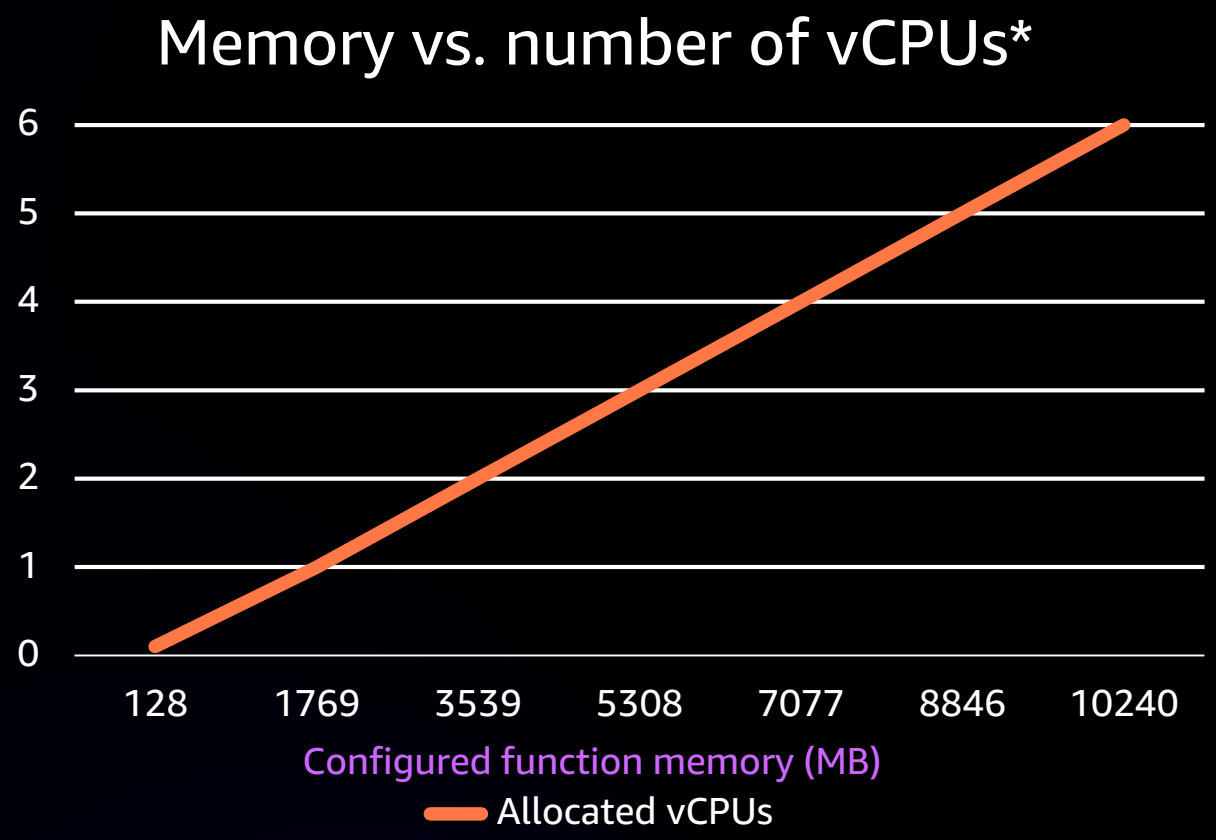
Run memory intensive workloads

- 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



*Numbers are an approximation of vCPU power

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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	11.72965 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

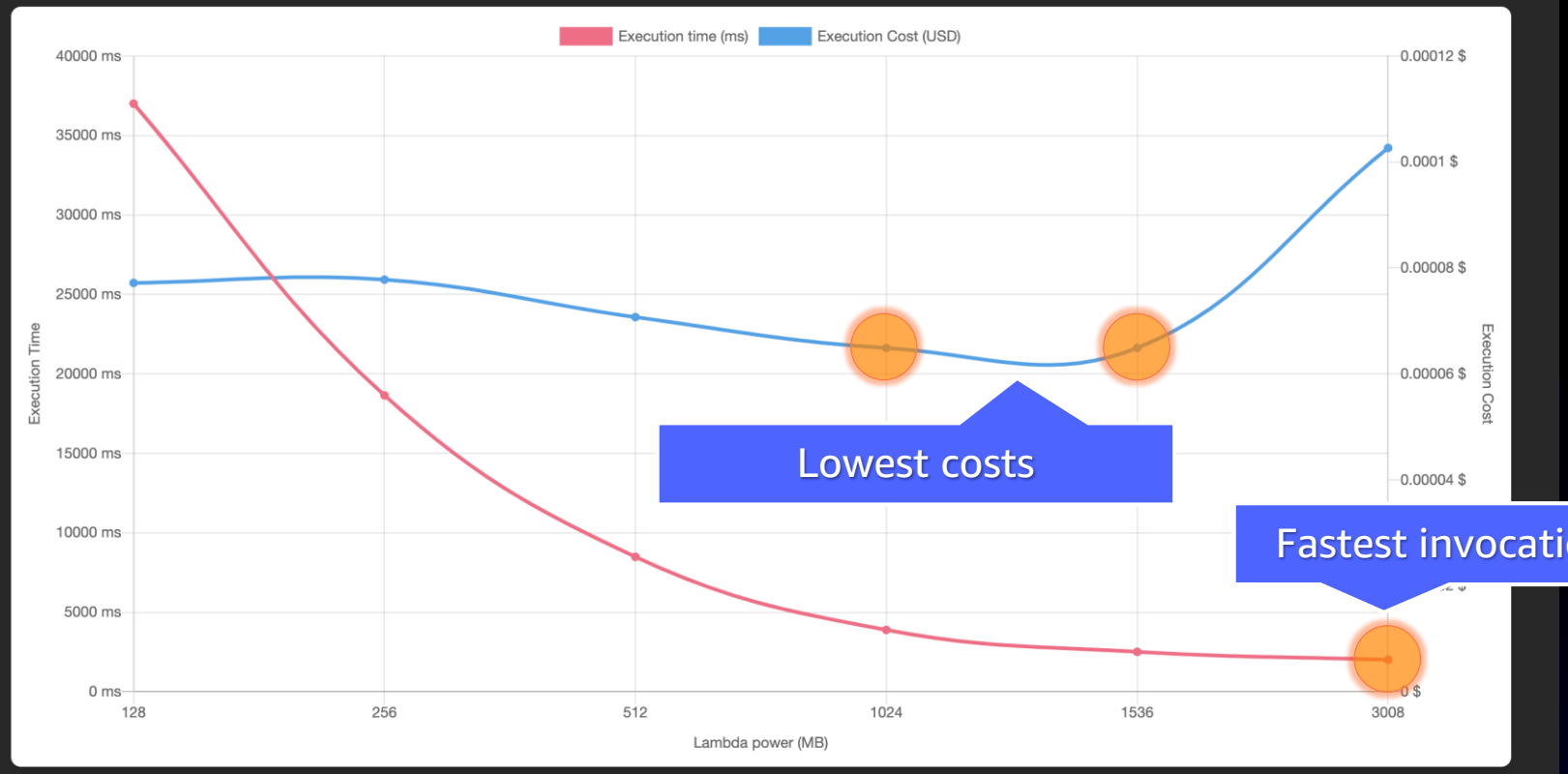
+$0.00001

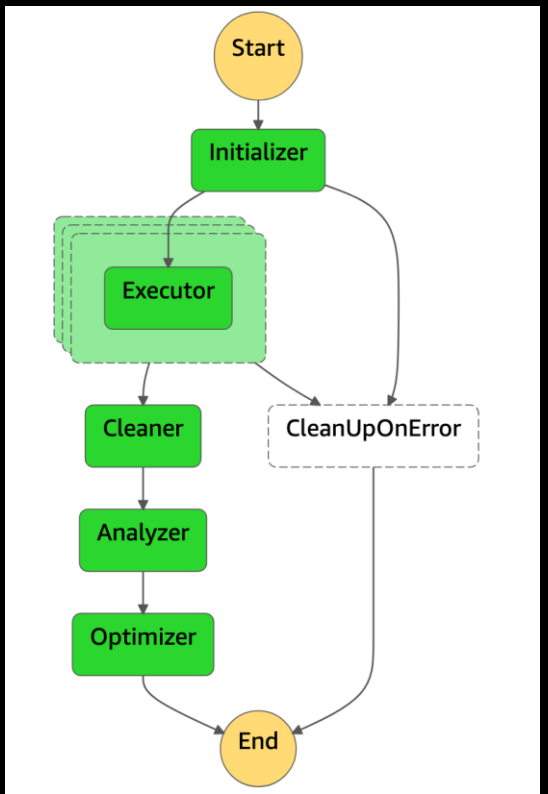
Green = Best **Red** = Worst

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda Power Tuning

DATA-DRIVEN COST AND PERFORMANCE OPTIMIZATION FOR AWS LAMBDA

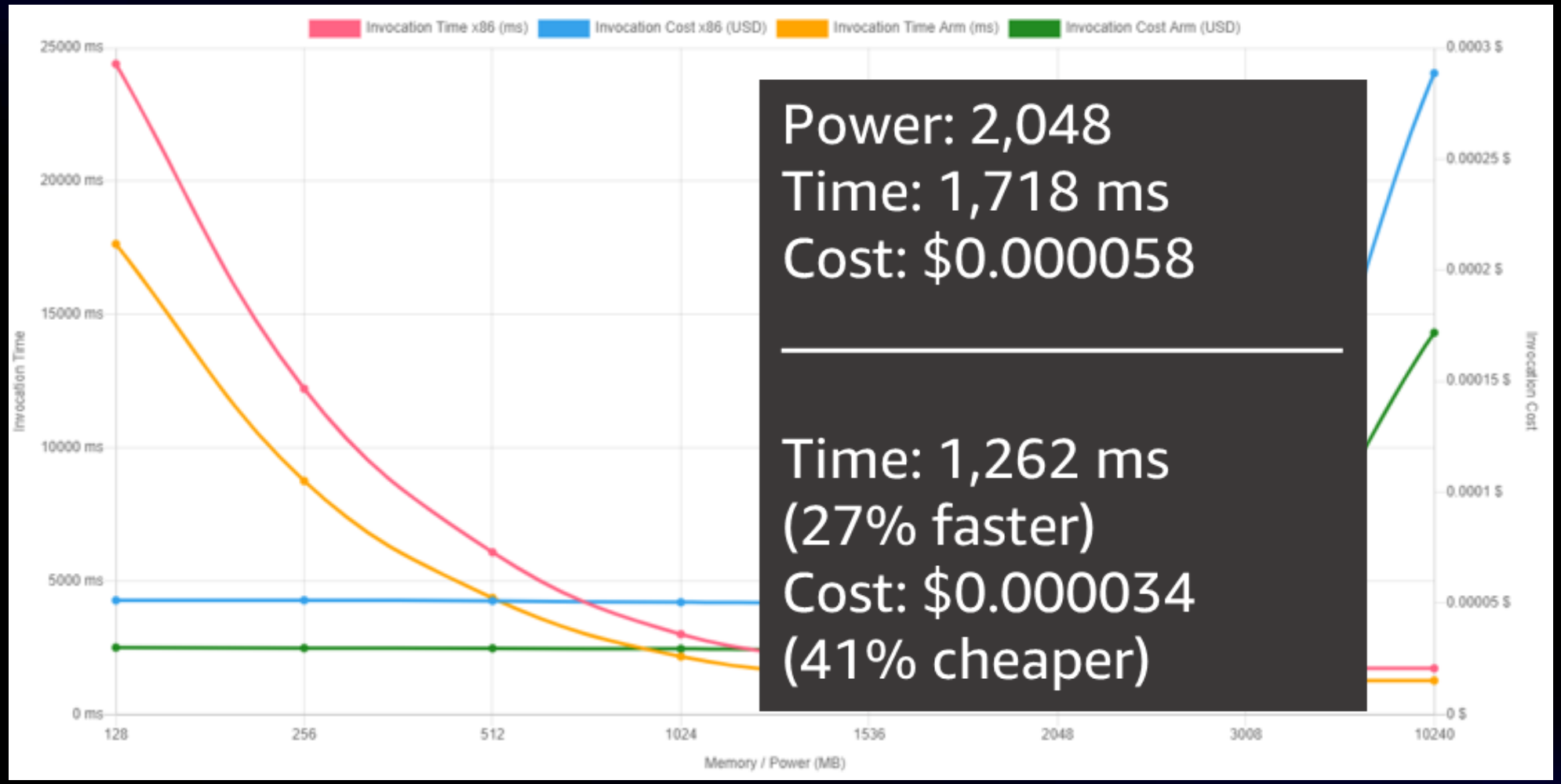




https://github.com/alexcasalboni/aws-lambda-power-tuning

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Comparing x86 to ARM/AWS Graviton2



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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
- Understand concurrency and quotas, how to reserve and provision
- Try out ARM/AWS Graviton2
- More memory = proportionally more CPU and I/O

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fabulous functions: Best practices

- Use different invocation models (async → sync)
- Understand concurrency and quotas, how to r
- 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

Optimizing AWS Lambda performance and cost for your serverless applications

AWS Online Tech Talks



s12d.com/lambda-performance-cost

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Configuration as code

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Infrastructure as code (IaC)

Build infrastructure using configuration files

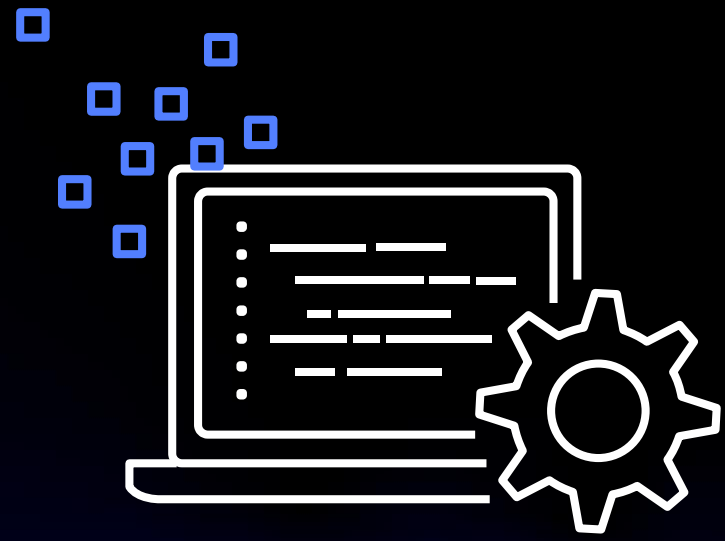
Treat configuration files as software code

With serverless, your infrastructure is your app

Automate the provisioning process

Reduce configuration drift

Deploy to multiple environments and accounts



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

IaC frameworks for serverless



AWS Serverless Application Model (AWS SAM)



AWS Cloud Development Kit (AWS CDK)

serverless

Architect

CHALICE

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless Application Model



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

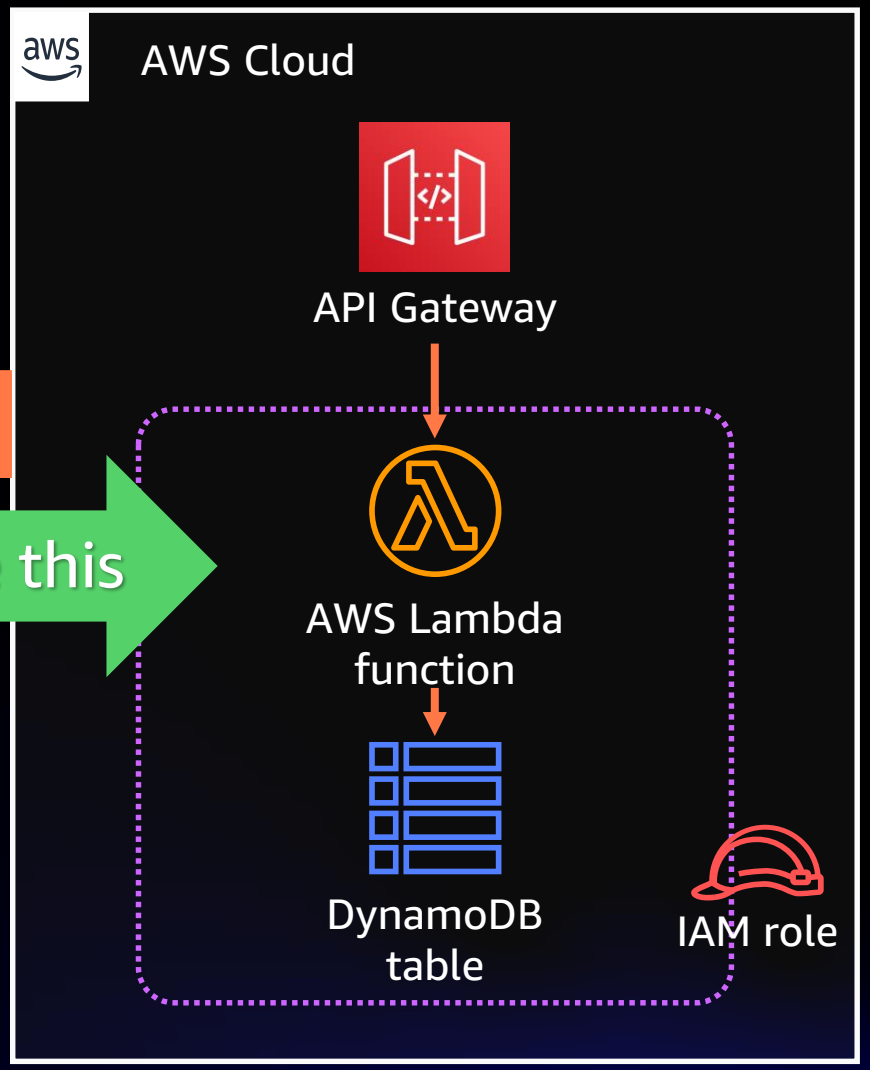
AWS SAM templates

Just 20 lines to create

```yaml
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



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Friends don’t let friends

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS SAM policy templates

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



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS SAM IAM policy templates

75+

Managed templates
with more being added

Best practice: 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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless Patterns Collection

s12d.com/patterns

Submit a pattern

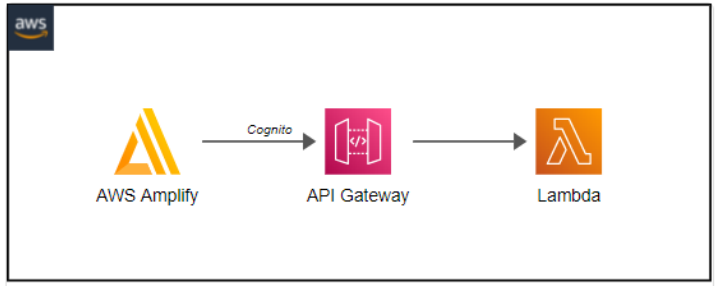
Use serverless patterns to quickly build integrations using AWS SAM and

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

Filters (102 templates) Reset

Services

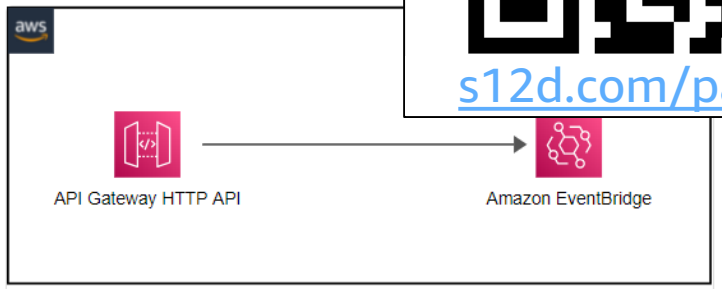
- [ ] AWS Amplify
- [ ] Amazon API Gateway
- [ ] AWS AppSync
- [ ] Amazon CloudFront
- [ ] Amazon Cognito
- [ ] Amazon DynamoDB
- [ ] Amazon EventBridge
- [ ] AWS Fargate
- [ ] AWS IoT
- [ ] Amazon Kinesis
- [ ] AWS Lambda



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

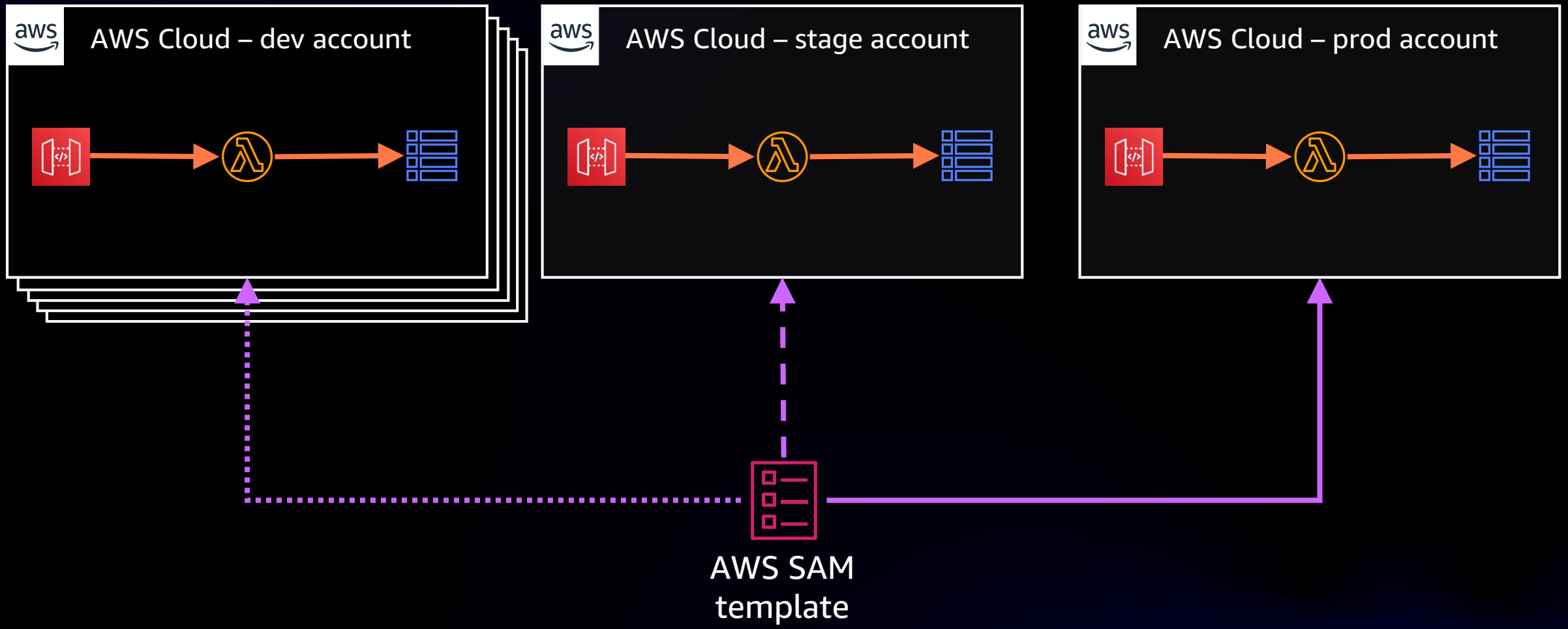
CDK View pattern





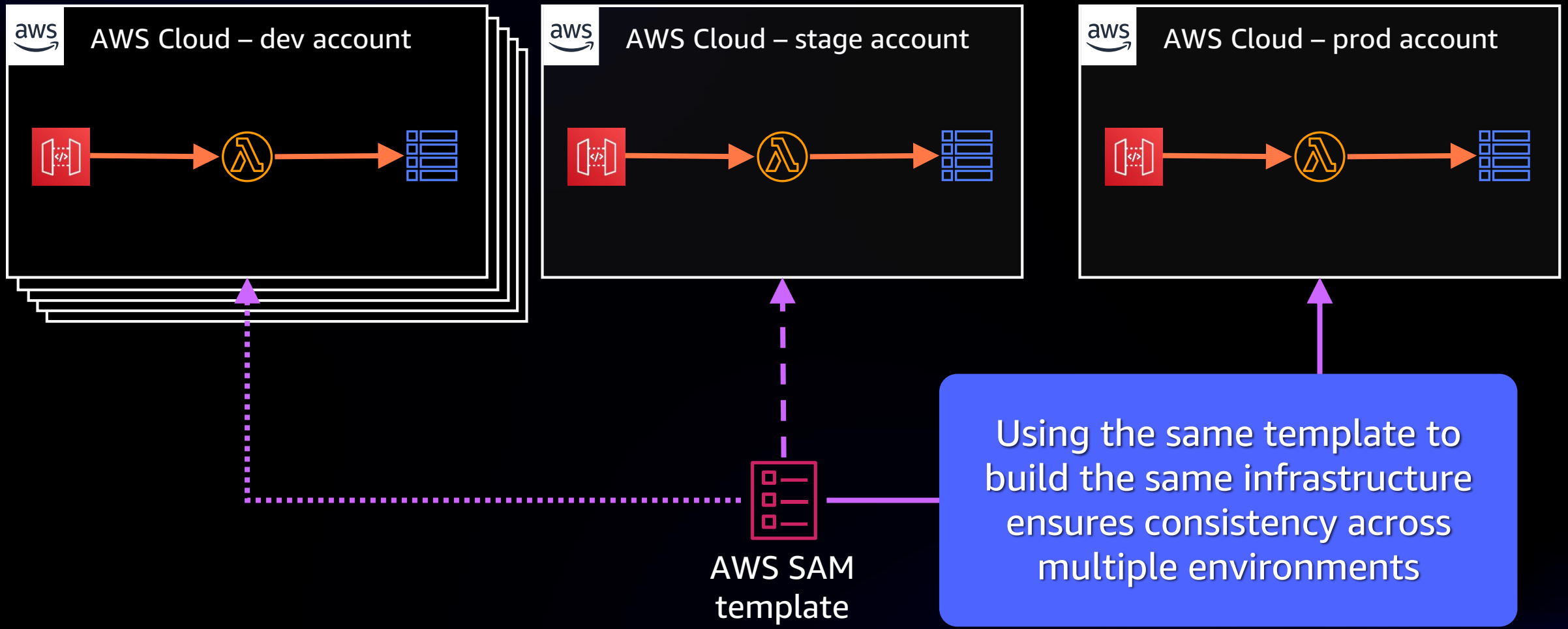
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reusable templates



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reusable templates



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Making a template reusable

USER DATA AND CONFIGURATION

Added at creation or update

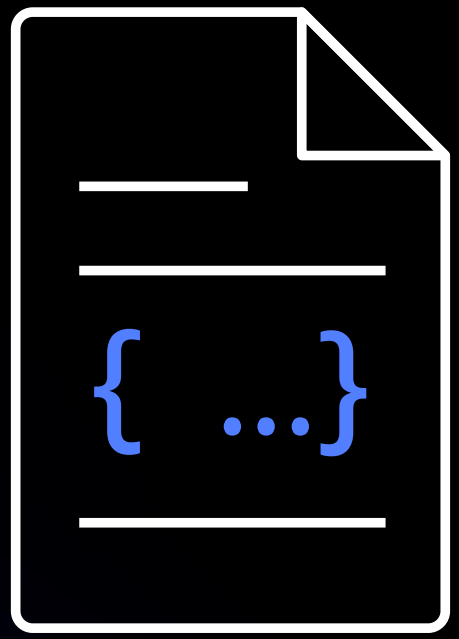
- Template parameters
- Environment parameters

Added at creation, update, or runtime

- AWS Systems Manager Parameter Store
- AWS Secrets Manager

Added at runtime

- AWS AppConfig



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless service discovery

Name: /serverlespresso/{Environment}/{Service}/{Name}/{Type}

AWS SAM template: core service

```
Resources:
  ServerlesspressoEventBus:
    Type: AWS::Events::EventBus
    Properties:
      Name: "Serverlesspresso"

  CoreEventBusNameParameter:
    Type: "AWS::SSM::Parameter"
    Properties:
      Name: !Sub /${AppName}/${Environment}/${Service}/eventbusname/name
      Description: EventBus Name
      Type: String
      Value: !Ref ServerlesspressoEventBus
```



AWS Systems Manager Parameter Store

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless service discovery

Name: /serverlespresso/prod/core/eventbus/name
Value: serverlespresso-core-eventbus

AWS SAM template: order-manager service

```
Parameters:
  CoreEventBusName:
    Type: AWS::SSM::Parameter::Value<String>
    Description: Event bus used by application

Resources:
  CreateOrderFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: code/CreateOrder
      Environment:
        Variables:
          BusName: !Ref CoreEventBusName
```



AWS Systems Manager Parameter Store

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless service discovery

My parameters

Name	Tier	Type
/Serverlesspresso/dev/capacity/IsCapacityAvailableFunction/name	Standard	String
/Serverlesspresso/dev/config/CloudFrontDistribution/url	Standard	String
/Serverlesspresso/dev/core/eventbus/name	Standard	String
/Serverlesspresso/dev/order-processor/statemachine/arn	Standard	String
/Serverlesspresso/prod/capacity/IsCapacityAvailableFunction/name	Standard	String
/Serverlesspresso/prod/config/CloudFrontDistribution/url	Standard	String
/Serverlesspresso/prod/core/eventbus/name	Standard	String
/Serverlesspresso/prod/order-processor/statemachine/arn	Standard	String

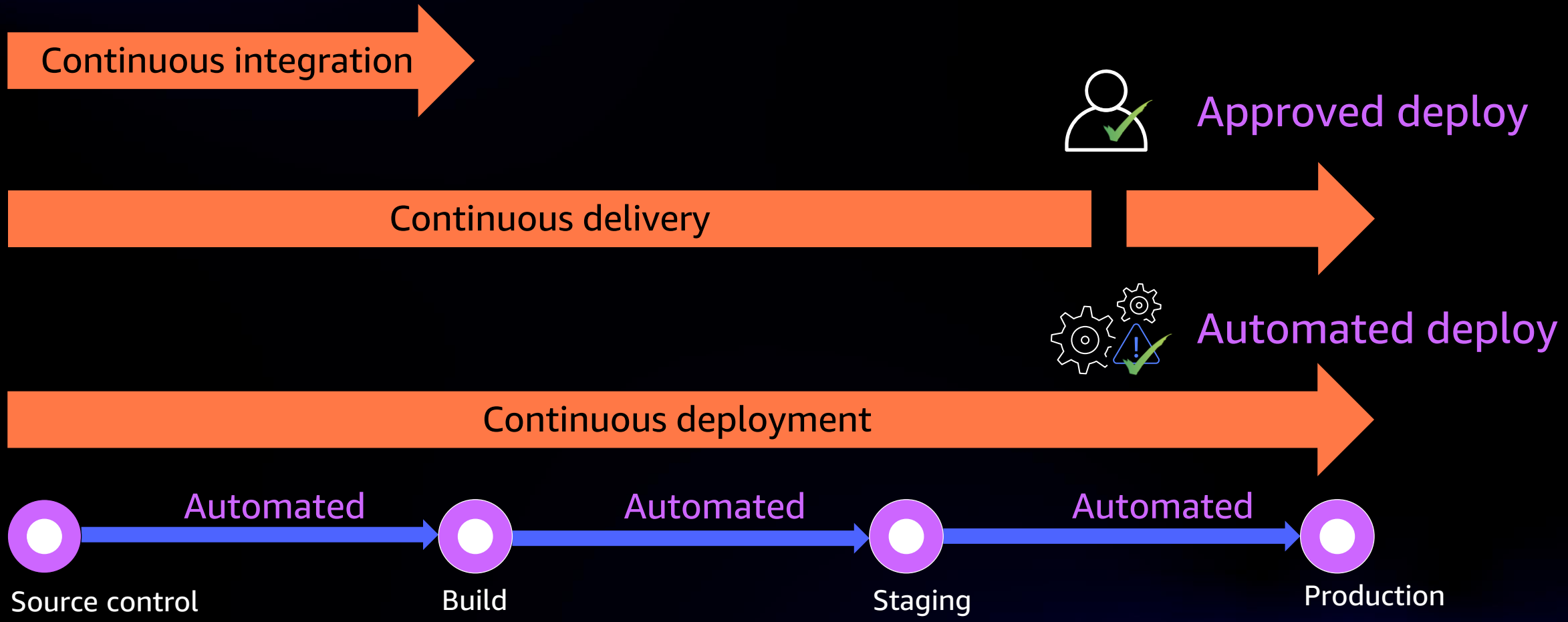


AWS Systems Manager Parameter Store

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

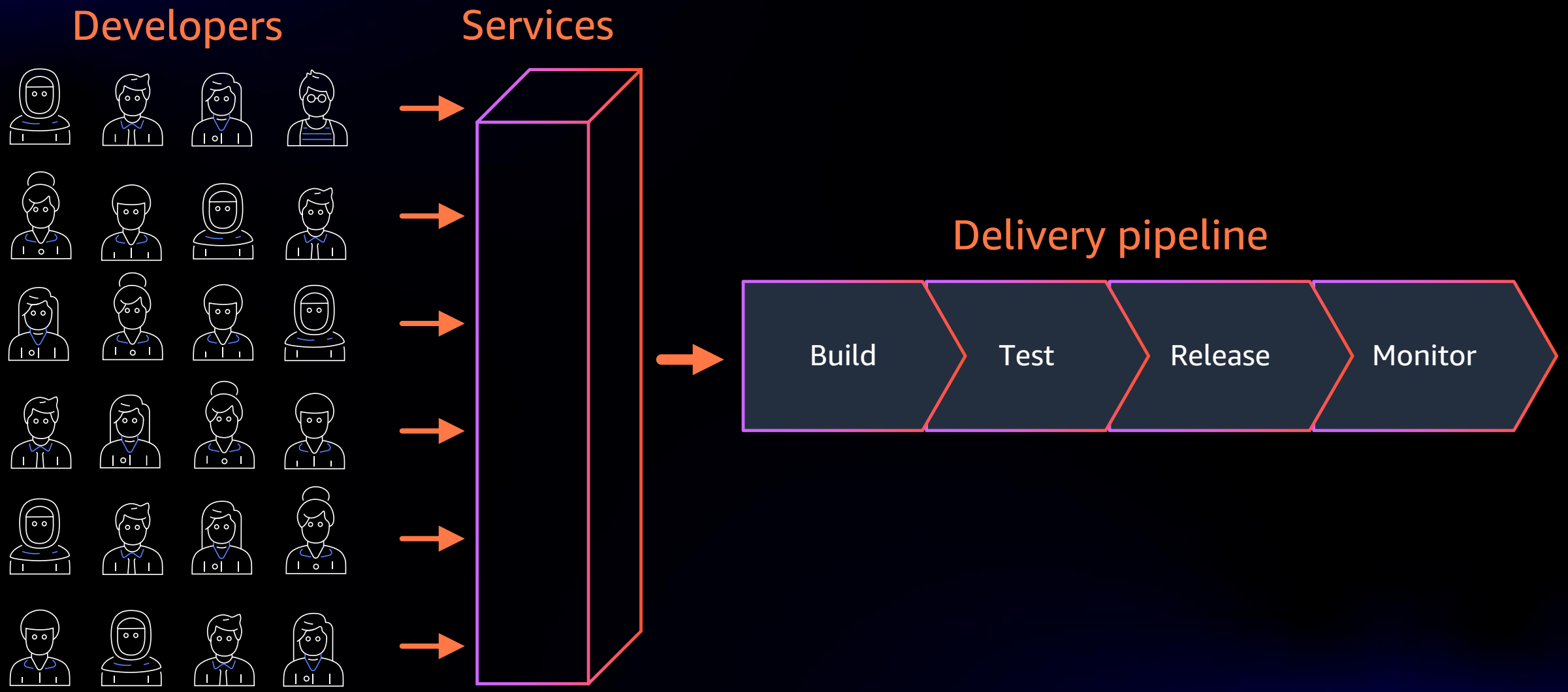
The CI/CD pipeline

CONTINUOUS INTEGRATION, DELIVERY, AND DEPLOYMENT PIPELINE



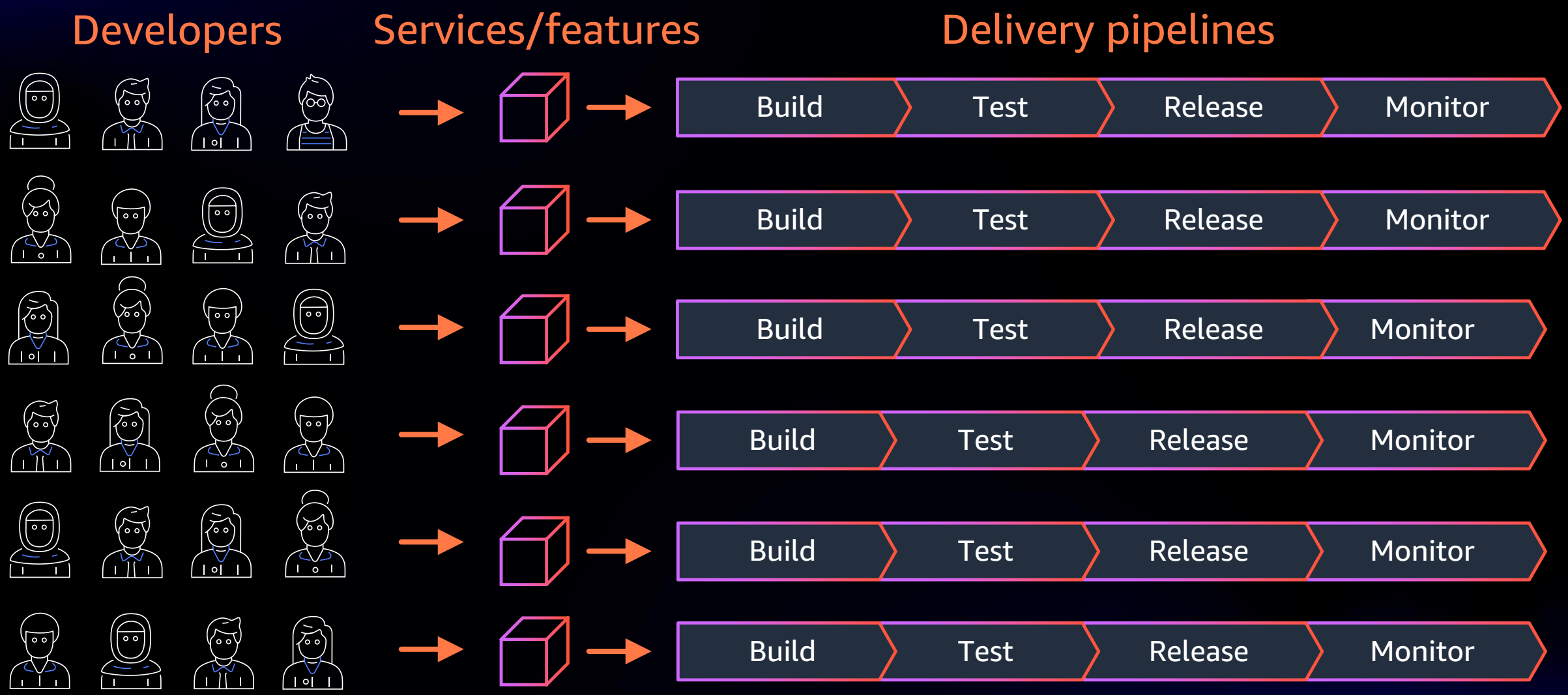
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Single development pipeline



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Multiple development pipelines



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS SAM pipelines

`sam pipeline init --bootstrap`

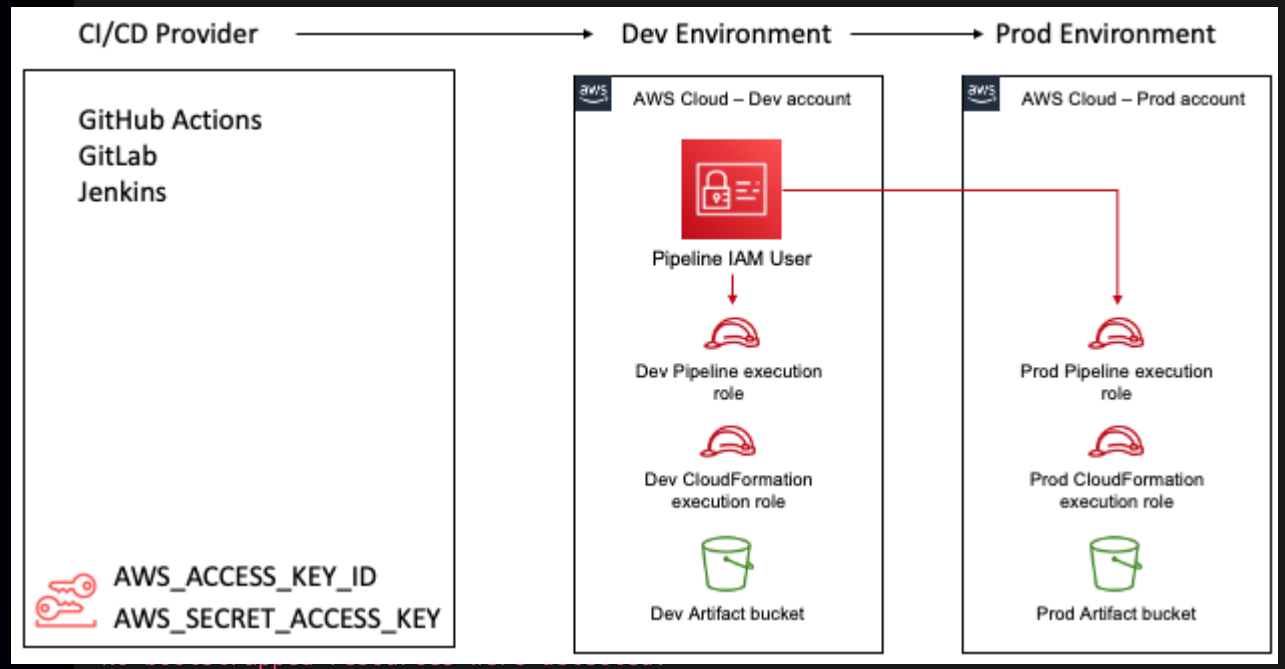
- Creates the AWS resources and permissions required to deploy application artifacts from your code repository into your AWS environments
- Walks you through setting up a CI/CD pipeline with AWS CodePipeline, Bitbucket, Gitlab, GitHub Actions, or Jenkins.

https://s12d.com/sam-pipelines

```
$ sam pipeline init --bootstrap

sam pipeline init generates a pipeline configuration file that you can use to
connect your
AWS account(s) to your CI/CD system. We will guide you through the process to
bootstrap resources for each stage, then walk through the details necessary for
creating the pipeline config file.

Please ensure you are in the root folder of your SAM application before you begin.
```




```
Do you want to go through stage setup process now? If you choose no, you can still
reference other bootstrapped resources. [y/N]: y

For each stage, we will ask for [1] stage definition, [2] account details, and [3]
reference application build resources in order to bootstrap these pipeline
resources.

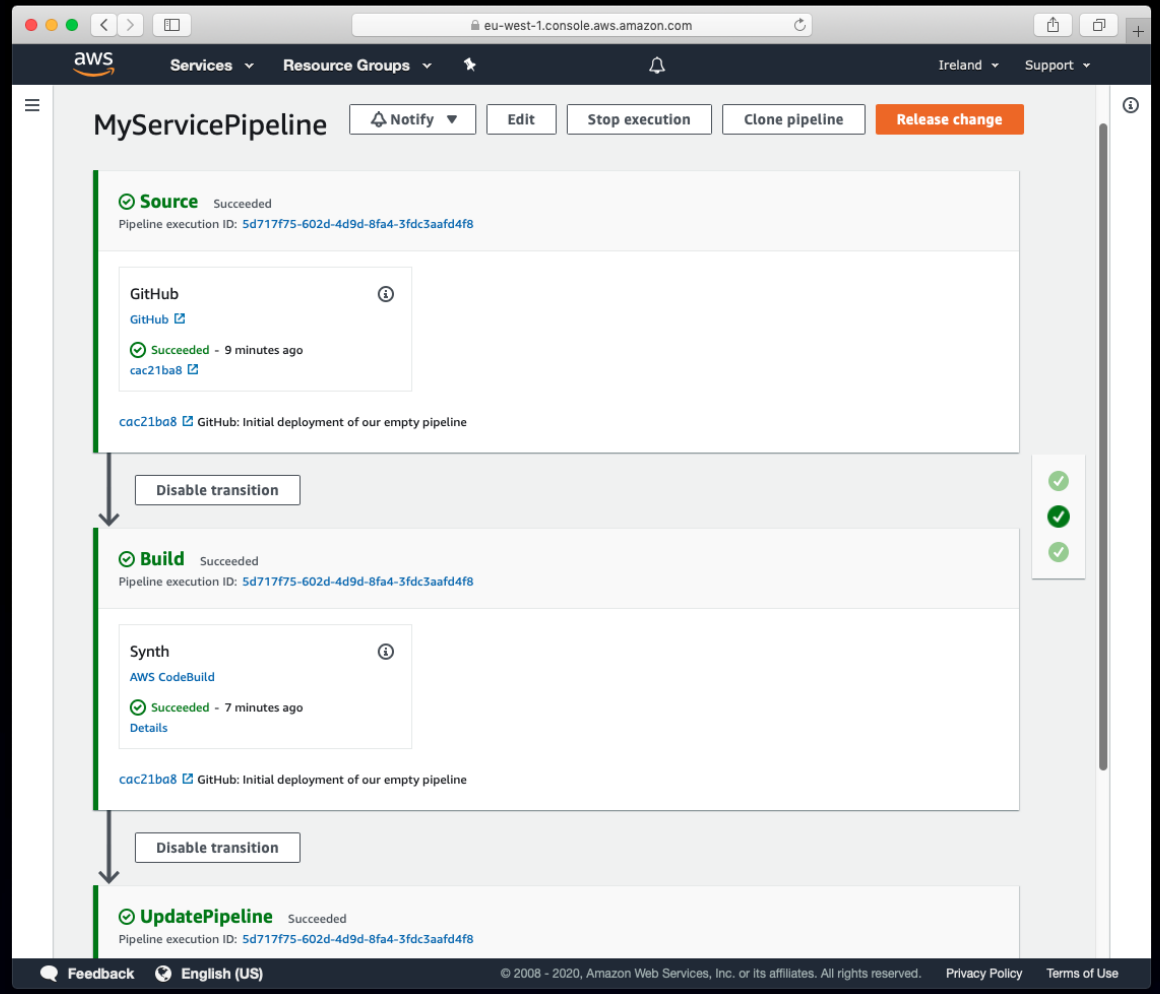
We recommend using an individual AWS account profiles for each stage in your
pipeline. You can set these profiles up using [little bit of info on how to do
this/docs].
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

https://s12d.com/cdk-pipelines



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Configuration as code: Best practices

- Use a framework!
- Single codebase in version control for all environments
- Stages isolated in separate accounts/environments
- Prototype new features using temporary environments
- Use stack parameters/service discovery for reusability
- Build/deploy/test on each commit
- Use a rollout “humanless” deployment mechanism
- Multiple stacks: split mutable and immutable resources

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Healthy serverless

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Retry/failure handling

Service	Retry/failure capabilities
AWS Lambda API	Retry/failure logic is client-dependent for synchronous invocations. Asynchronous invocations are retried twice by AWS Lambda (configurable between 0 and 2. Maximum event age 60 seconds – 6 hours).
Amazon SNS	If AWS Lambda is not available, Amazon SNS retries 3 times without delay, 2 times at 1 second apart, then 10 times with exponential backoff from 1 second to 20 seconds, and finally 100,000 times every 20 seconds, for a total 100,015 attempts over more than 23 days before the message is discarded.
Amazon EventBridge	Retry with exponential backoff for up to 24 hours for a specific invocation, except in scenarios where your target resource is constrained
Amazon SQS	Messages remain in the queue until deleted and are prevented from being accessed by other consumers during a period of time known as the “visibility timeout.” Successful AWS Lambda invocations will cause deletions of messages automatically. If an invocation fails or doesn’t delete a message during the visibility timeout window, it is made available again for other consumers.
Amazon Kinesis Data Streams	When using the Amazon Kinesis Client Library, it maintains a checkpoint/cursor of processed records and will retry records from the same shard in order until the cursor shows completion. Can use Bisect On Function Error for retries

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Retry/failure handling

Service	Retry/failure capabilities
AWS Lambda API	Sync = client dependent Async = 0-2 retries, max event age 60 seconds – 6 hours
Amazon SNS	Retry with exponential backoff, up to 23 days
Amazon EventBridge	Retry with exponential backoff, up to 24 hours
Amazon SQS	Messages remain in queue until deleted or age beyond visibility timeout
Amazon Kinesis Data Streams	Retry until completion, can bisect batch

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda retries/failure handling

Synchronous

- The function returns an error to the caller
- Retries up to the client/invoking service

Asynchronous

- Retries by default (configurable)
- Configure on-failure destination on the AWS Lambda function

Event source mapping (queues)

- Batches retried, can be split and retry failed parts (configurable)
- Configure a dead-letter queue (DLQ) on the source queue

Event source mapping (streams)

- Retried until expiration, can also be split (configurable)
- Configure on-failure destination on the event source mapping

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Observability then and now



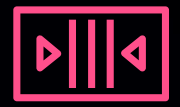
Traditional server



AWS Lambda function



S3 bucket



Amazon SQS queue



AWS Lambda function

Centralized code and application

Distributed code and application

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Observability is monitoring more than failures

Is it behaving as expected?

What is the usage?

What is the business impact?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Different types of logging

Unstructured

```
[INFO] 2021-04-21T12:34:56Z order 'fdbf7245' created with 5 products
```

Semi-structured

```
[INFO] 2021-04-21T12:34:56Z {"message": " Order 'fdbf7245' created with 5 products"}
```

Structured

```
{
    "loglevel": "INFO",
    "timestamp": "2021-04-21T12:34:56Z",
    "message": " Order 'fdbf7245' created with 5 products",
    "orderId": "fdbf7245",
    "productCount": 5
}
```

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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"
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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": "[redacted]",
  "CartId": "58dd189f",
"LogLevel": "INFO",
  "Timestamp": "2019-12-11
12:44:40.300473",
  "Message": "Added 2 items
'a23390f3' to cart '58dd189f'"
}
```

Log entry

```
"functionVersion": "$LATEST",
96,
e-cart",
egoryId"]
t", "Unit": "None"},
Cart", "Unit": "None"}
"CategoryId": "bca4cec1"
"PriceInCart": 100,
"QuantityInCart": "2"
}
```

Open-source client libraries available for

- Node.js
- Python
- Java

https://s12d.com/cwl-emf-client

aws

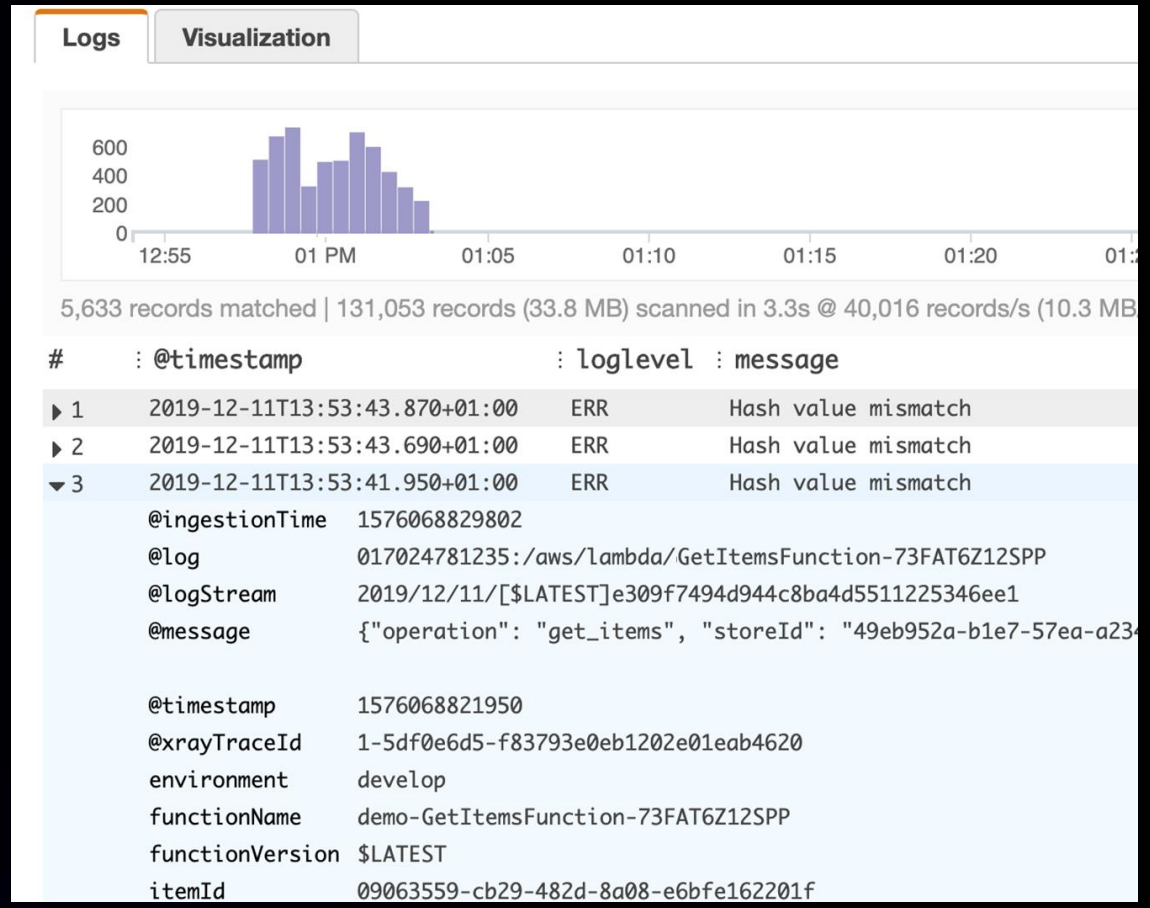
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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
```



https://s12d.com/loginsights-examples

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Dashboards

Key Metrics

Orders

15 k — Orders Created

14.3 k — Orders Fulfilled

-- — Orders Failed

Errors

0 — Lambda Errors

0 — API Gateway 5XX

-- — DynamoDB Errors

CreateOrder Latency

513 ms — p50

604 ms — p90

2.13 s — p99

GetOrder Latency

11.9 ms — p50

20.3 ms — p90

34.7 ms — p99

Useful links

CloudWatch Logs Insights queries:

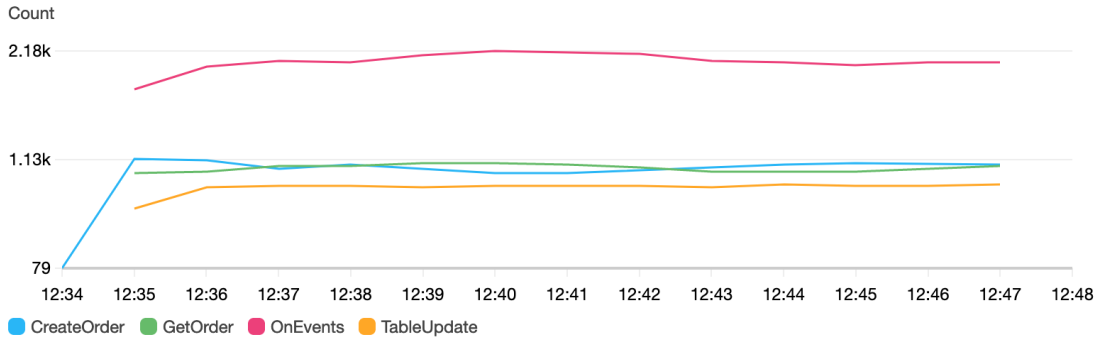
- Last 100 warning messages

Other links:

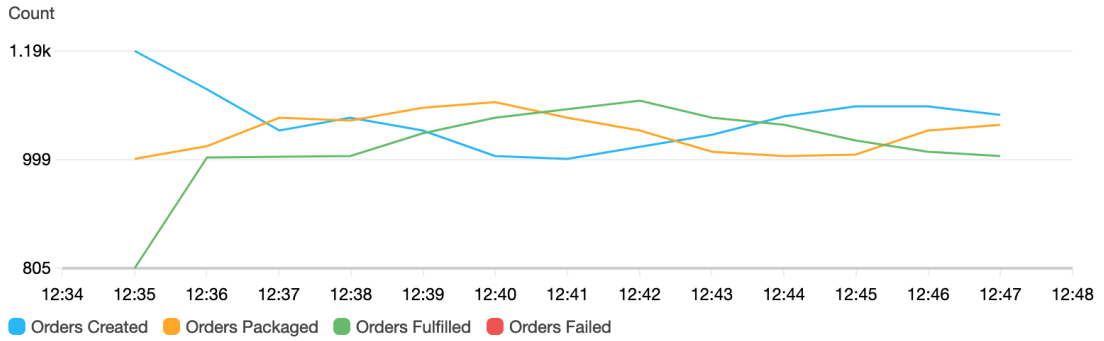
- Repository

Traffic

Requests



Orders



Latency and Duration

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda extensions

A NEW WAY TO INTEGRATE WITH THE AWS LAMBDA EXECUTION ENVIRONMENT

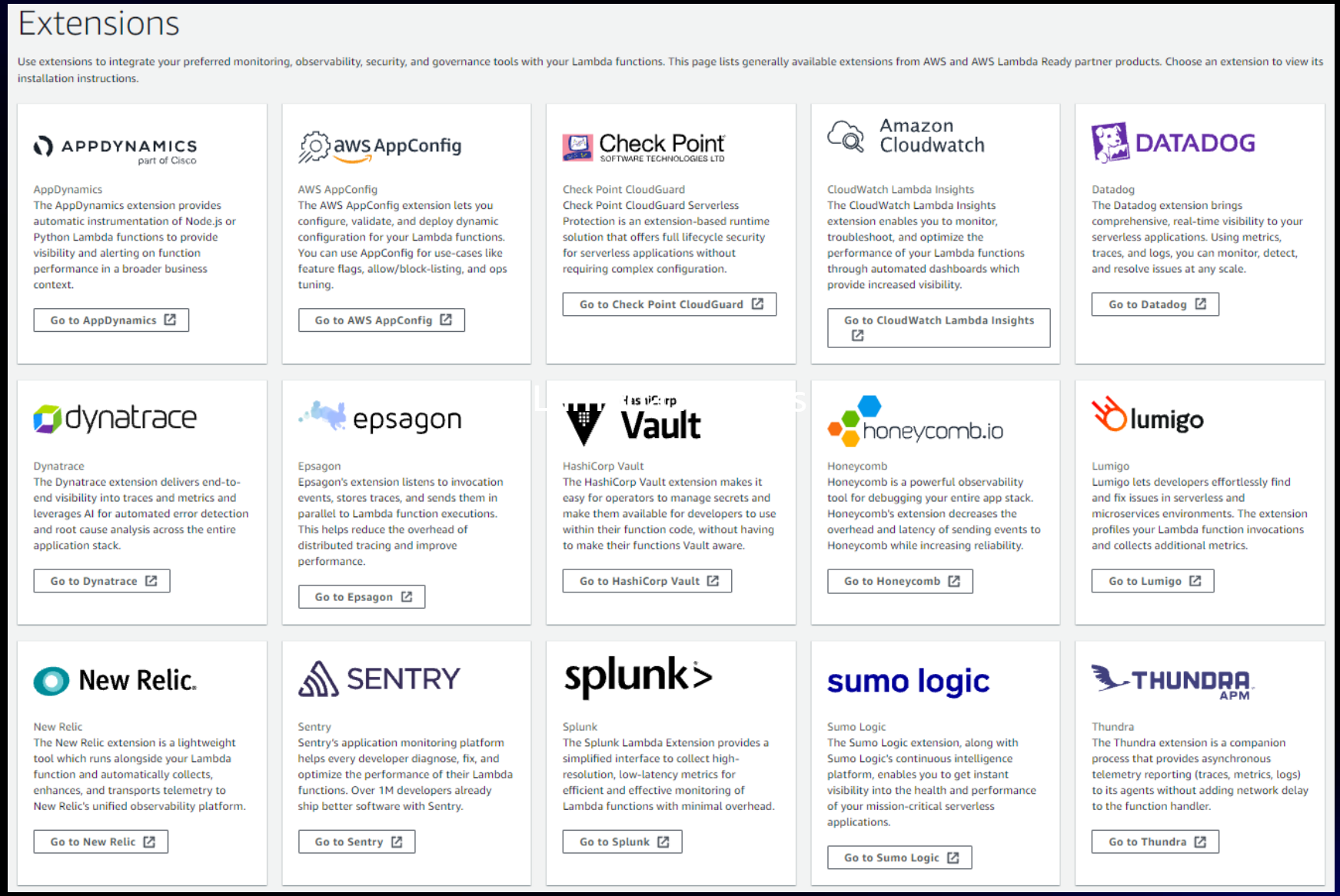
- Easier to integrate your existing tools deeply with AWS Lambda
- No complex installation or configuration

- Capture diagnostic information before, during, and after function invocation
- Automatically instrument your code without needing code changes
- Fetch configuration settings or secrets before the function invocation
- Detect and alert on function activity through hardened security agents

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

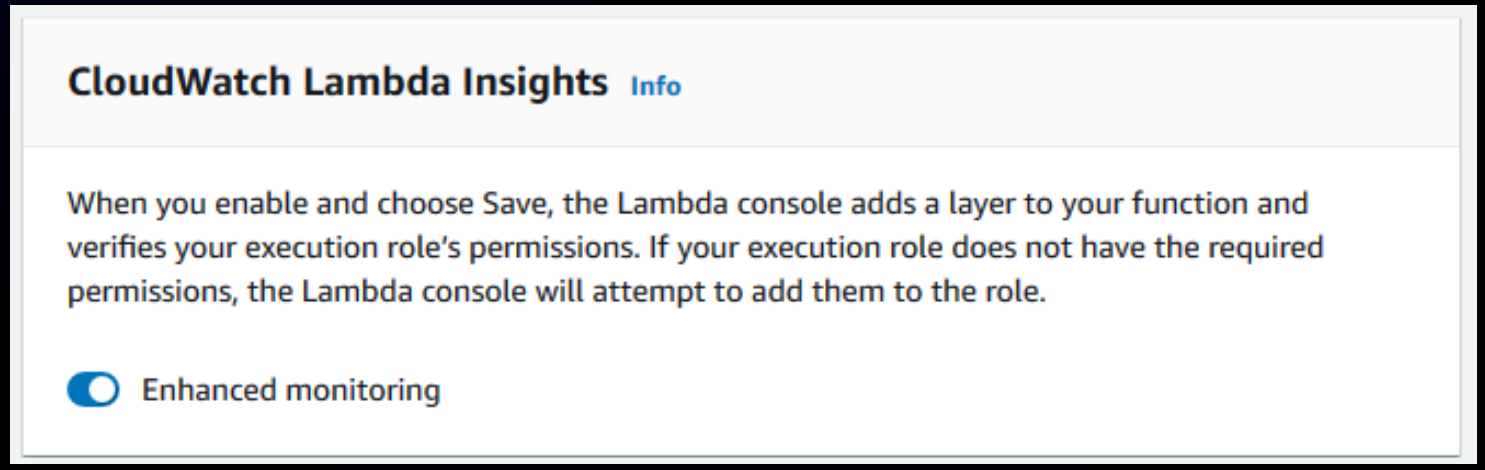
Extensions available today



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon CloudWatch Lambda Insights

MONITOR, TROUBLESHOOT, AND OPTIMIZE THE PERFORMANCE OF LAMBDA FUNCTIONS



Use cases

- Identify function issues, such as memory leaks
- Identify high-cost functions
- Identify performance changes caused by new function versions
- Understand latency drivers in functions

Features

- Deep linking and correlations across metrics, logs, and traces
- Automated dashboards showing function performance and health
- Deeper integration with CloudWatch Logs Insights and CloudWatch ServiceLens
- New Lambda performance metrics with observable context

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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, Amazon SQS batch processing, and more

Python

https://github.com/awslabs/aws-lambda-powertools-python



Java

https://github.com/awslabs/aws-lambda-powertools-java



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Healthy serverless: Best practices

Embrace failure!

- Observability = gathering data and more than just monitoring metrics
- Use platform retry mechanisms (losing events is bad!)
- Use embedded metric format = cheaper, faster metrics from structured logs
- Use CloudWatch Logs Insights to interactively query and analyze log data
- Configure log retention
- ServiceLens gives you that “single-pane” view into your applications
- AWS Lambda extensions – more easily use your existing tools
- AWS Lambda PowerTools – coding utilities for observability

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless observability learning path



s12d.com/mastering-observability

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

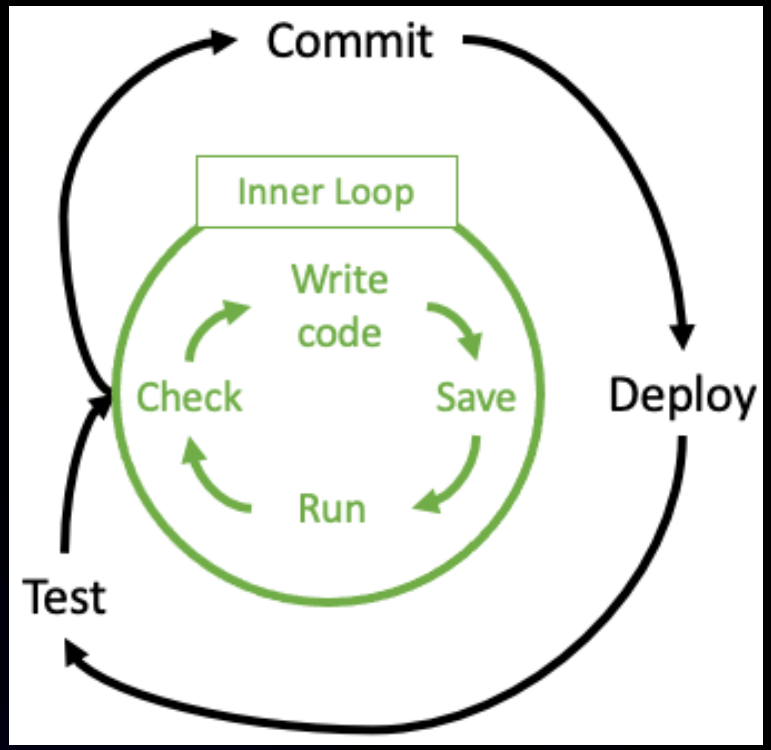
Development workflow

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Traditional developer workflow

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



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Service emulation



In serverless applications you configure the interfaces between services

Do not try to emulate these services on your local machine

Do mock event payloads

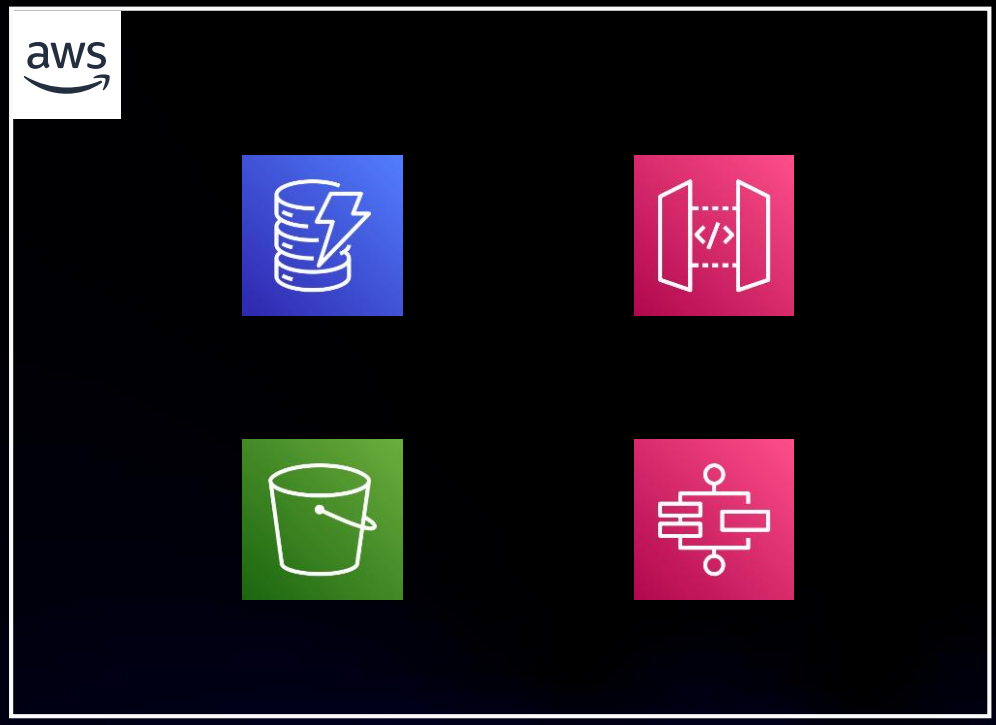
Image by Pixel_perfect from Pixabay

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Minimal service emulation







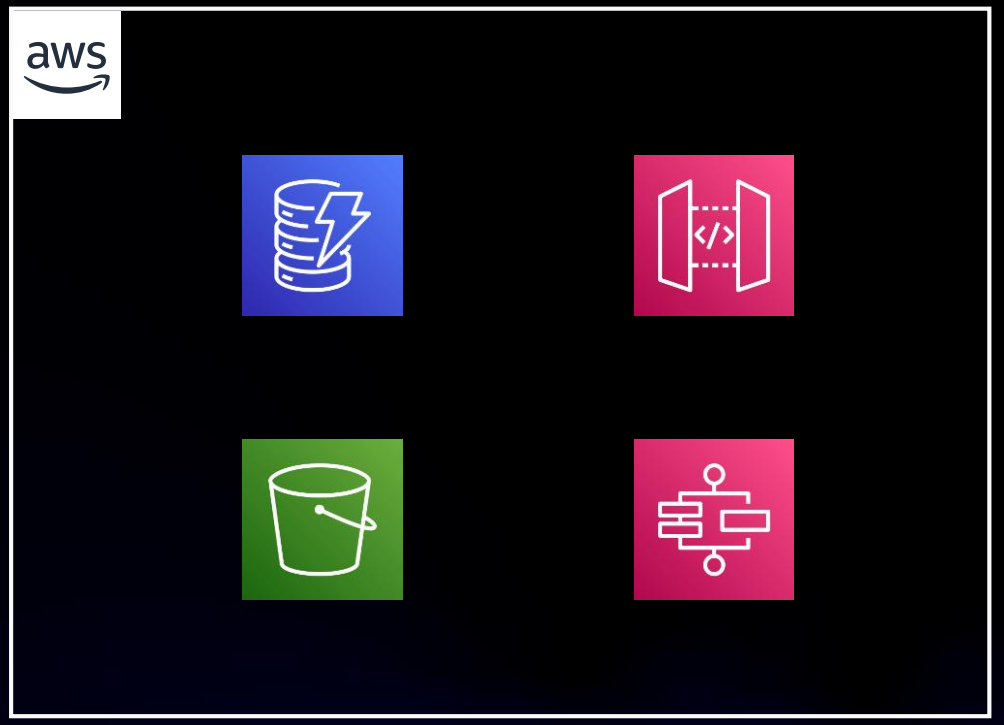
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS SAM Accelerate

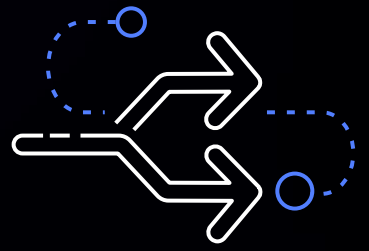
ITERATE AGAINST THE CLOUD WITH THE SPEED OF LOCAL DEVELOPMENT





AWS SAM sync

Quickly sync code changes with the cloud



Incremental builds

Build only the parts of your code that have changed



Aggregated feedback

Tail application logs that include output from all resources in a single command

s12d.com/accelerate

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Development workflow: Best practices

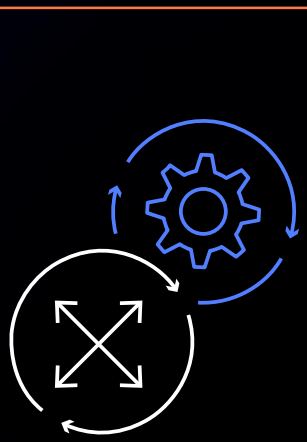
- Don’t emulate services locally
- Test business logic locally, the rest in the cloud
- Try AWS SAM Accelerate

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building well-architected serverless applications



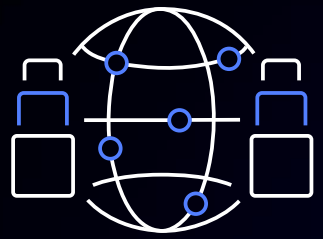
s12d.com/well-architected-serverless



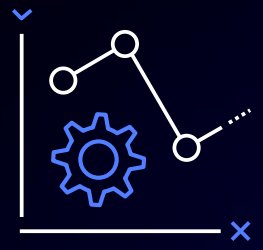
Operational Excellence



Security



Reliability



Performance Efficiency



Cost Optimization

Serverless Applications Lens – AWS Well-Architected Framework

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Summary

Thinking in events

Service-full serverless

Fabulous functions

Healthy serverless

Development workflow

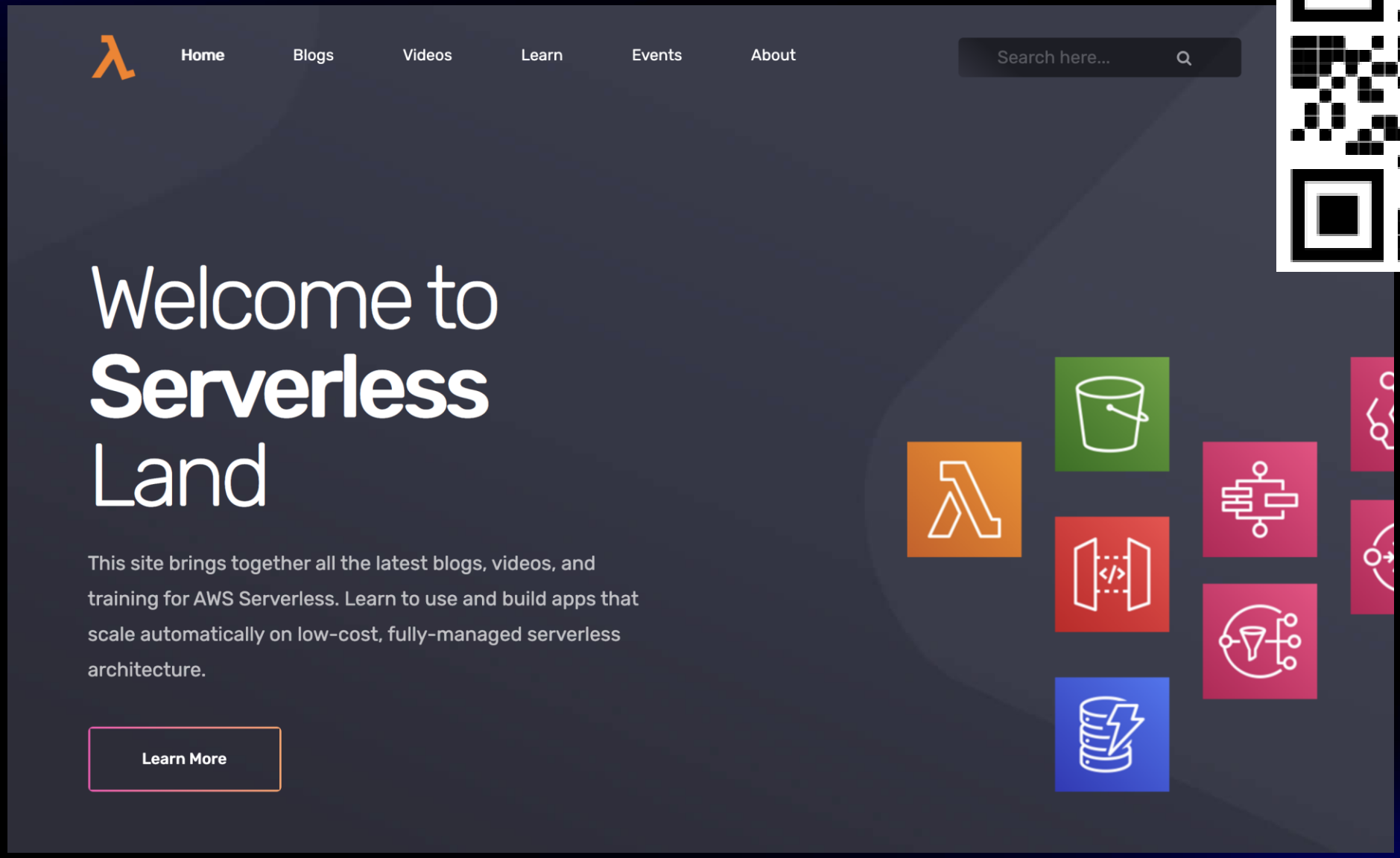
Resources



s12d.com/svs402-ri21

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverlessland.com



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Julian Wood

 @julian_wood
Email: jrwood@amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.