



AWS
User Groups
Anápolis, Brazil

AWS Serverless: Simplificando Soluções em Nuvem



AWS
User Groups
Anápolis, Brazil

AWS Serverless: Simplifying Cloud Solutions

About the Speaker

Weder Mariano de Sousa

Specialist Senior Java - GFT

Technician **System Development**
Graduated **Computer Science**

Post Graduate in **Midias UFG**

Post Graduate in **Information Security**



GOJava
Leader



- <https://www.linkedin.com/in/wedermarianodesousa/>
- <https://github.com/weder96>
- <https://twitter.com/weder96>
- <https://dev.to/weder96>



SERVERLESS COMPUTING



What is Serverless ?

“Removing the developer's responsibility for configuring computational resources, leaving only the responsibility for writing the functions that make up the developed artifact “

[Hendrickson et al. 2016]



https://aws.amazon.com/serverless/?nc1=h_ls

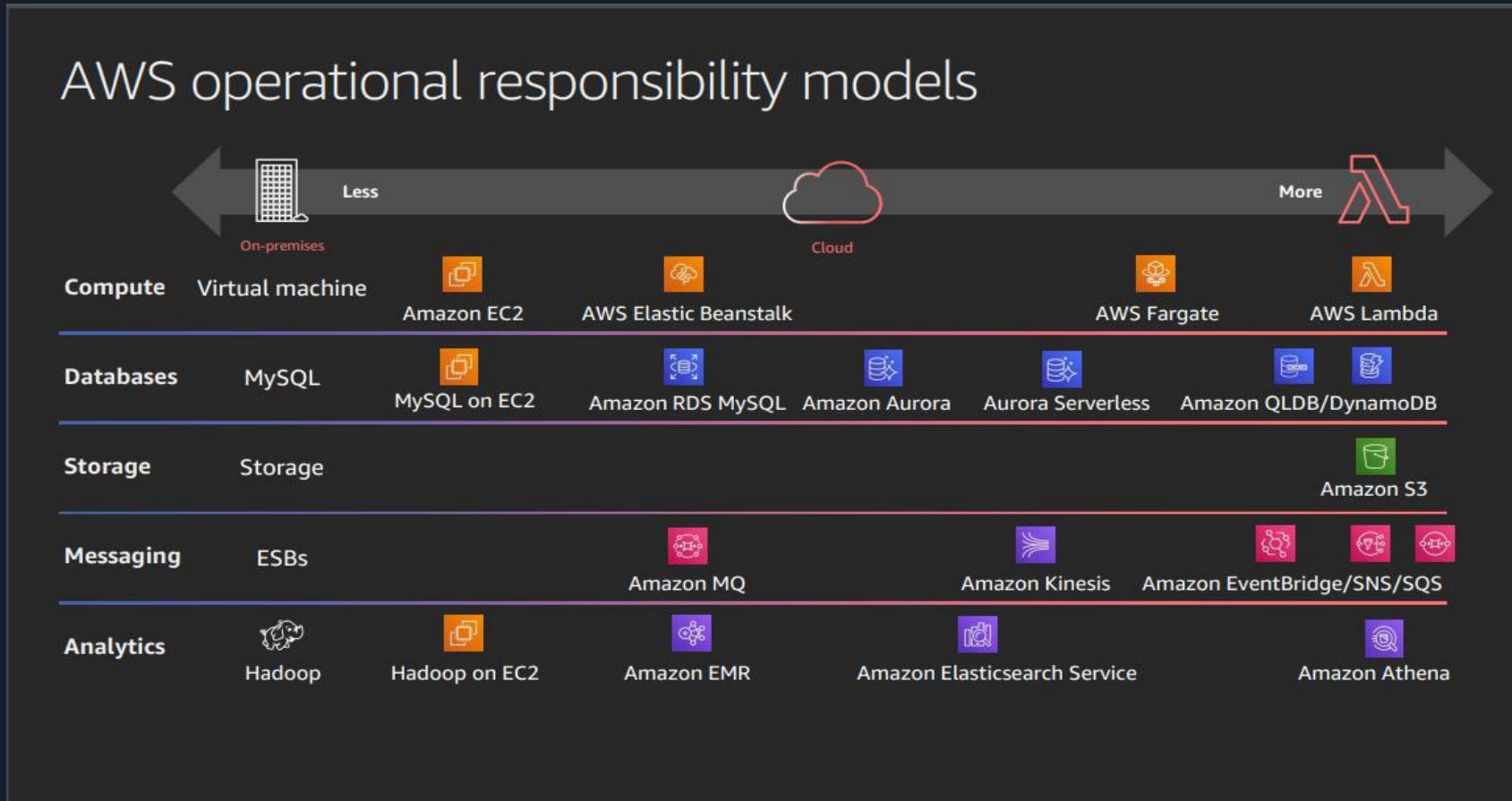
What is AWS Serverless ?

AWS Serverless is a set of services offered by Amazon Web Services (AWS) that enable developers to build and run applications and services without having to manage the underlying infrastructure. With the serverless model, users can focus on application logic, while AWS automatically handles resource provisioning, scaling, and server administration. This approach facilitates agile development and increases efficiency by eliminating routine infrastructure management tasks.



https://aws.amazon.com/serverless/?nc1=h_ls

Serverless Explained : AWS Operational Responsibility Model



Comparison of operational responsibility

More opinionated



AWS Manages

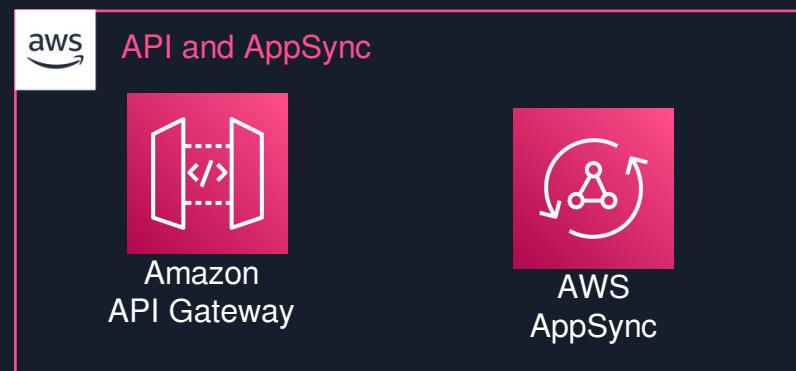
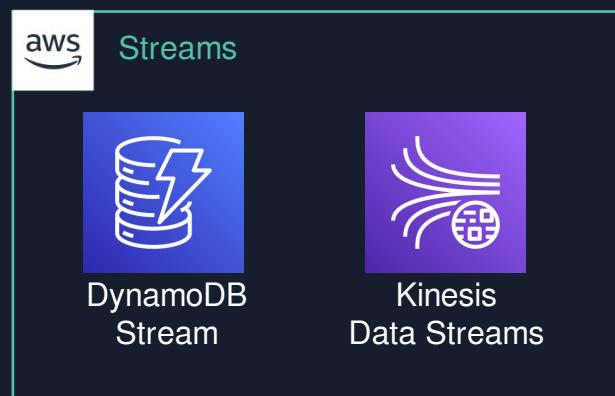
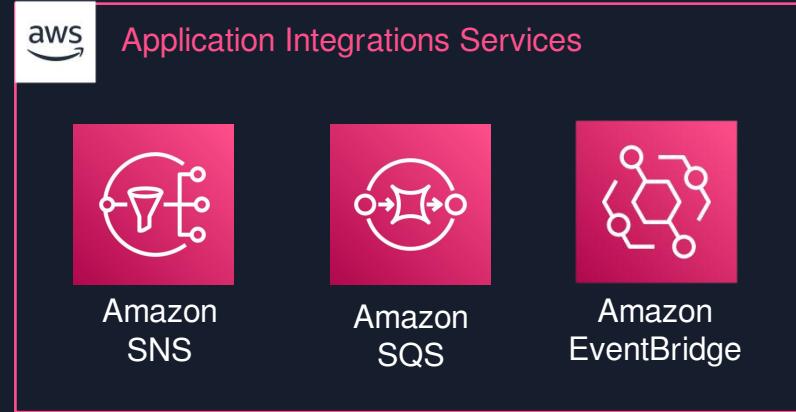
- Data source integrations
 - Physical hardware, software, networking, and facilities
 - Provisioning
- Container orchestration, provisioning
 - Cluster scaling
 - Physical hardware, host OS/kernel, networking, and facilities
- Container orchestration control plane
 - Physical hardware software, networking, and facilities
- Physical hardware software, networking, and facilities

Customer Manages

- Application code
- Application code
- Data source integrations
- Security config and updates, network config, management tasks
- Application code
- Data source integrations
- Work clusters
- Security config and updates, network config, firewall, management tasks
- Application code
- Data source integrations
- Scaling
- Security config and updates, network config, management tasks
- Provisioning, managing scaling and patching of servers

Less opinionated

AWS Services Serverless





Why Serverless Simplifies Cloud Solutions ?



Why Serverless Simplifies Cloud Solutions ?

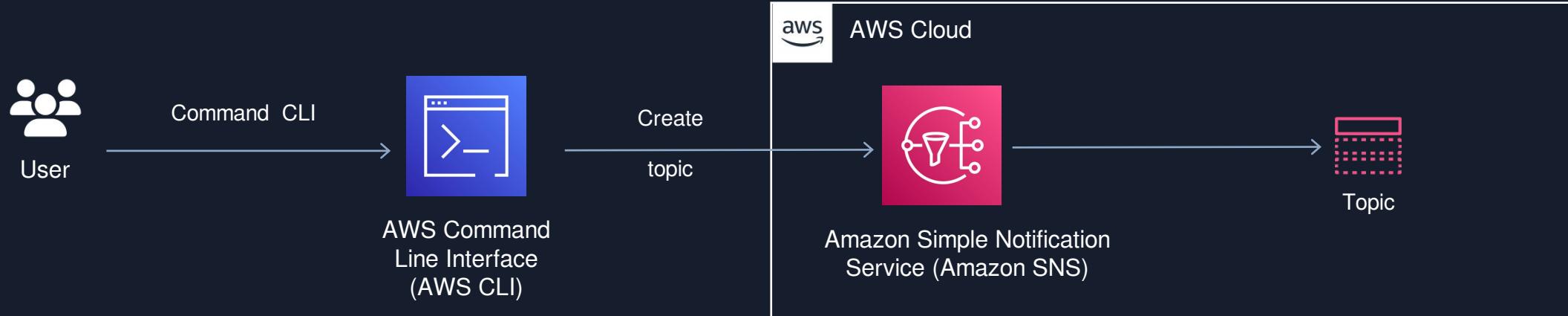
-  Reduced Infrastructure Management
-  Automatic Scalability
-  Consumption-Based Billing
-  Accelerated Development
-  Less Operational Overhead
-  Flexibility in Language and Tool Choice
-  Simple Deployment
-  Easy Integration
-  Resiliency and High Availability
-  Focus on business logic
-  Reduced Time to Market
-  Increased Security
-  Facilitated Microservices
-  Integrated Monitoring and Analytics
-  Easy Experimentation and Prototyping



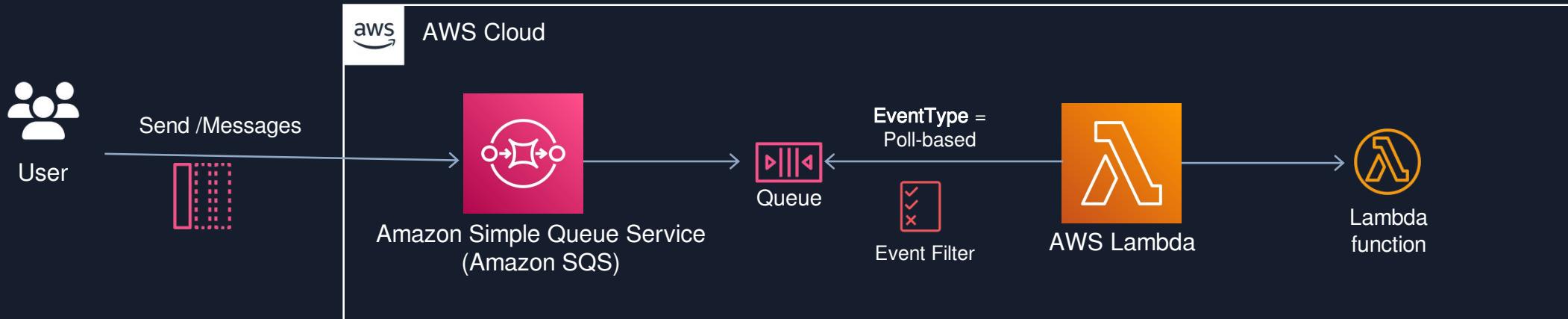
Architecture Serverless



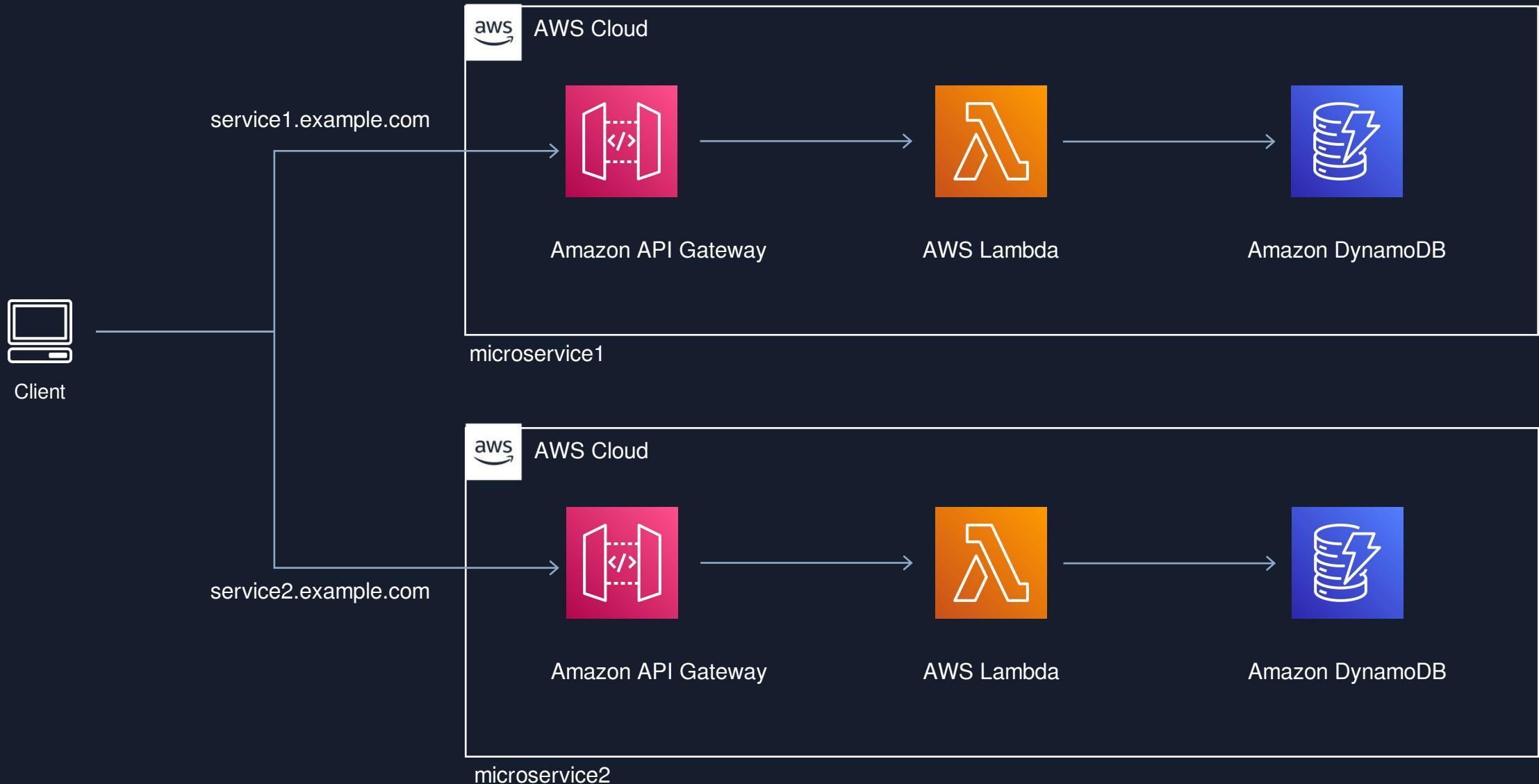
Amazon SNS with AWS Management Console



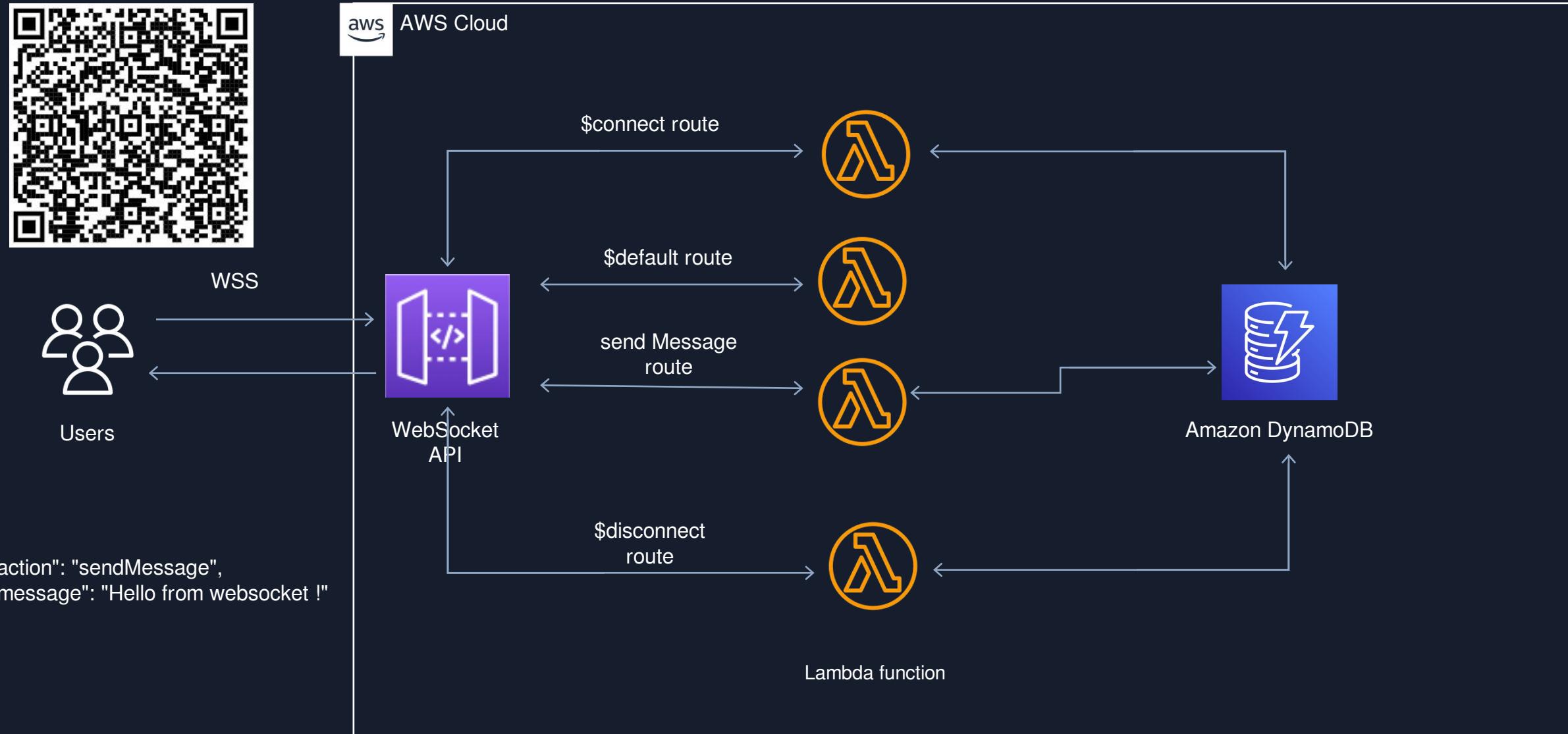
Amazon SQS Queue Polling From AWS Lambda



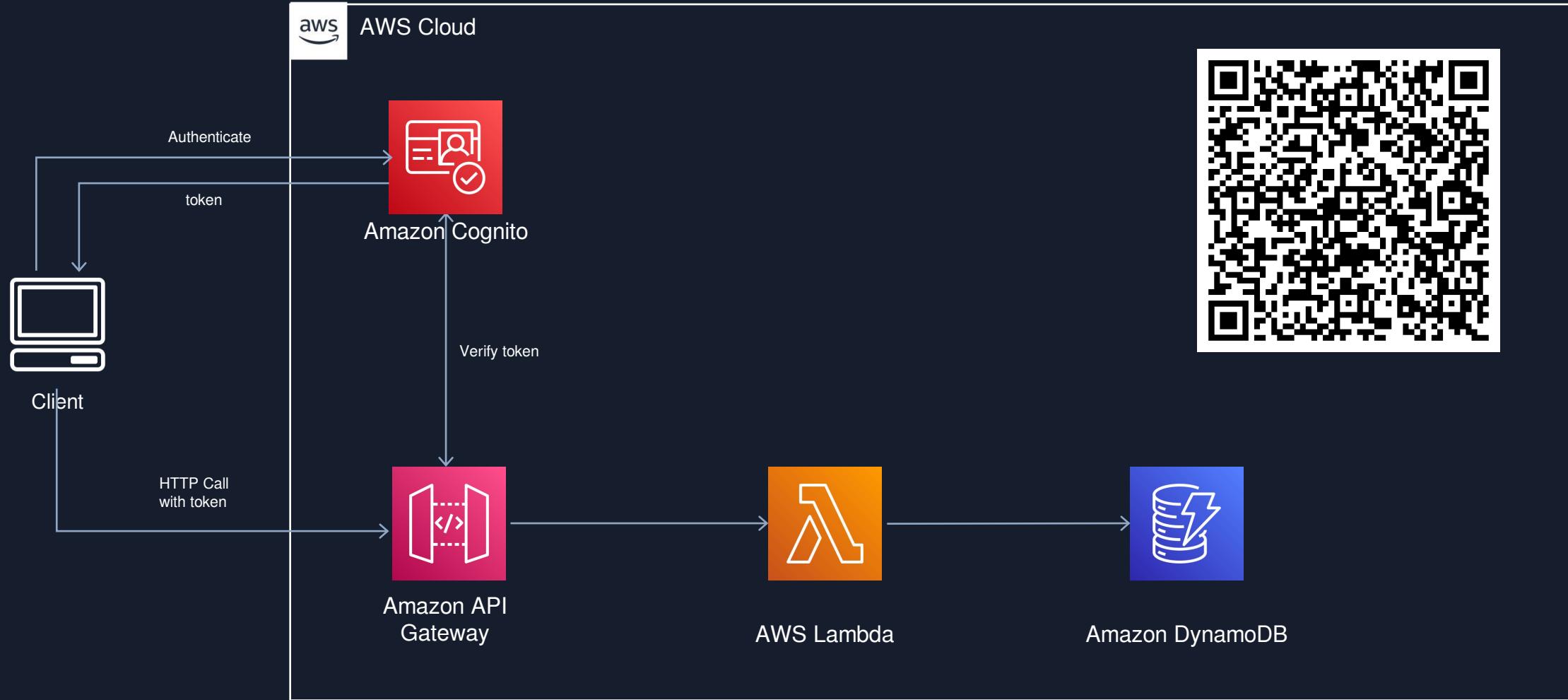
Microservices “REST”



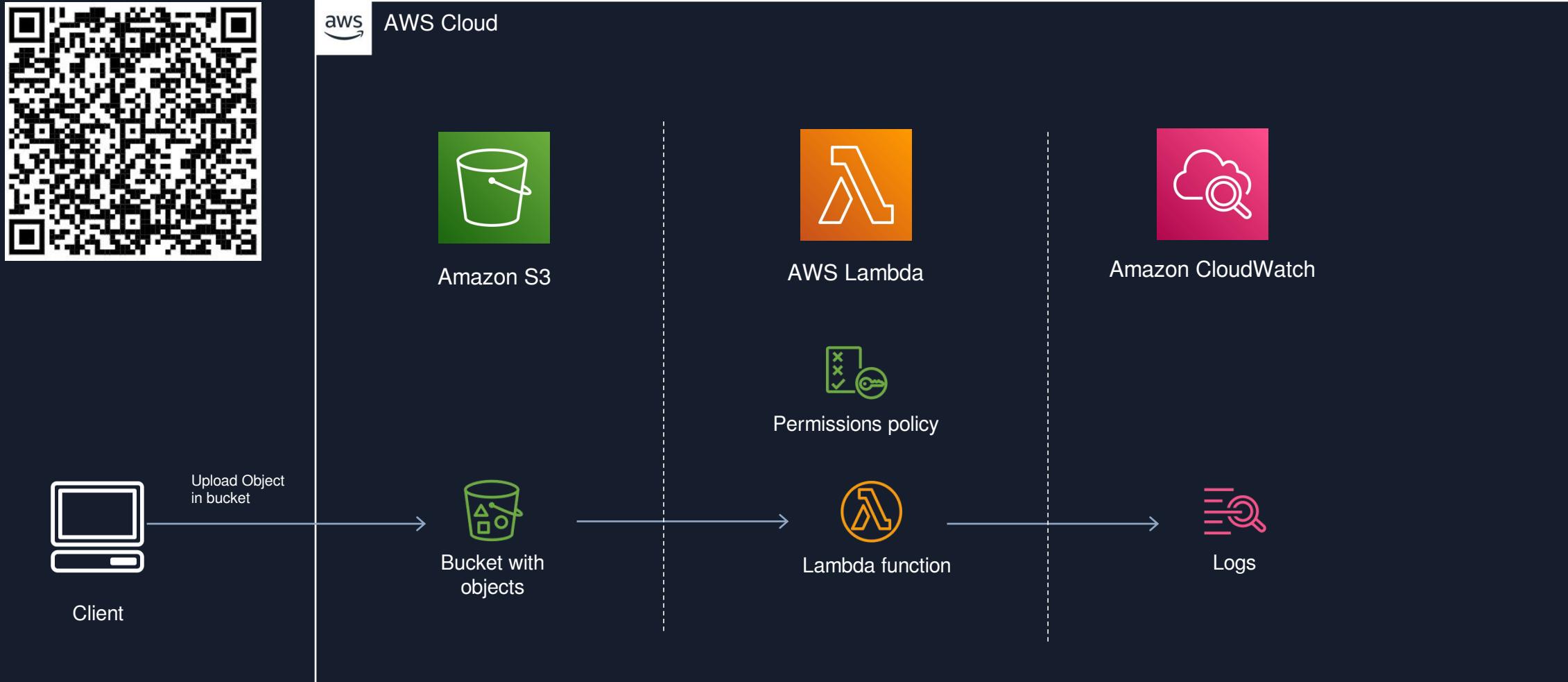
Build Serverless Chat App with a WebSocket API and Lambda



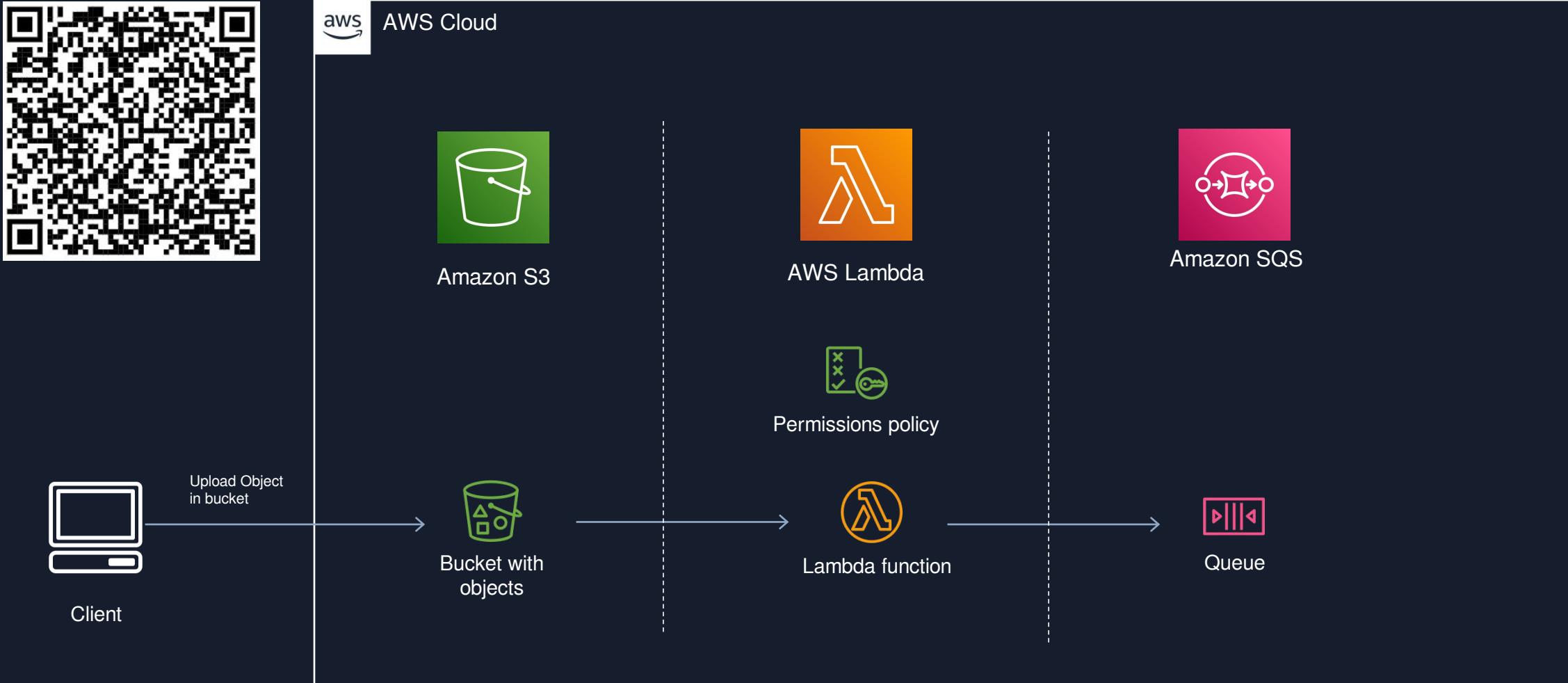
Secure your API Gateway with Amazon Cognito User Pools



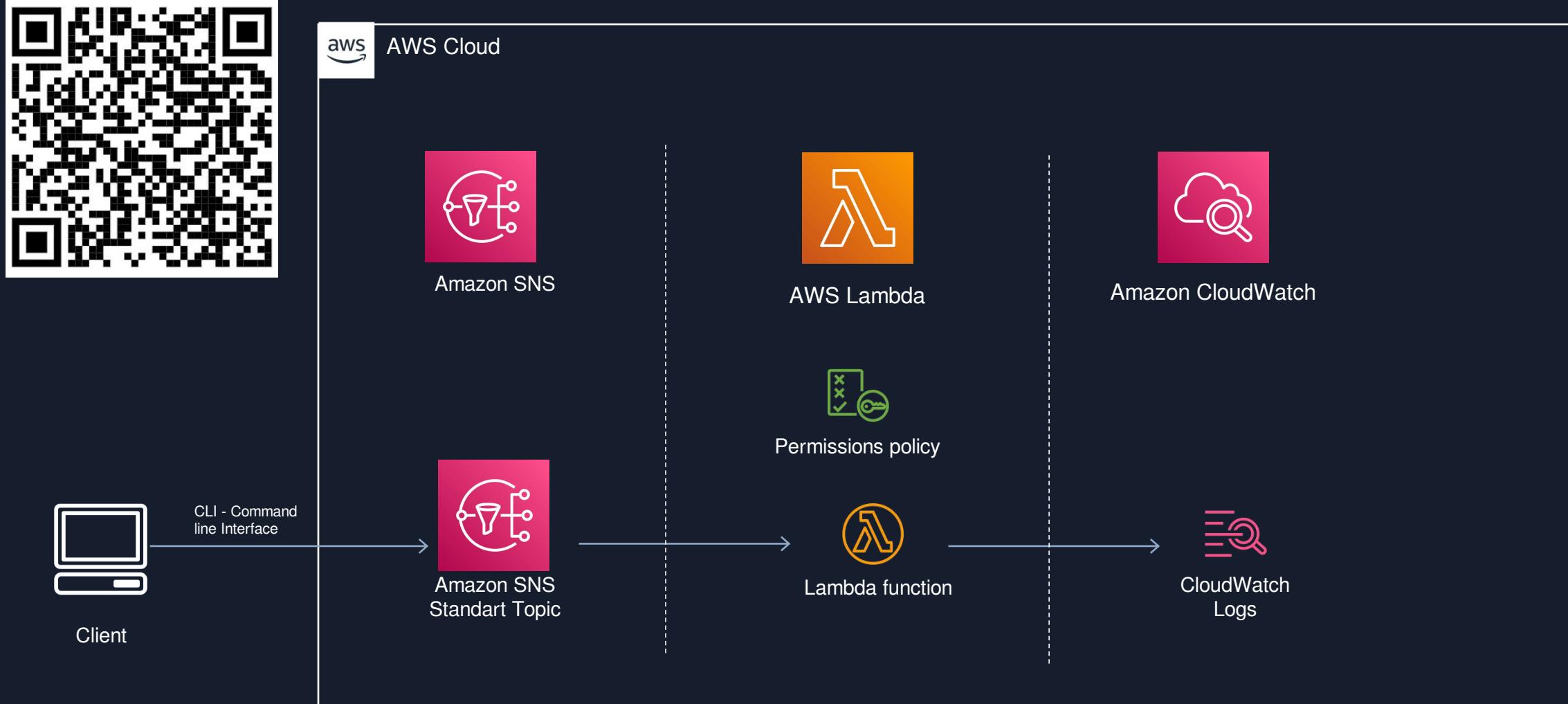
Using an Amazon S3 trigger to invoke a Lambda function



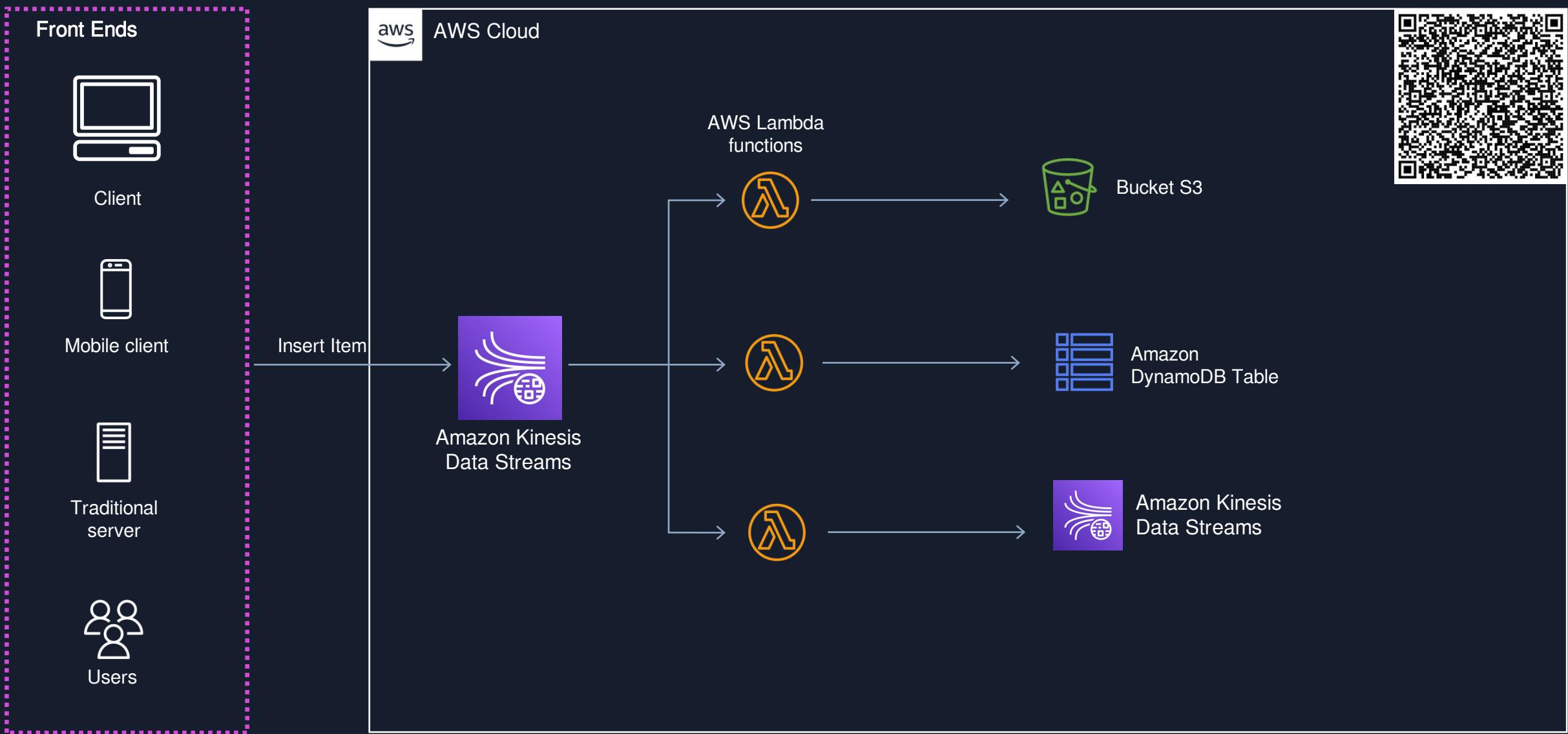
AWS Lambda Destination to SQS - DLQ Case



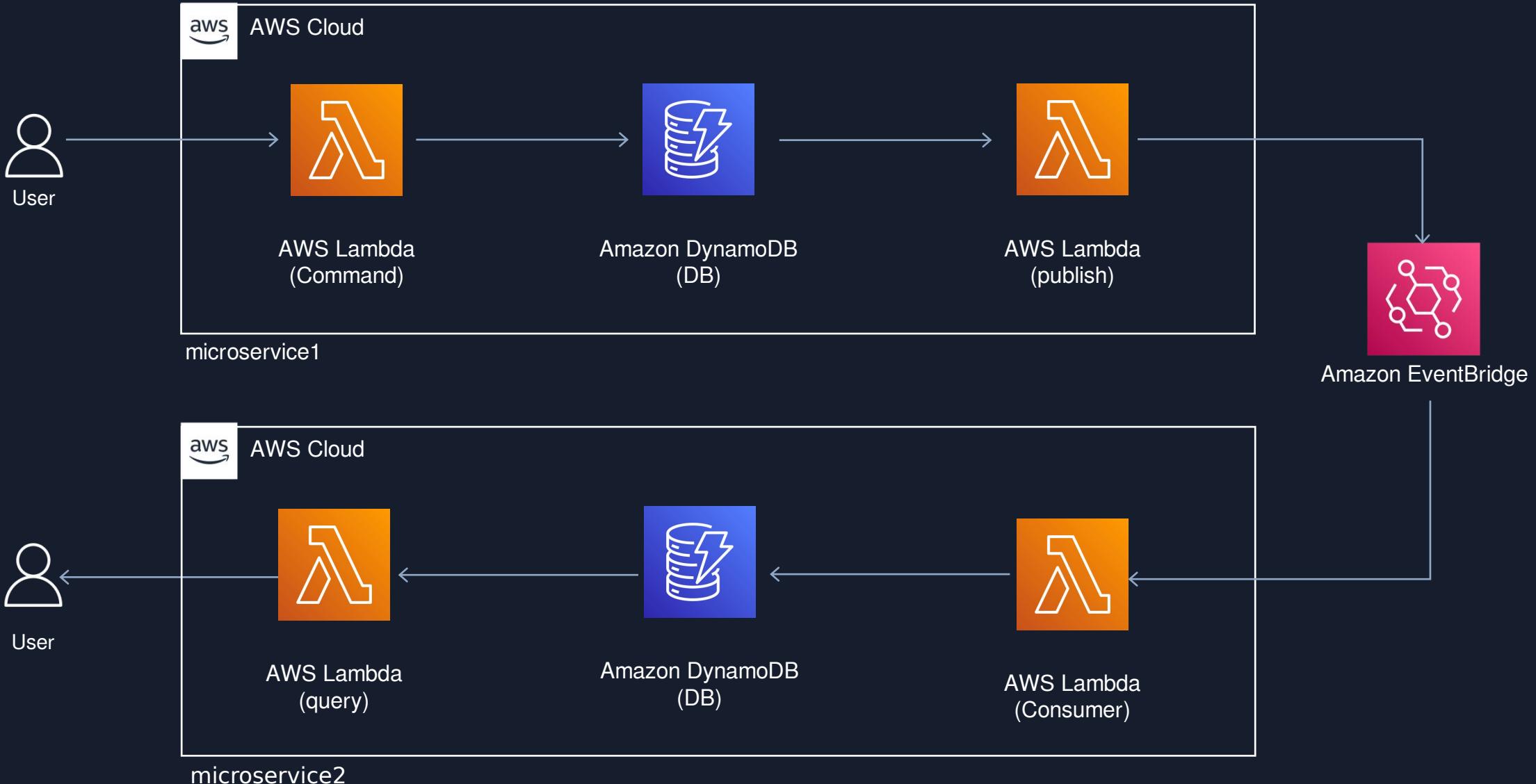
Using an Amazon SNS to send message to Lambda function



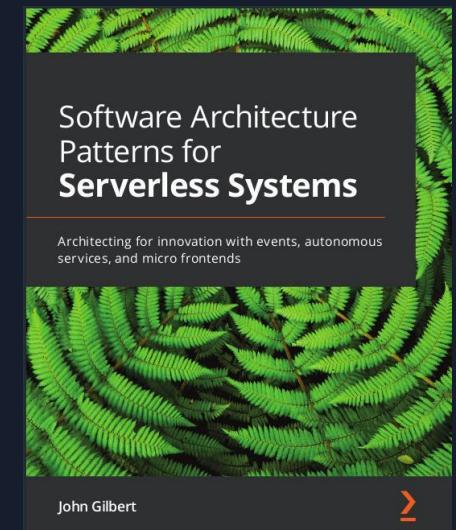
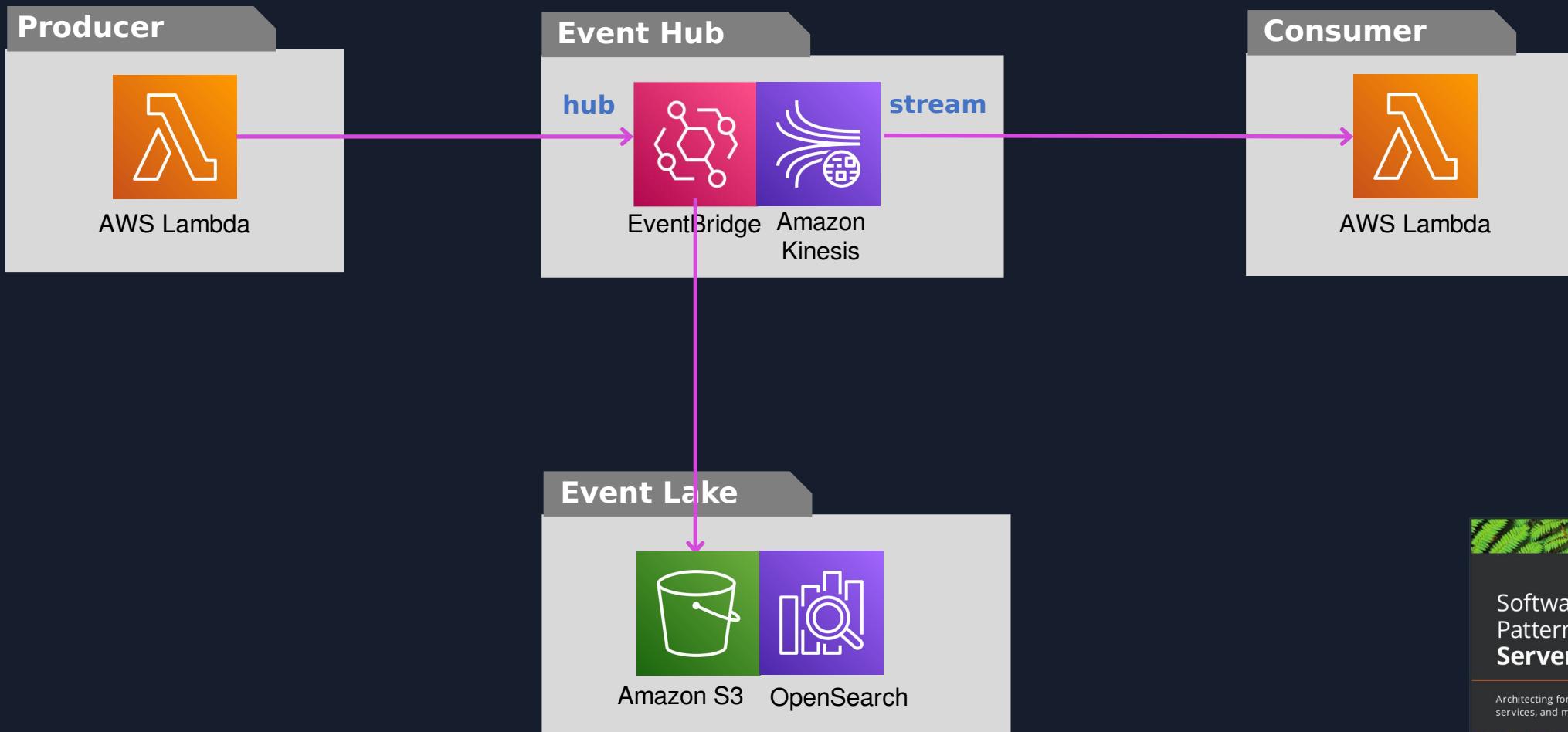
Processing performance with Amazon Kinesis Data Streams and AWS Lambda



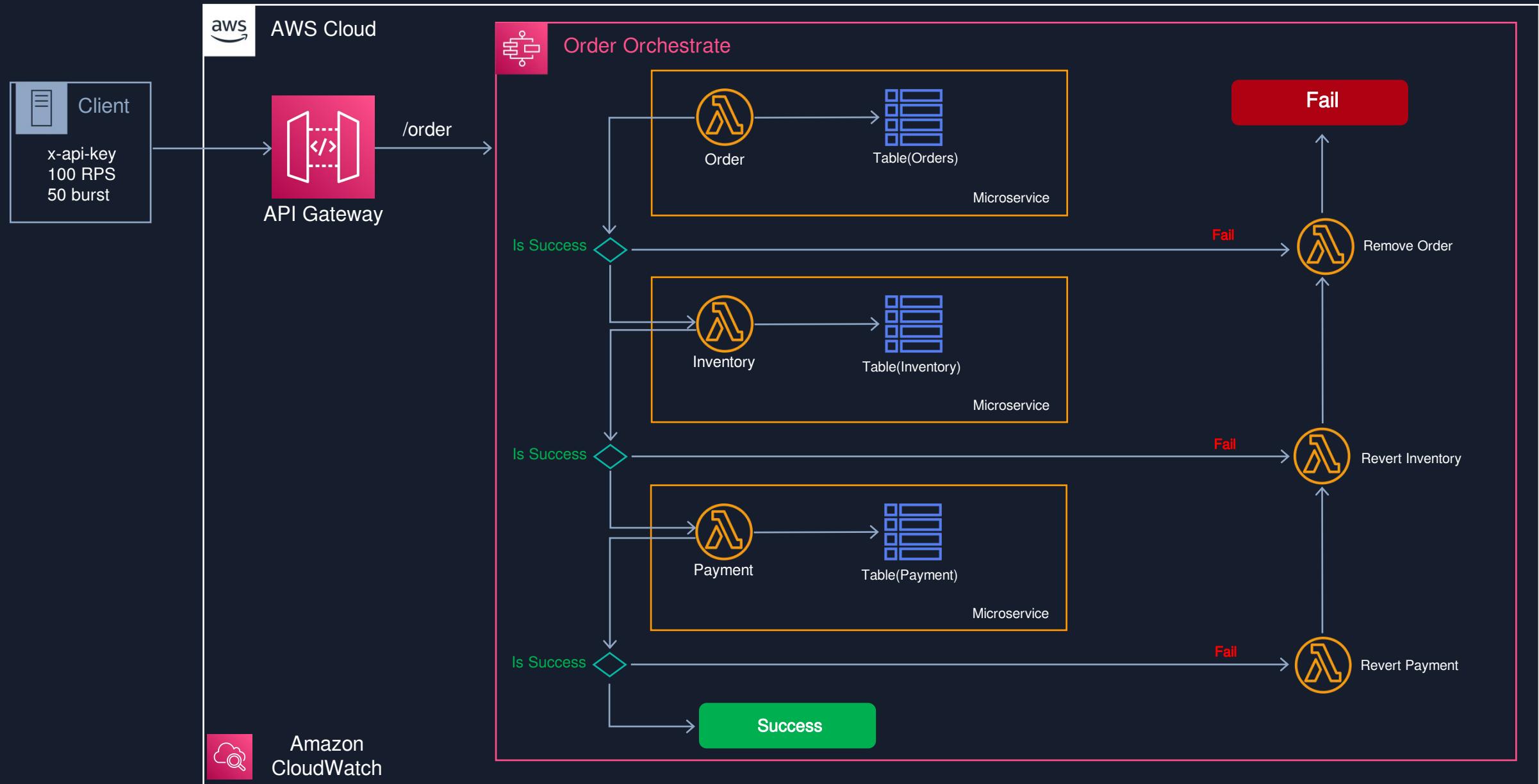
Command, Publish, Consume, Query (CPCQ)



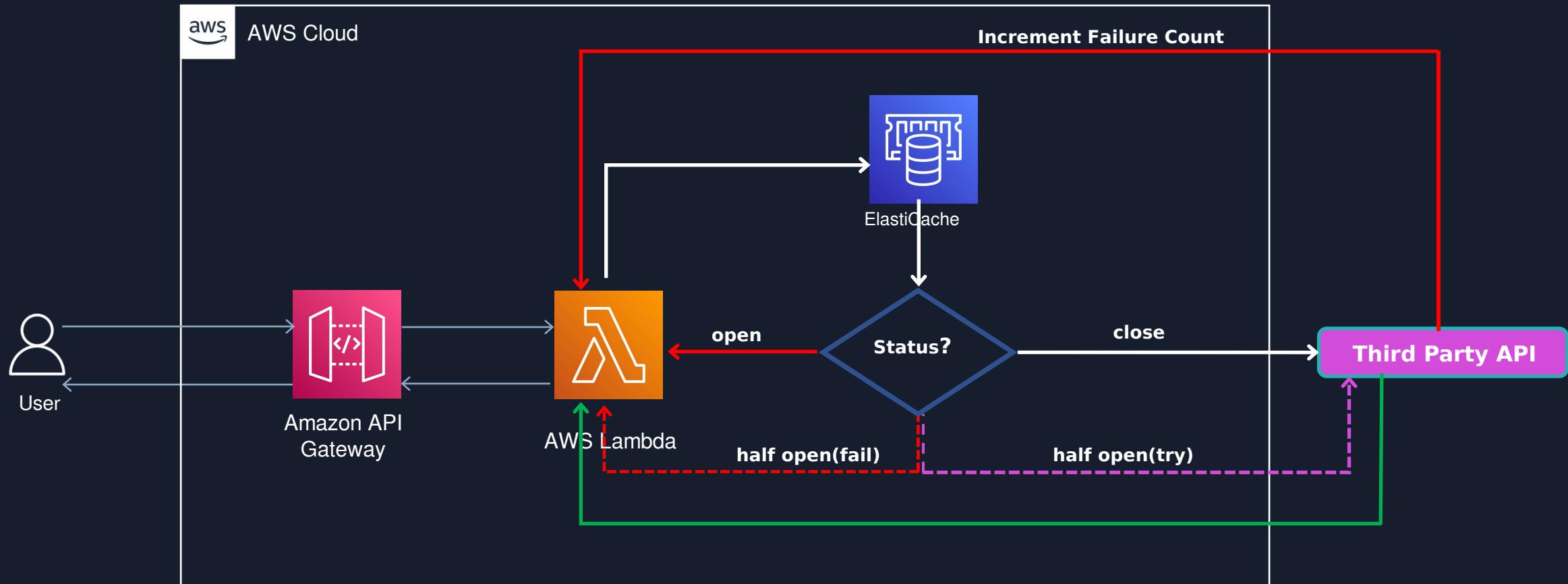
Event hub



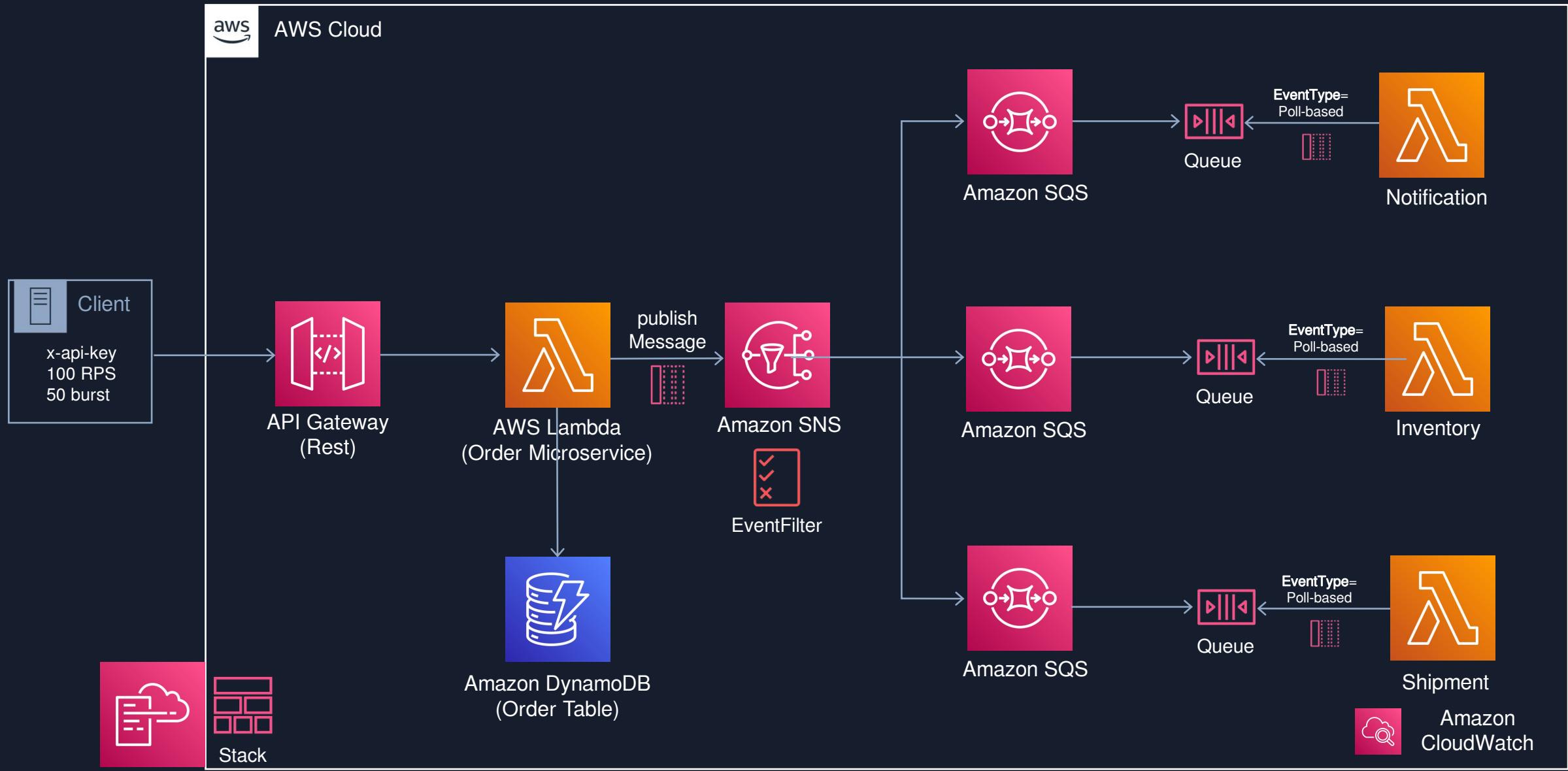
Saga Pattern For Orchestrate Distributed Transaction (AWS StepFunctions)



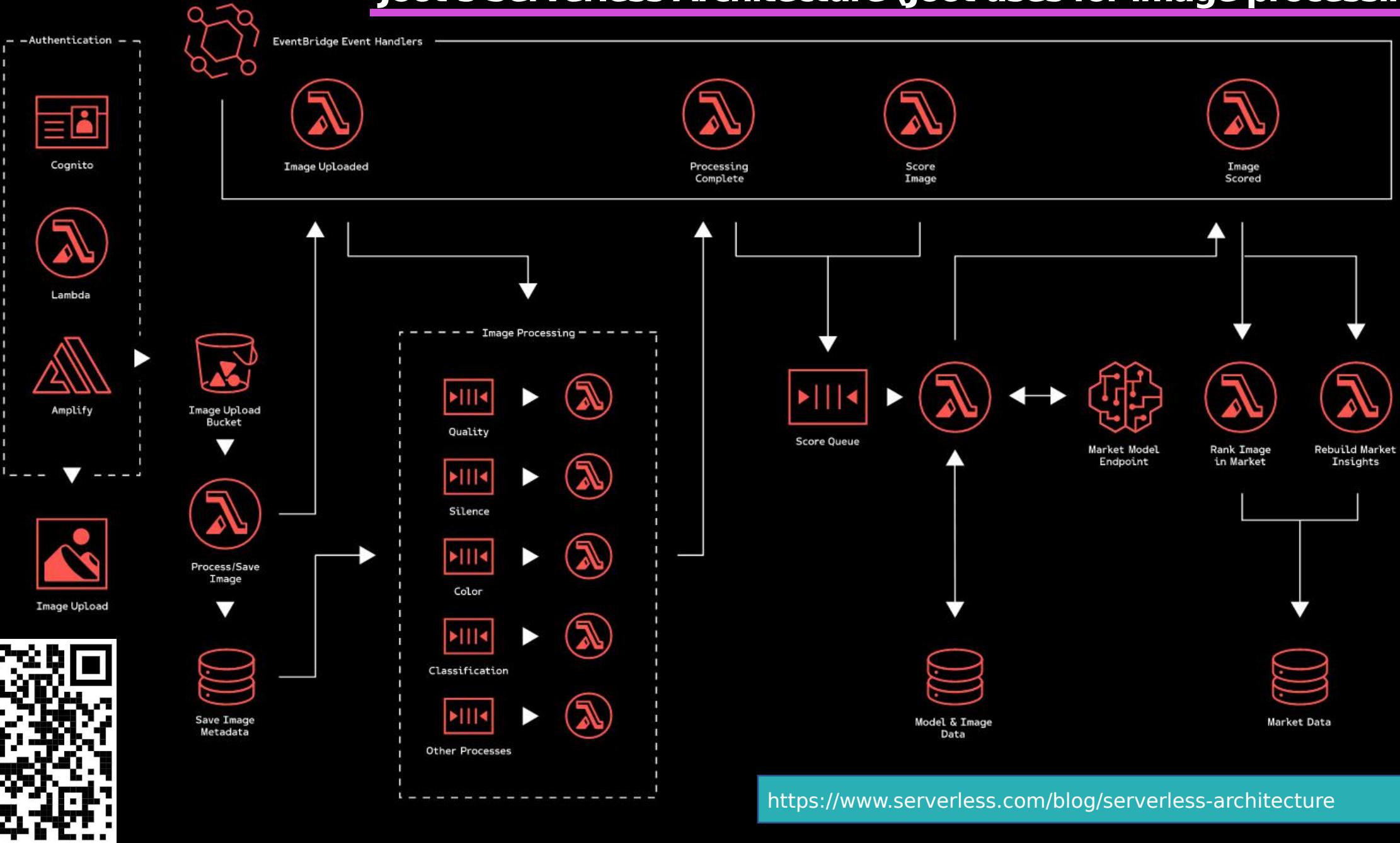
Circuit Breaker and Many More by Jeremy Daly



Fan-Out Serverless Architectures Using SNS, SQS and Lambda



Joot's Serverless Architecture (Joot uses for image processing)



How to build, run and deploy?



When starting designing, and building serverless function, you might wonder how to deploy your function into the cloud. With AWS, there are some ways we can deploy, test and invoke your function:



Using the AWS Console Management: we can create lambda function, upload code, add triggers, and test your Lambda function manually. You might use this way when first.



AWS CLI: you also can use AWS Lambda CLI to create, deploy, invoke, manage, monitor your Lambda function. You can use existing commands to deploy and test your Lambda function automatically without manual process. But this isn't good for production and large project.



AWS Serverless Application Model (AWS SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings.



AWS Cloud Development Kit (AWS CDK) is an open source software development framework to define your cloud application resources using familiar programming languages.

How to build, run and deploy?



[Serverless Framework](#) - The Serverless Framework consists of an open source CLI and a hosted dashboard. Together, they provide you with full serverless application lifecycle management.



[Chalice](#) is a framework for writing serverless apps in Python. It allows you to quickly create and deploy applications that use AWS Lambda.



[Arc.codes](#) provides everything you need to build massively scalable serverless apps with low code, clear and terse config, and zero ceremony.



[Claudia.js](#) makes it easy to deploy Node.js projects to AWS Lambda and API Gateway.



AWS Storage



Amazon S3 CLI Commands

joinCommunity2023 / learning / 09_S3 / README.md

Preview

Code

Blame

200 lines (144 loc) · 4.59 KB

Code 55% faster with GitHub Copilot

Create S3 Bucket

```
aws s3 mb s3://s3-join-community-2023-files
```



Create S3 bucket in specific AWS region

```
aws s3 mb s3://s3-join-community-2023-files-us-ea-2 --region us-east-2
```



put version at bucket

```
aws s3api put-bucket-versioning --bucket s3-join-community-2023-files-us-ea-2 --versioning-configuration Status=Enat
```



check bucket status version

```
aws s3api get-bucket-versioning --bucket s3-join-community-2023-files-us-ea-2
```



Suspending versioning

```
aws s3api put-bucket-versioning --bucket s3-join-community-2023-files-us-ea-2 --versioning-configuration Status=Susp
```



Copy files to S3 bucket

```
aws s3 cp images/ s3://s3-join-community-2023-files --recursive --include "*.jpg"
```



List S3 buckets

```
aws s3 ls
```



Amazon S3



AWS Database



DynamoDB Primary Key, Partition Key and Sort Key

A primary key **uniquely identifies** each item in the table, so two items can have the same key. DynamoDB supports two different kinds of primary keys:

Partition key

Partition key and sort key

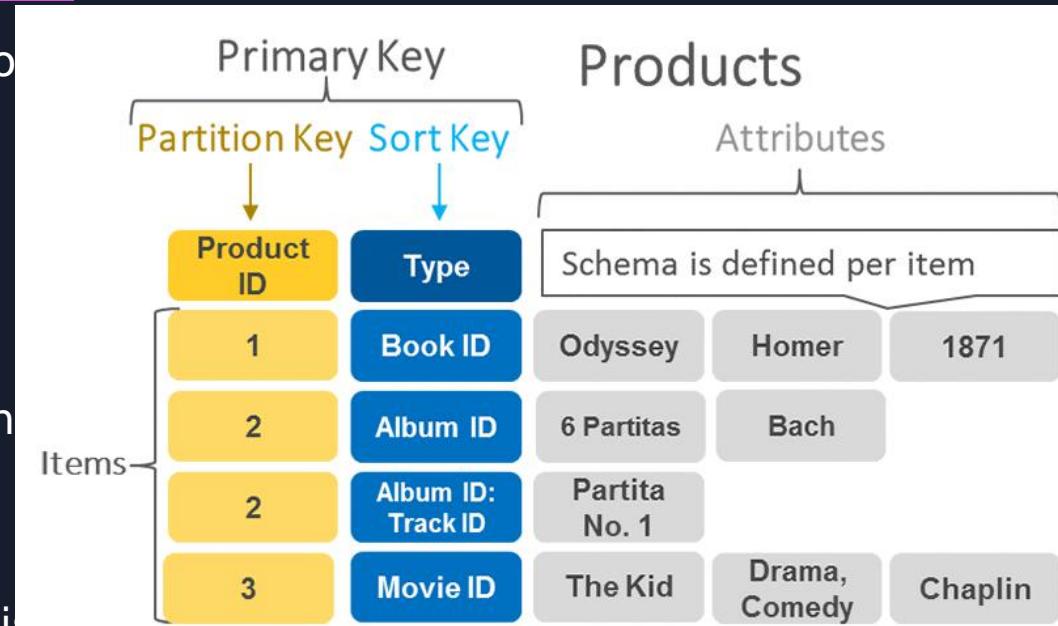
Partition key

A simple primary key, composed of one attribute known as the partition key.

Partition key and Sort Key

It is Referred to as a composite primary key, this type of key is composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key.

DynamoDB uses the partition key value as input to an internal hash function. A composite primary key gives you additional flexibility when **querying data**



General information

Partition key
orderId (String)

Sort key
orderDate (String)

Alarms
No active alarms

Point-in-time recovery (PITR) [Info](#)
Off

► Additional info



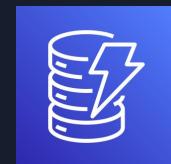
DynamoDB Python(Boto3)

Files

weder96 learning serverless with cli commands

Code Blame 39 lines (32 loc) · 1004 Bytes Code 55% faster with GitHub Copilot

```
1 import json
2 import boto3
3 import uuid
4
5 client = boto3.resource('dynamodb')
6
7 def lambda_handler(event, context):
8     print('Starting input lambda function call')
9     print(event['body'])
10    table = client.Table('orders')
11    myuuid = uuid.uuid4()
12
13    print('Your UUID is: ' + str(myuuid))
14
15    order=json.loads(event['body'])
16
17    data = table.put_item(
18        Item={
19            'orderId': str(myuuid),
20            'orderDate': order['orderDate'],
21            'status': order['status'],
22            'desc': order['desc'],
23            'updateOrderDate': order['updateOrderDate'],
24            'Name': order['Name'],
25            'Email': order['Email']
26        }
27    )
28
```



https://github.com/weder96/joinCommunity2023/tree/main/learning/08_dynamodb

DynamoDB CLI Commands

JoinCommunity2023 / learning / 08_dynamodb /

Name	Last commit message
...	
crud	🚀 add boto3 dynamodb with lambda
README.md	🚀 add boto3 dynamodb with lambda

README.md

Dynamodb

CLI

Create table DynamoBD

```
aws dynamodb create-table \
--table-name orders \
--attribute-definitions \
  AttributeName=id,AttributeType=S \
  AttributeName=status,AttributeType=S \
--key-schema \
  AttributeName=id,KeyType=HASH \
  AttributeName=status,KeyType=RANGE \
--provisioned-throughput \
  ReadCapacityUnits=5,WriteCapacityUnits=5 \
--table-class STANDARD
```

Describe table DynamoBD

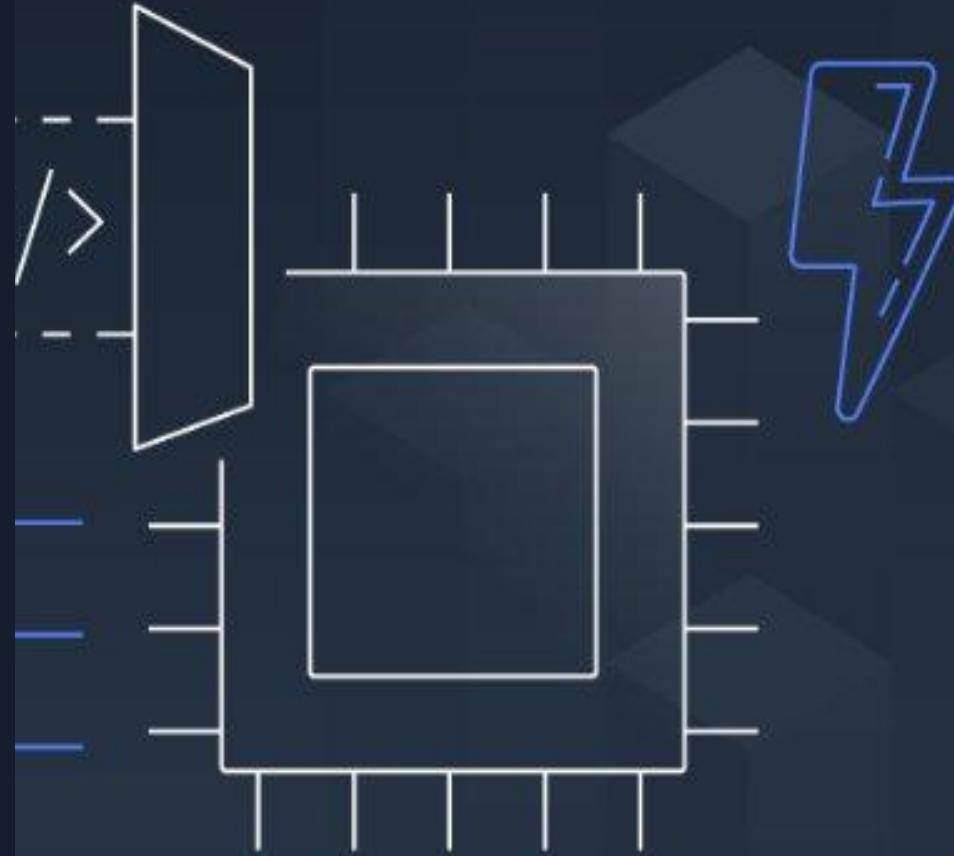
```
aws dynamodb describe-table --table-name Order
```



https://github.com/weder96/joinCommunity2023/tree/main/learning/08_dynamodb



AWS API Gateway



Amazon API Gateway Concepts

API Deployment - a point-in-time snapshot of your API Gateway API resources and methods. To be available for clients to use, the deployment must be associated with one or more API stages.

API endpoints - host names APIs in API Gateway, which are deployed to a specific region and of the format: rest-api-id.execute-api.region.amazonaws.com

API key - An alphanumeric string that API Gateway uses to identify an app developer who uses your API.

API stage - A logical reference to a lifecycle state of your API. API stages are identified by API ID and stage name.

Model - Data schema specifying the data structure of a request or response payload.

Private API - An API that is exposed through interface VPC endpoints and isolated from the public internet

Private integration - An API Gateway integration type for a client to access resources inside a customer's VPC through a private API endpoint without exposing the resources to the public internet.

Proxy integration - You can set up a proxy integration as an HTTP proxy integration type or a Lambda proxy integration type.

Usage plan - Provides selected API clients with access to one or more deployed APIs. You can use a usage plan to configure throttling and quota limits, which are enforced on individual client API keys.



Amazon
API Gateway

Amazon API Gateway CLI Commands

Files

main

Go to file

- assets
- finalPresentation
- learning
 - 01_start
 - 02_calc
 - 03_logStream
 - 04_Name
 - 05_Greeting
 - 06_function_with_url
 - 07_product
 - 08_dynamodb
 - 09_S3
 - 10_ApiGateway
- README.md
- app.py
- function.zip
- lambda-role-policy.json

joinCommunity2023 / learning / 10_ApiGateway /

Create a role

We use the [create-role](#) command :

create the lambda role

```
aws iam create-role --role-name apigw-lambda-role-py --assume-role-policy-document 'file://lambda-role-policy.json' -
```

Or

```
aws iam create-role --role-name apigw-lambda-role-py --assume-role-policy-document '{"Version": "2012-10-17", "Statement": {}}
```

Attach the execution policy to it

We use the [attach-role-policy](#) command :

attach the [AWSLambdaBasicExecutionRole](#) policy to the lambda role

attach-role AWSLambdaBasicExecutionRole managed policy:

```
aws iam attach-role-policy --role-name apigw-lambda-role-py --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

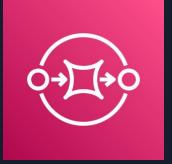
Zip file Linux



Amazon
API Gateway



SNS

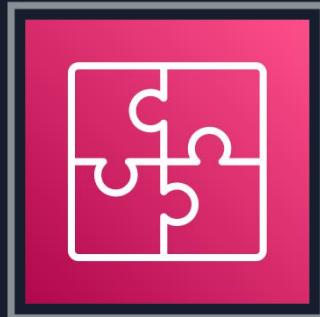


SQS



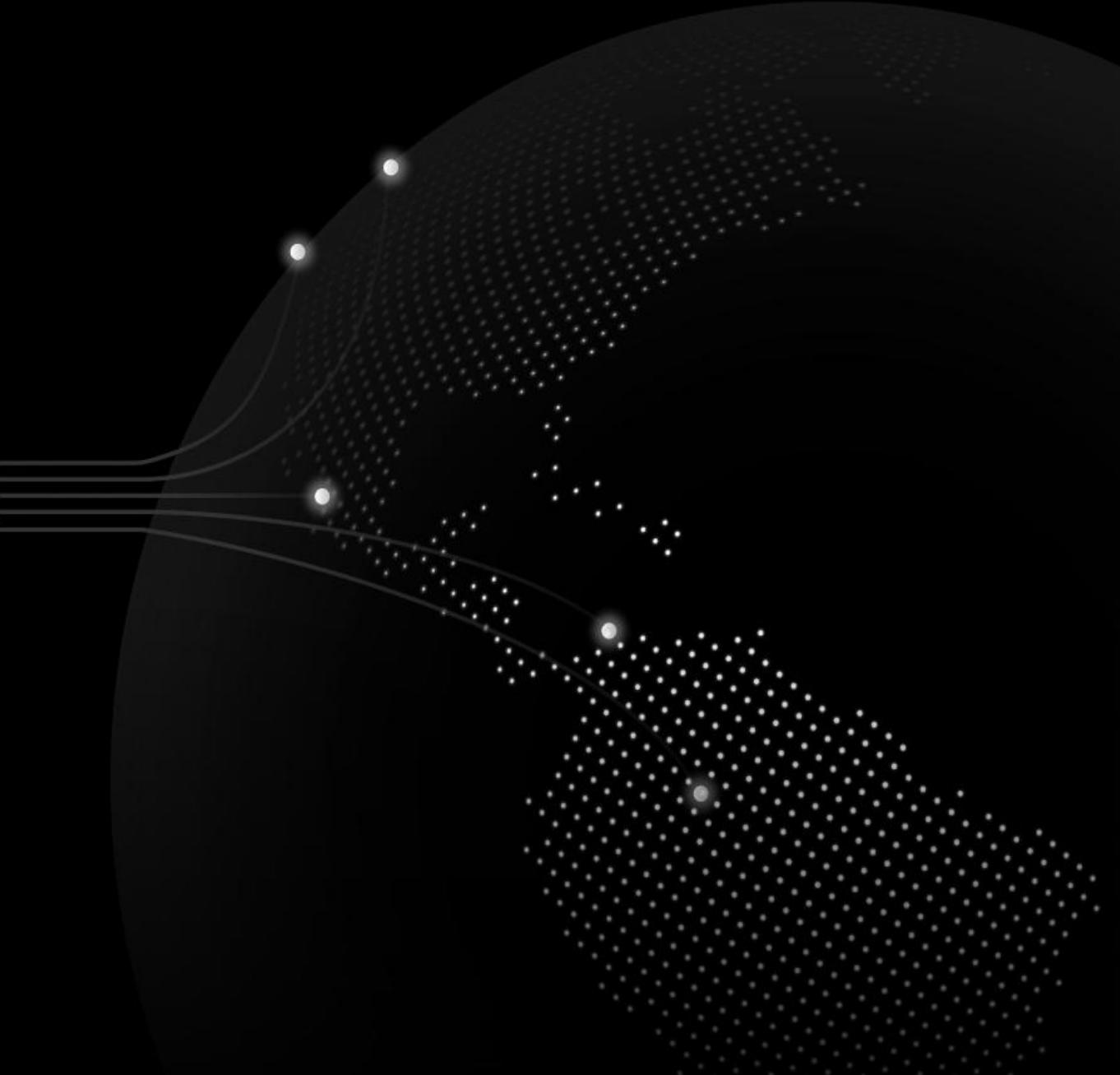
EVENTBRIDGE

AWS Application Integration





Amazon SNS



Amazon SNS: Fully Managed Pub/Sub Messaging



Application
Integration

Application integration

The Fanout scenario is when a message published to an SNS topic is replicated and pushed to multiple endpoints.

Application alerts

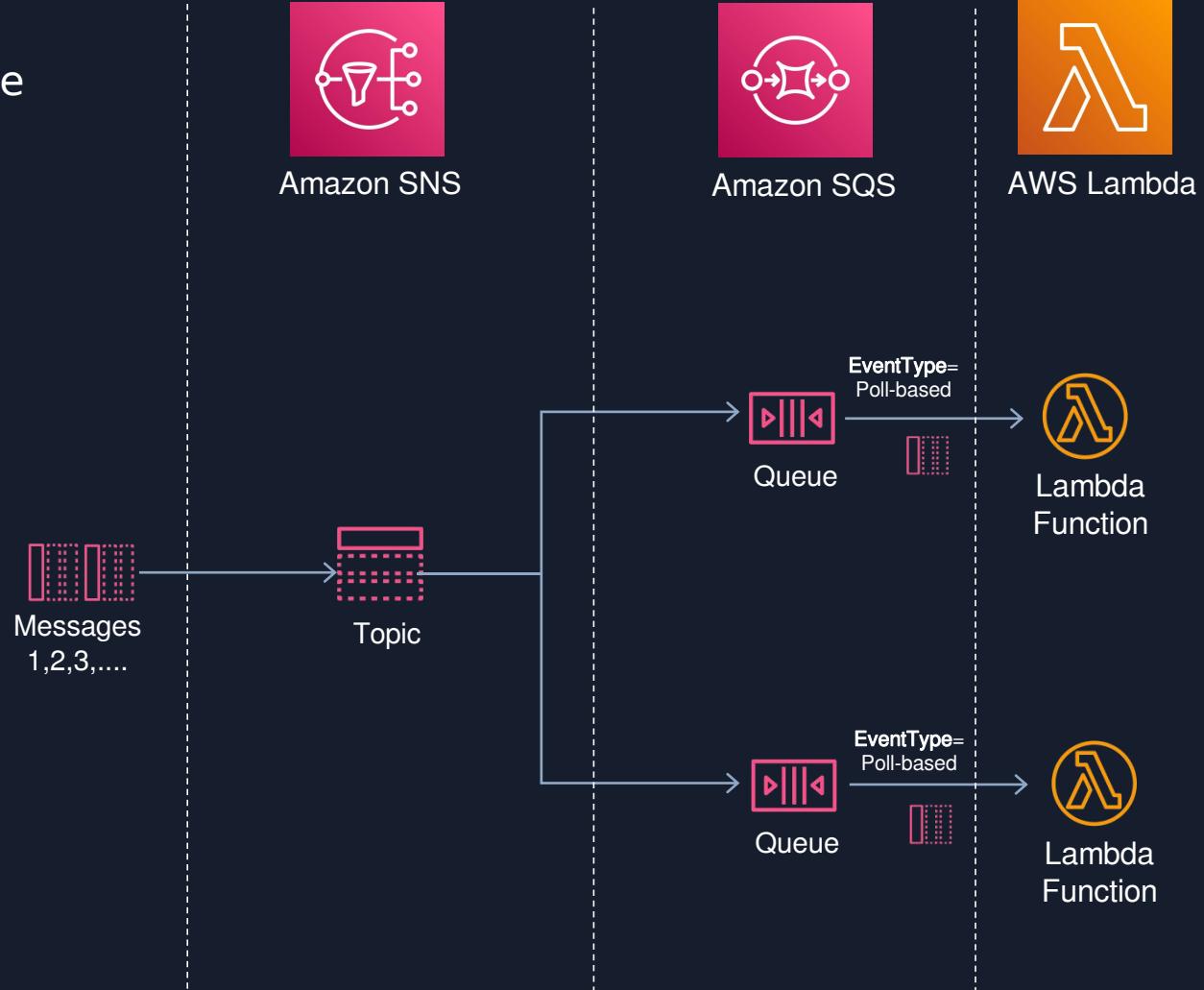
Amazon SNS can send notifications to specified users via SMS and email.

User notifications

Amazon SNS can send push email messages and text messages to individuals or groups.

Mobile push notifications

Mobile push notifications enable you to send messages directly to mobile apps.



Amazon SNS CLI Commands and (Python) Boto3

joinCommunity2023 / learning / 15_lambda sns / README.md 

 weder96 🚀 topics cli

Preview Code Blame 106 lines (75 loc) · 3.5 KB 

Lambda and SNS with Python

Zip file Linux

```
zip function.zip app.py
```



Create Lambda Function

```
aws lambda create-function \
--function-name snsTopic \
--runtime python3.10 \
--zip-file fileb://function.zip \
--handler app.lambda_handler \
--role arn:aws:iam::710304818543:role/lambda-role-py
```



Create a topic

To create a topic, use the sns create-topic command and specify the name to assign to the topic.

```
aws sns create-topic --name my-topic
```



Amazon SNS

https://github.com/weder96/joinCommunity2023/blob/main/learning/15_lambda sns/README.md





Amazon SQS



What is Amazon SQS ?



Application
Integration

Amazon SQS stands for **Simple Queue Service** is **fully managed message queues** for microservices, distributed systems, and serverless applications.

Enables you to **decouple** and **scale microservices**, distributed systems, and serverless applications.

Eliminates the **complexity** and overhead associated with managing and operating message-oriented middleware.

Send, store, and receive messages between software components at any volume.

Two types of message queues.

Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery.

FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.

Integrate and **decouple** distributed software systems and components.



Provides a **generic web services API** that you can access using any programming language that the **AWS SDK** supports.

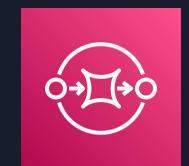
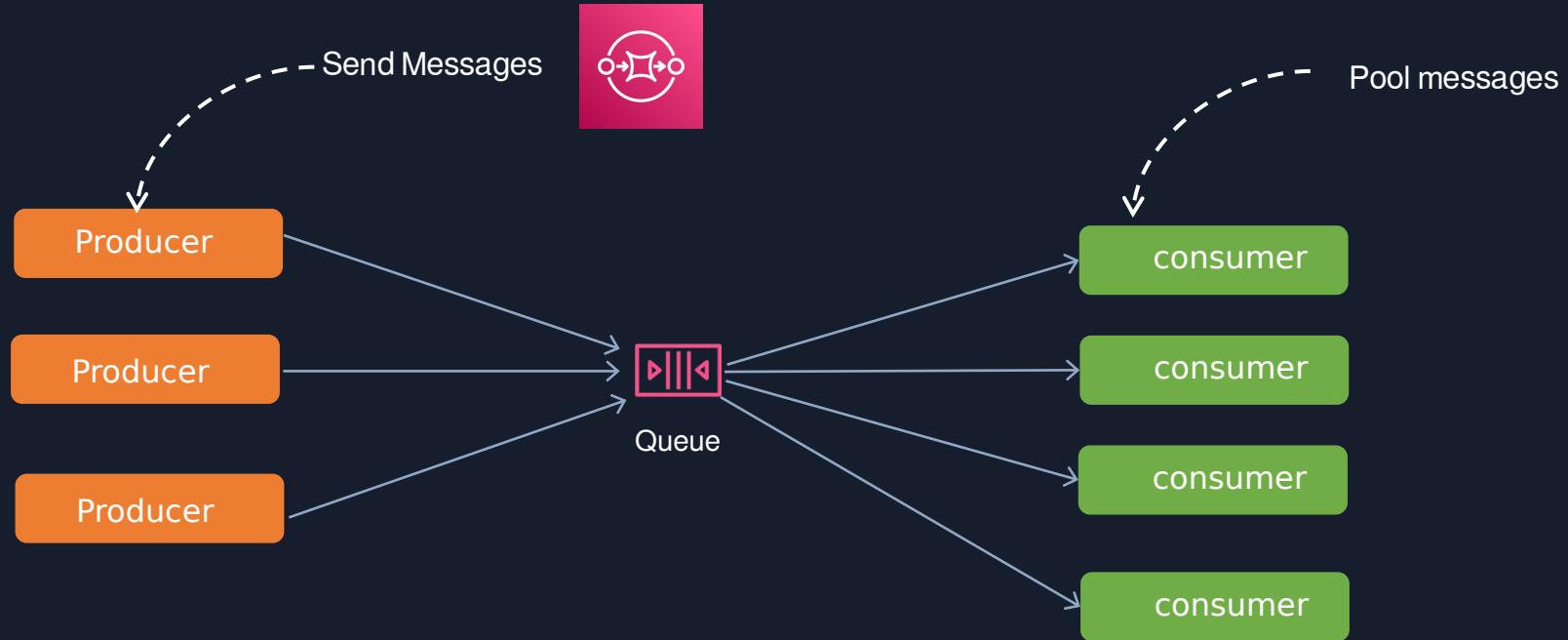
Amazon SQS

https://aws.amazon.com/sns/?nc1=h_ls

Amazon SQS What's a queue?



Application
Integration



Amazon SQS

https://aws.amazon.com/sns/?nc1=h_ls

Amazon SQS CLI Commands and (Python) Boto3



Application
Integration

joinCommunity2023 / learning / 16_lambda_sqs / README.md

Preview | Code | Blame 237 Lines (170 loc) · 6.14 KB ⚡ Code 55% faster with GitHub Copilot

Command Line Interface

```
aws sqs create-queue --queue-name cli-queue-attr --attributes file://q-attributes.json
```

```
aws sqs get-queue-attributes --queue-url https://sqs.us-east-1.amazonaws.com/710304818543/cli-queue-attr --attribute-
```

Sending Messages to our AWS SQS Queue

```
aws sqs send-message --queue-url https://sqs.us-east-1.amazonaws.com/710304818543/cli-queue-attr --message-body "IOT-"
```

Reading Messages from the AWS SQS Queue

```
aws sqs --region us-east-1 receive-message --queue-url https://sqs.us-east-1.amazonaws.com/710304818543/cli-queue-attr
```

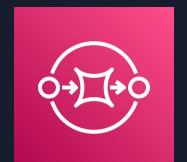
Deletion of a Processed Message from the AWS SQS Queue

```
aws sqs --region us-east-1 delete-message --queue-url https://sqs.us-east-1.amazonaws.com/710304818543/cli-queue-attr
```

Cleaning Up

```
aws sqs --region us-east-1 delete-queue --queue-url https://sqs.us-east-1.amazonaws.com/710304818543/cli-queue-attr
```

Get message with long polling



Amazon SQS



Amazon EventBridge



What is Amazon EventBridge ?

Serverless **event bus** service for AWS services

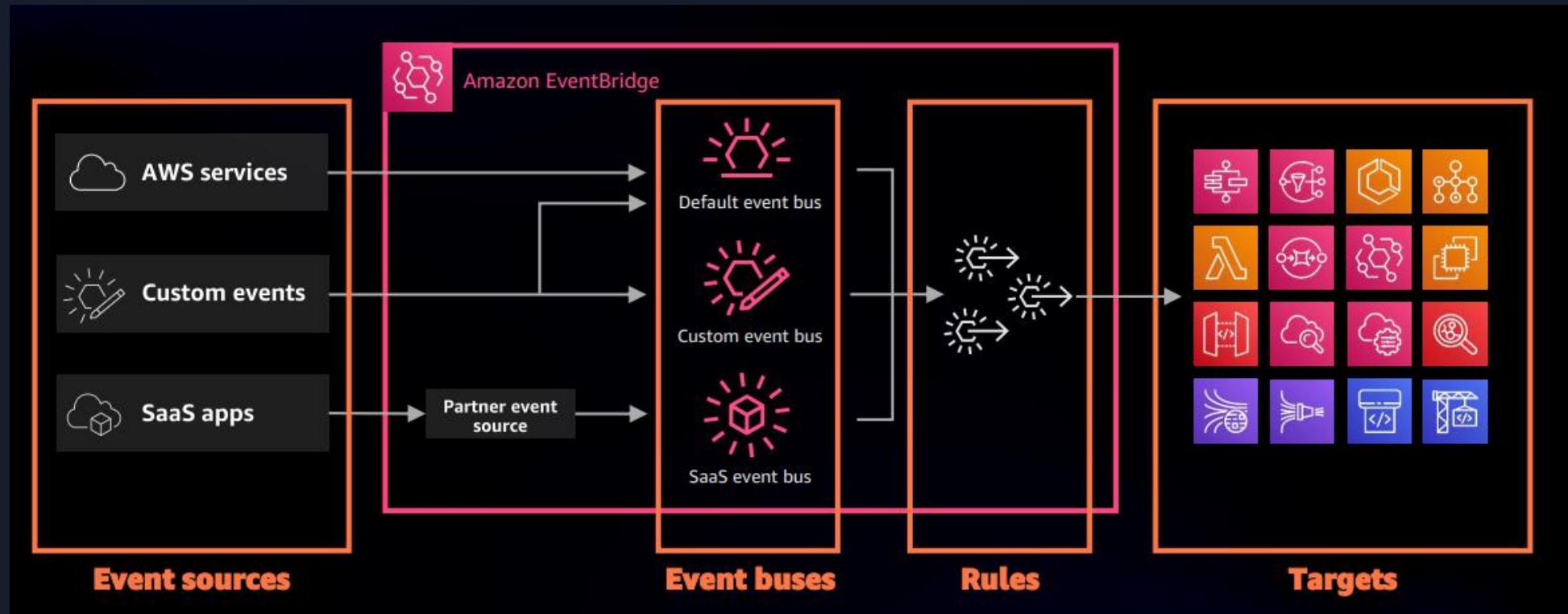
Build **event-driven** applications at scale using events generated from your apps

Use to connect your applications with data from a variety of sources.

Integrated SaaS applications

AWS services to targets such as AWS Lambda functions

Formerly **called Amazon CloudWatch Events**

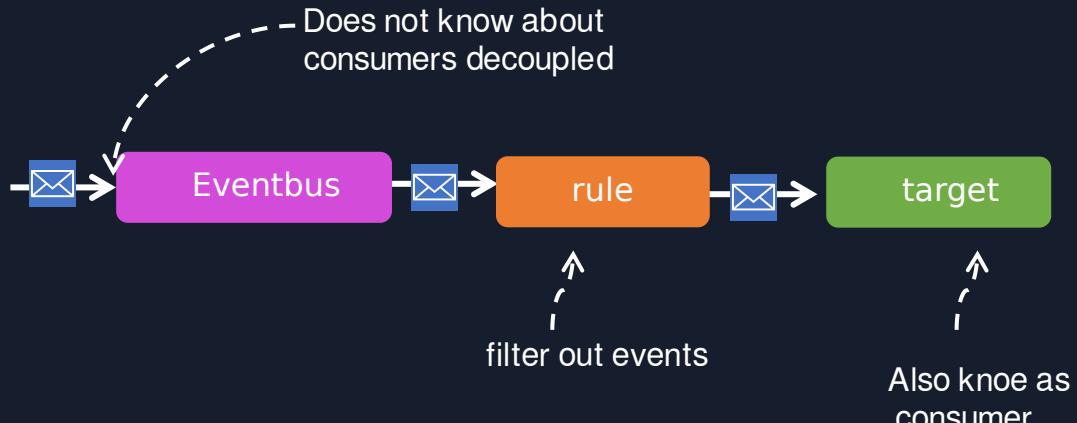


What are events?

1. An event is defined in JSON
2. "Detail" is application specific
3. Envelope attributes are provided by
4. Amazon EventBridge
5. Producers create events
6. Consumers choose which events to
7. Listen to by using rules

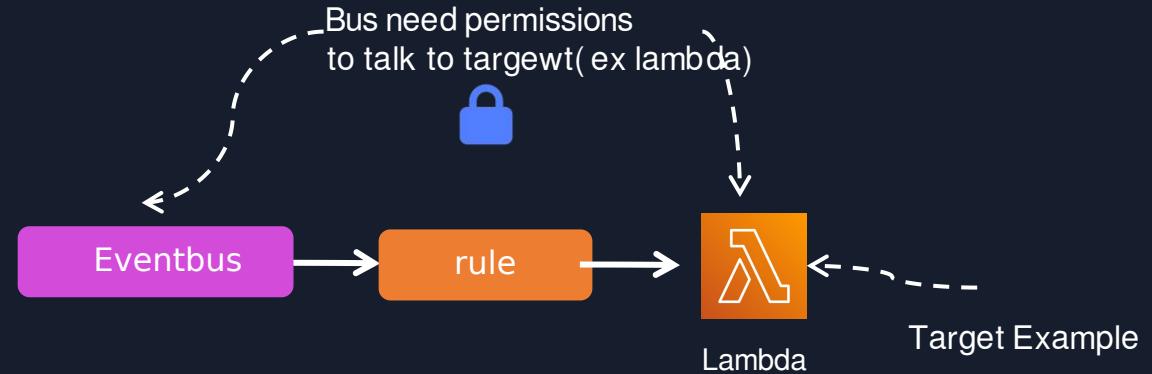
```
{  
  "version": "0",  
  "id": "6ac4e27b-1234-1234-1234-5fb02c880319",  
  "detail-type": "OrderProcessor.OrderStarted",  
  "source": "awsserverlessda.serverlesspresso",  
  "account": "123456789012",  
  "time": "2021-11-28T13:12:30Z",  
  "region": "us-west-2",  
  "detail": {  
    "userId": "jbesw",  
    "orderId": "eYmAfqLD67v1bdUVile_D",  
    "drinkOrder": {  
      "icon": "barista-icons_cafe-latte",  
      "modifiers": [],  
      "drink": "Latte"  
    }  
  }  
}
```

Understanding EventBridge target



What are target ?

Targets are consumer of you events
use rules to process events before reaching targets



Target Permissions

Your eventBridge bus need permissions to send events to targets

Amazon EventBridge CLI Commands and (Python) Boto3

joinCommunity2023 / learning / 19_eventbridge / README.md

weder96 learning serverless with cli commands

Preview Code Blame 201 lines (145 loc) · 4.4 KB Code 55% faster with GitHub Copilot

EventBridge

CLI

Amazon EventBridge

Amazon EventBridge - Developing with AWS SDK

EventBridge Client - AWS SDK for Python Boto3

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/events.html>

Commands : PutEventsCommand PutRuleCommand PutTargetsCommand

Event Bridge CLI Commands

Create an event bus

```
aws events create-event-bus --name "event-bridge-demo"
```

Create Role for EventBridge Event

Create Execution Role for AWS Lambda functions with AWS CLI

```
aws iam create-role --role-name eventbridge-role-py --assume-role-policy-document '{"Version": "2012-10-17", "Statement": [{"Action": "sts:AssumeRole", "Effect": "Allow", "Principal": "events.amazonaws.com"}]}
```



Amazon EventBridge



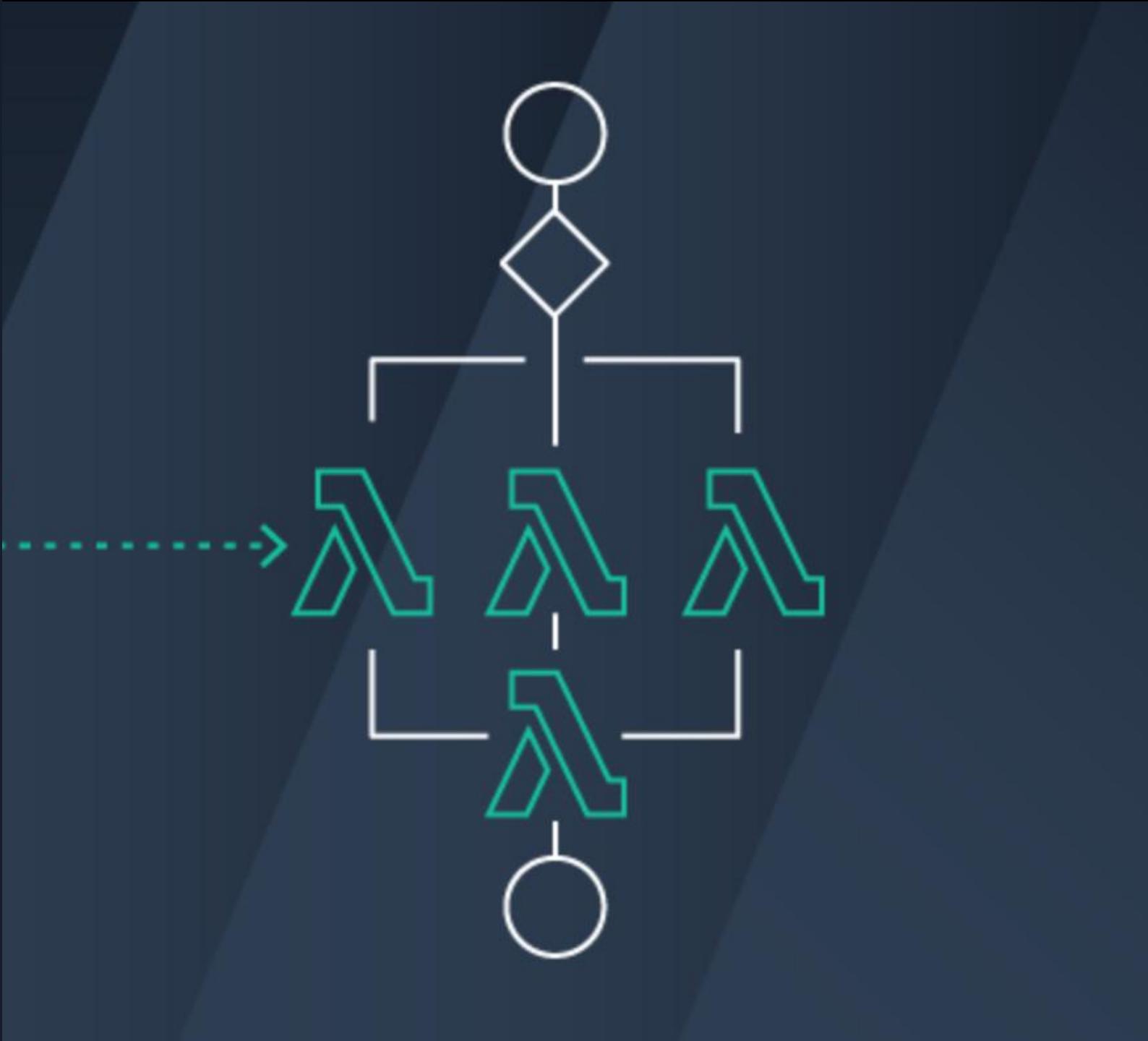
STEP FUNCTIONS

AWS Application Orchestration





AWS Step Functions



State types

Task	A single unit of work	 AWS Lambda Invoke	 Amazon SNS Publish	 Amazon ECS RunTask	 AWS Step Functions StartExecution	 AWS Glue StartJobRun
Choice	Adds branching logic				Choice	Adds if-then-else logic.
Parallel	Fork and join the data across tasks				Parallel	Adds parallel branches.
Wait	Delay for a specified time				Wait	Delays for a specified time.
Fail	Stops an execution and marks it as a failure				Fail	Stops and marks as failure.
Succeed	Stops an execution successfully				Success	Stops and marks as success.
Pass	Passes its input to its output				Pass	Transforms data or acts as placeholder.
Map	Loop and Foreach				Map	Adds a for-each loop.



Create a Serverless Workflow with AWS Step Functions and AWS Lambda



Step Functions > State machines > CallCenterStateMachine > Edit

Edit CallCenterStateMachine

Cancel Start execution Save

Definition
Define your workflow using [Amazon States Language](#). Test your data flow with the new [Data Flow Simulator](#).

Generate code snippet ▾ Format JSON

```
23  {
24      "Variable": "$.Status",
25      "NumericEquals": 1,
26      "Next": "Close Case"
27  },
28  {
29      "Variable": "$.Status",
30      "NumericEquals": 0,
31      "Next": "Escalate Case"
32  }
33  ],
34 },
35 "Close Case": {
36     "Type": "Task",
37     "Resource": "arn:aws:lambda:us-east-
1: :function:CloseCaseFunction",
38     "End": true
39 },
40 "Escalate Case": {
41     "Type": "Task",
42     "Resource": "arn:aws:lambda:us-east-
1: :function:EscalateCaseFunction",
43     "Next": "Fail"
44 },
45 "Fail": {
46     "Type": "Fail",
```

Workflow Studio New

```
graph TD
    Start((Start)) --> OpenCase[Open Case]
    OpenCase --> AssignCase[Assign Case]
    AssignCase --> WorkOnCase[Work on Case]
    WorkOnCase --> IsCaseResolved{Is Case Resolved}
    IsCaseResolved --> EscalateCase[Escalate Case]
    EscalateCase --> CloseCase[Close Case]
    EscalateCase --> Fail[Fail]
    CloseCase --> End((End))
    Fail --> End
```

<https://aws.amazon.com/pt/tutorials/create-a-serverless-workflow-step-functions-lambda/>

Amazon Step Functions CLI Commands and (Python) Boto3

joinCommunity2023 / learning / 21_step_function / README.md

weder96 learning serverless with cli commands

Preview Code Blame 123 lines (85 loc) · 3.23 KB Code 55% faster with GitHub Copilot

Step Functions

CLI

create-state-machine

```
aws stepfunctions create-state-machine \
--name HasCreateCliStateMachine \
--definition "file://stateMachineDefinition.json" \
--role-arn arn:aws:iam::710304818543:role/service-role/StepFunctions-HelloWorld-role-6df6dc83
```

list state machines

```
aws stepfunctions list-state-machines
```

Execute state machines

```
aws stepfunctions start-execution --state-machine-arn arn:aws:states:us-east-1:710304818543:stateMachine:HasCreateCli
```



AWS Step Functions



AWS Lambda Overview



AWS Lambda Overview

Serverless on AWS, Build and run applications without thinking about servers

The **most popular serverless** compute platform that is using millions of customer

Running billions of invocations all over the world

Compute service that runs code without thinking any servers or underlying services

Serverless function that you only **responsible for your actual code.**



https://aws.amazon.com/serverless/?nc1=h_ls
<https://aws.amazon.com/blogs/compute/serverless-icymi-q2-2023/>

AWS Lambda Invocation Types

Triggered lambda functions with different AWS Lambda Invocation Types

AWS Lambda has 3 Invocation Types;

Lambda Synchronous invocation

Lambda Asynchronous invocation

Lambda Event Source Mapping with polling invocation



Lambda Function Code

AWS Lambda runs instances of **your function to process events**.

Invoke function directly **using the Lambda API**, or configure an AWS service or resource to invoke your function.

Lambda function **has code to process the events that you pass** into the function or that other AWS services send to the function with event json object.

The event object contains all the information about the **event that triggered this Lambda**.

The **context object contains info about** the runtime our Lambda function

Return the function **with the results**



```
app.py M X
lambda > 01_start > app.py > ...
1 import json
2
3 def lambda_handler(event, context):
4     print('Starting input lambda function call')
5     print(event)
6
7     return {
8         'statusCode': 200,
9         'body': json.dumps('Hello from Lambda!!!')
10    }
11
```



AWS Lambda Execution Role

AWS Lambda Permissions

Lambda Execution Role
Resource-based policy

Lambda Execution Role

IAM role that Lambda has permissions to assume when invoking lambda function.

Create an execution role when creating a new lambda function, and after that we can also modify the policies **associated with the IAM role**.

if you have additional targets from your lambda function
performing crud operations on DynamoDB table
sending notification to SNS
retrieve messages from queue or streams

Lambda function's execution role required permissions to interact with those AWS services

Grant least privilege access to your Lambda execution role



```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "lambda.amazonaws.com"  
      },  
      "Action": "sts:AssumeRole"  
    }  
  ]  
}
```

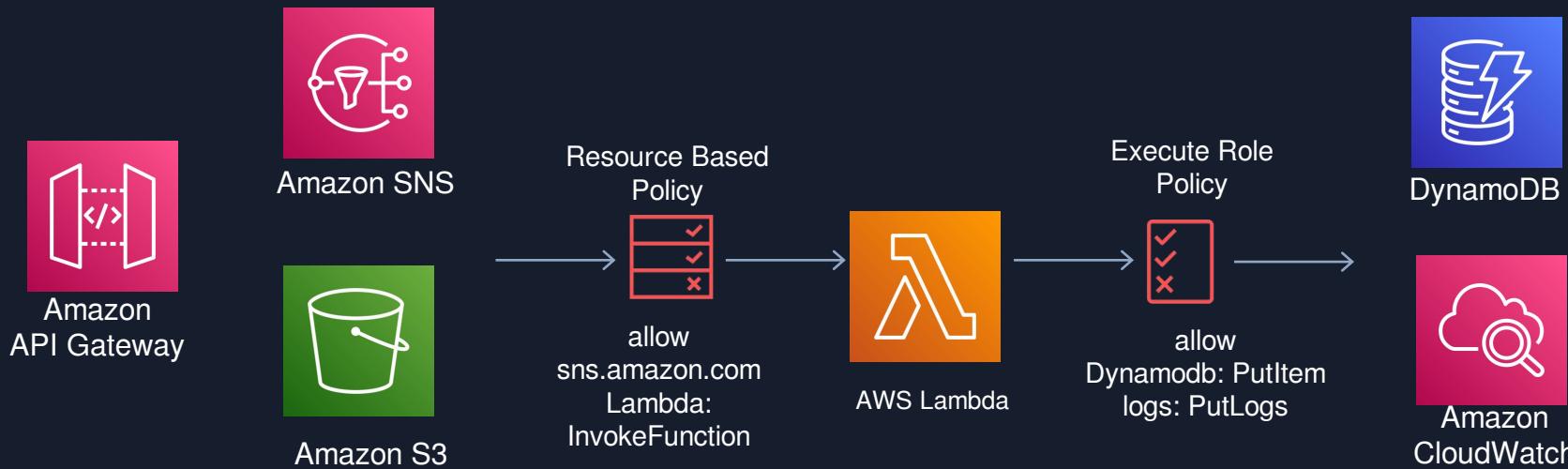
AWS Lambda Resource-based Policy

Lambda Resource-based policy

When any AWS service invokes Lambda function sync or async way. It lets you grant usage permission to other AWS accounts or organizations on a per-resource basis. Also use a resource-based policy to allow an AWS service to invoke your function on your behalf.

API Gateway that targets to Lambda function, we should add resource-based policy permission to invoke lambda function from API gateway.

Amazon S3 upload event triggers to lambda function asynchronously, so we should also add Resource-based policy into our Lambda function grants S3 invocation.



Next Steps



The AWS Step Functions Workshop



The AWS Step Functions Workshop

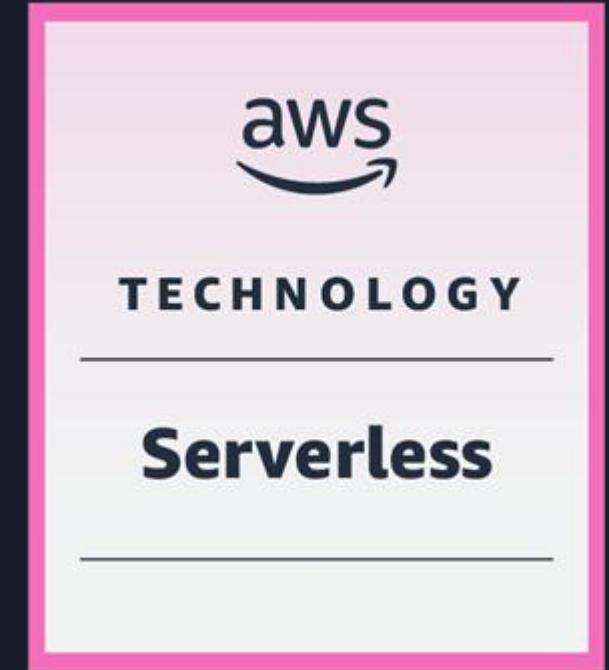


© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

<https://catalog.workshops.aws/stepfunctions/en-US>

AWS Serverless Digital Learning Badges

[AWS Skill Builder](#)



<https://aws.amazon.com/blogs/compute/introducing-new-aws-serverless-digital-learning-badges/>

Want to Try?

Serverless Security Workshop



Serverless Security Workshop

Serverless Security Workshop



Important

The purpose of the workshop is to provide a starter API which **does NOT follow many security best practices** on purpose. The tutorial modules guide you to identify security gaps in the starter app, and implement protection measures for them. Furthermore, the modules **do not cover ALL** the security measures that should be applied. After completing all modules, we recommend you to explore additional protections, such as ensuring the principle of least privilege. See the **Extra Credit** section for more details.

In this workshop, you will learn techniques to secure a serverless application built with AWS Lambda, Amazon API Gateway and RDS Aurora. We will cover AWS services and features you can leverage to improve the security of a serverless applications in 5 domains:

1. identity & access management
2. code
3. data
4. infrastructure
5. logging & monitoring

You'll start by deploying a simple serverless application that allows third party companies to submit unicorn customizations. This will help Wild Rydes receive ad revenue and allow third party companies to market their brand leveraging Wild Rydes's popularity.

<https://catalog.us-east-1.prod.workshops.aws/workshops/026f84fd-f589-4a59-a4d1-81dc543fc30/en-US>

Want to Try?

Serverland AWS Lambda Fundamentals



AWS Lambda Fundamentals



s12d.com/lambda-fundamentals

[https://s12d.com/lambda-
fundamentals](https://s12d.com/lambda-fundamentals)

Want to Try?

Serverless airline - Multiple patterns/practices



The image displays four sequential screenshots of a mobile application interface for a flight booking service:

- Screenshot 1: Search Screen**
Title: Flight App
Text: "Where next?"
Fields: "Departure airport: LGW", "Arrival airport: MAD", "Pick a date: Wed, 24 Apr 2019".
Button: "SEARCH FLIGHTS >"
- Screenshot 2: Flight Selection Screen**
Title: Flight App
Text: "Select your flight"
List:
 - DEPARTURE LGW London Gatwick 16 JAN 2019 MAD Madrid Barajas
Time: 08:00 Duration: 2h15m End: 11:15
Price: 400 EUR Flight No: #1812
 - DEPARTURE LGW London Gatwick 16 JAN 2019 MAD Madrid Barajas
Time: 10:30 Duration: 2h15m End: 13:45
Price: 200 EUR Flight No: #1813
 - DEPARTURE LGW London Gatwick 16 JAN 2019 MAD Madrid Barajas
Time: 12:00 Duration: 2h15m End: 15:15
Price: 1000 EUR Flight No: #1814
- Screenshot 3: Review Selection Screen**
Title: Flight App
Text: "Review your selection"
List:
 - DEPARTURE LGW London Gatwick 16 JAN 2019 MAD Madrid Barajas
Time: 08:00 Duration: 2h15m End: 11:15
Price: 400 EUR Flight No: #1812
- Screenshot 4: User Profile Screen**
Title: Flight App
Text:
 - Heitor F. Lessa (purple)
 - 4,554,234 Points
 - 10% Next Tier Progress

Payment details:
Name: Name on card
Country:
Postcode: Postcode
Card number: 1234 1234 1234 1234
Expiry date: MM / YY
CVC: CVC

Preferences:
Dietary requirements
Luggage

Buttons: "SIGN OUT"

<https://github.com/aws-samples/aws-serverless-airline-booking>

Want to Try? CDK - Workshop



AWS CDK
Workshop

Search

English

Prerequisites

- TypeScript Workshop
- Python Workshop
- .NET Workshop
- Java Workshop
- Go Workshop
- Construct Hub
- Congrats!

Submit a correction

CDK on GitHub

CDK on StackOverflow

CDK on Gitter

Welcome Developers!

Hey there, and thanks for joining us! Hope you can't wait to play with this new thing we call the "AWS Cloud Development Kit" or in short, the AWS CDK.

The AWS CDK is a new software development framework from AWS with the sole purpose of making it fun and easy to define cloud infrastructure in your favorite programming language and deploy it using AWS CloudFormation.

So what are we going to build? Nothing too fancy...

We'll spend some time setting up your development environment and learning a little about how to work with the CDK Toolkit to deploy your app to an AWS environment.

Then, you'll write a little "Hello, world" Lambda function and front it with an API Gateway endpoint so users can call it via an HTTP request.

Next, we'll introduce the powerful concept of **CDK constructs**. Constructs allow you to bundle up a bunch of infrastructure into reusable components which anyone can compose into their apps. We'll walk you through writing your own construct.



<https://cdkworkshop.com/>

Want to Try?

CDK Patterns



aws
CDK Patterns

About

Find A Pattern

CDK Patterns is more than "just AWS CDK examples"

Check Out Our 4 Content Distribution Platforms:



Star 2,130

Follow

YouTube

The Practical Dev

[https://cdkpatterns.co
m/](https://cdkpatterns.co)

Want to Try?

hands-on.cloud

The screenshot shows the homepage of hands-on.cloud. At the top left, there are social media icons for Facebook, YouTube, and Twitter. In the top center is the hands-on.cloud logo, which consists of the text "hands-on.cloud" with a small blue cloud icon above the "o". To the right of the logo is a search bar with the placeholder text "serverless". Below the header is a navigation bar with links: "Python Boto3 Tutorials", "Courses (In progress)", "AWS Certifications", "How-To Articles", "Write For Us", and a "Logout" button. The main content area features a large, semi-transparent background image of a person's hands typing on a keyboard. Overlaid on this image is the text "Hands-On.Cloud" in a large, bold, dark blue font. Below it is the subtitle "Tutorials, How-Tos for Cloud Engineers" and the word "serverless" in a large, bold, dark blue font. At the bottom of the page is a white footer bar containing a "SERVERLESS" button and a magnifying glass icon in a blue search bar.

<https://hands-on.cloud/>

Resources

<https://cdkworkshop.com>

<https://github.com/aws-samples/aws-cdk-examples>

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

<https://aws.amazon.com/pt/developer/language/java/>

<https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/setup-toolkit.html>

<https://aws.amazon.com/pt/intellij/>

https://docs.aws.amazon.com/code-library/latest/ug/java_2_code_examples.html

https://docs.aws.amazon.com/pt_br/prescriptive-guidance/latest/patterns/deploy-a-ci-cd-pipeline-for-java-microservices-on-amazon-ecs.html

<https://docs.aws.amazon.com/lambda/latest/dg/lambda-java.html>

<https://aws.amazon.com/pt/blogs/compute/java-17-runtime-now-available-on-aws-lambda/>

<https://www.slideshare.net/AmazonWebServices/java-on-aws>

<https://www.jrebel.com/blog/aws-java-application-setup>

<https://www.slideshare.net/VadymKazulkin/adopting-java-for-the-serverless-world-at-jax-2022>

<https://towardsaws.com/deploy-spring-boot-application-to-aws-ec2-using-docker-f359e7ad2026>

<https://aws.amazon.com/pt/blogs/developer/stepfunctions-fluent-api/>

<https://aws.amazon.com/blogs/compute/java-17-runtime-now-available-on-aws-lambda/>

<https://docs.aws.amazon.com/lambda/latest/dg/snapstart.html>

About the Speaker

Weder Mariano de Sousa

Specialist Senior Java - GFT

Technician **System Development**

Graduated **Computer Science**

Post Graduate in **Midias UFG**

Post Graduate in **Information Security**



**AWS User
Group Anápolis
Leader**



**GOJava
Leader**

Q & A



AWS community builder
Serverless

- <https://www.linkedin.com/in/wedermarianodesousa/>
- <https://github.com/weder96>
- <https://twitter.com/weder96>
- <https://dev.to/weder96>





AWS User Groups

Anápolis, Brazil

THANK YOU