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**

AWS User Group Anápolis Leader

GOJava Leader

https://www.linkedin.com/in/wedermarianodesousa/

https://github.com/weder96

https://twitter.com/weder96

https://dev.to/weder96

AWS Certified Cloud Practitioner – Foundational

AWS Certified Solutions Architect – Associate

AWS Certified Developer – Associate

AWS Community Builder – Serverless

AWS Educate – Getting Started with Storage

AWS Cloud Economics Accredited – Partner

AWS Technical Accredited – Partner

AWS Business Accredited – Partner

AWS Technology – File Storage – Specialized

AWS Technology – Storage Data Migration – Specialized

AWS Technology – Storage Data Protection & Disaster Recovery – Specialized

AWS Well-Architected – Proficient

AWS Technology – Amazon Braket

AWS Technology – Architecting

AWS Technology – Migration Foundations

AWS Technology – Amazon EKS

AWS Technology – Networking Core

AWS Technology – Serverless

AWS Technology – Events and Workflows

AWS Technology – Compute

AWS Technology – Cloud Essentials

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]

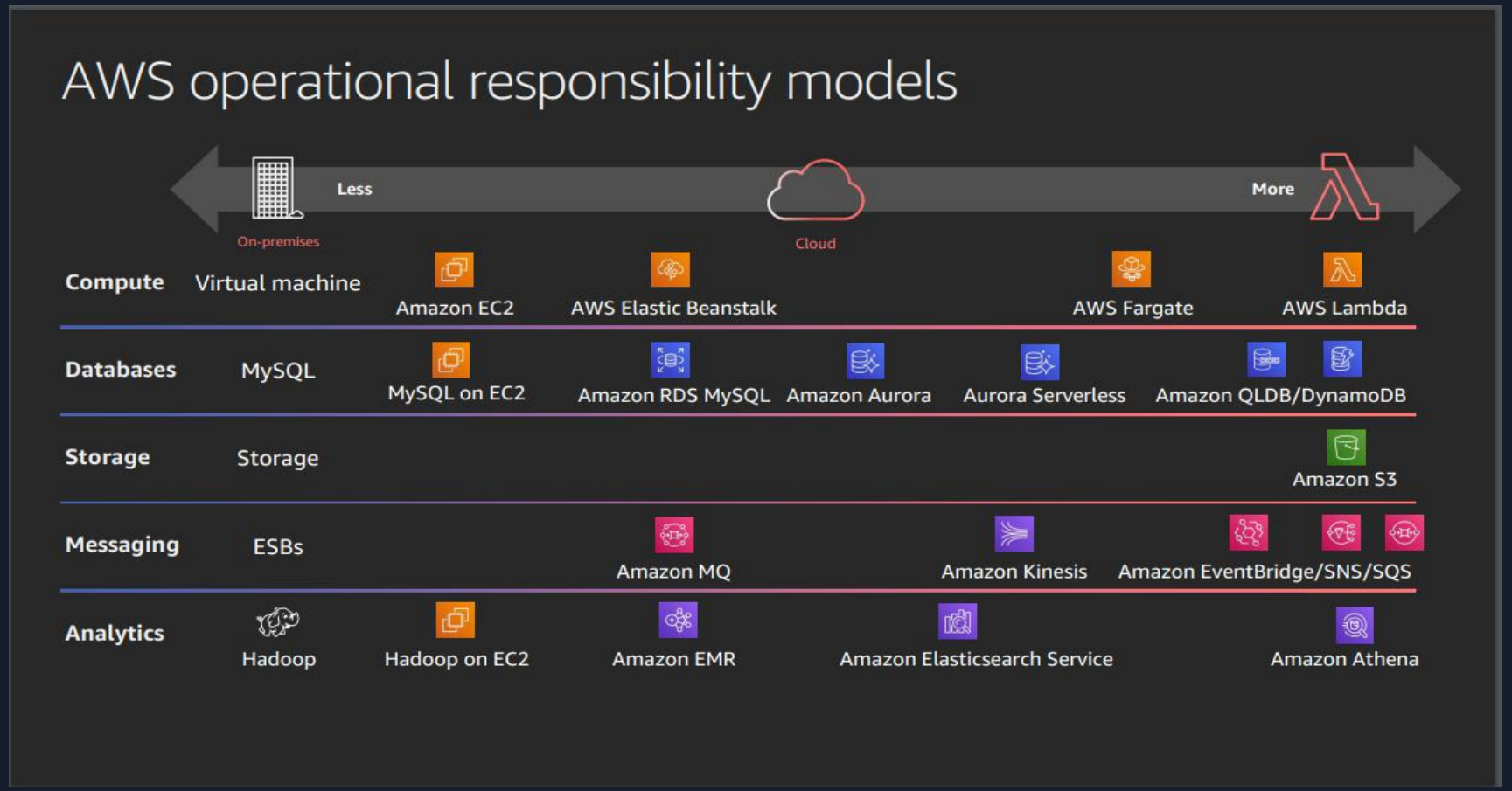
https://aws.amazon.com/serverless/?nc1=h_ls

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 Explanained : AWS Operational Responsibility Model



https://d1.awsstatic.com/events/reinvent/2019/REPEAT_3_Serverless_architectural_patterns_and_best_practices_ARC307-R3.pdf

Comparison of operational responsibility

	AWS Manages	Customer Manages
AWS Lambda Serverless functions	Data source integrations – Physical hardware, software, networking, and facilities – Provisioning	– Application code
AWS Fargate Serverless containers	– Container orchestration, provisioning – Cluster scaling – Physical hardware, host OS/kernel, networking, and facilities	– Application code – Data source integrations – Security config and updates, network config, management tasks
ECS/EKS Container-management as a service	– Container orchestration control plane – Physical hardware software, networking, and facilities	– Application code = Data source integrations –Work clusters –Security config and updates, network config, firewall, management tasks
EC2 Infrastructure-as-a-Service	– Physical hardware software, networking, and facilities	– Application code – Data source integrations – Scaling – Security config and updates, network config, management tasks – Provisioning, managing scaling and patching of servers

AWS Services Serverless

Compute

AWS Lambda

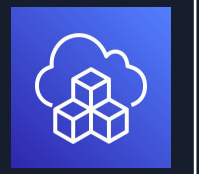
AWS Fargate

Authentication and Authorization

Amazon Cognito

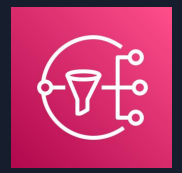
Serverless Deployment Frameworks

AWS CloudFormation

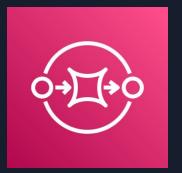


AWS CDK

Application Integrations Services



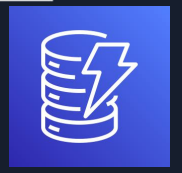
Amazon SNS



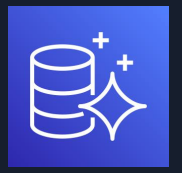
Amazon SQS

Amazon EventBridge

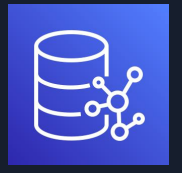
Data Persistence



DynamoDB



Aurora Serverless



Neptune Serverless

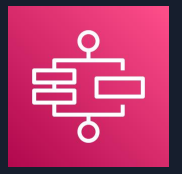
RDS proxy instance

Research and Analysis

Amazon Redshift

OpenSearch Service

Application Orchestration



AWS Step Functions

Storing & Hosting

Amazon S3

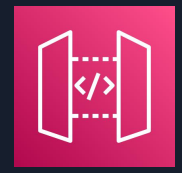
Amazon EFS

Streams

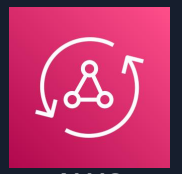
DynamoDB Stream

Kinesis Data Streams

API and AppSync



Amazon API Gateway



AWS AppSync

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

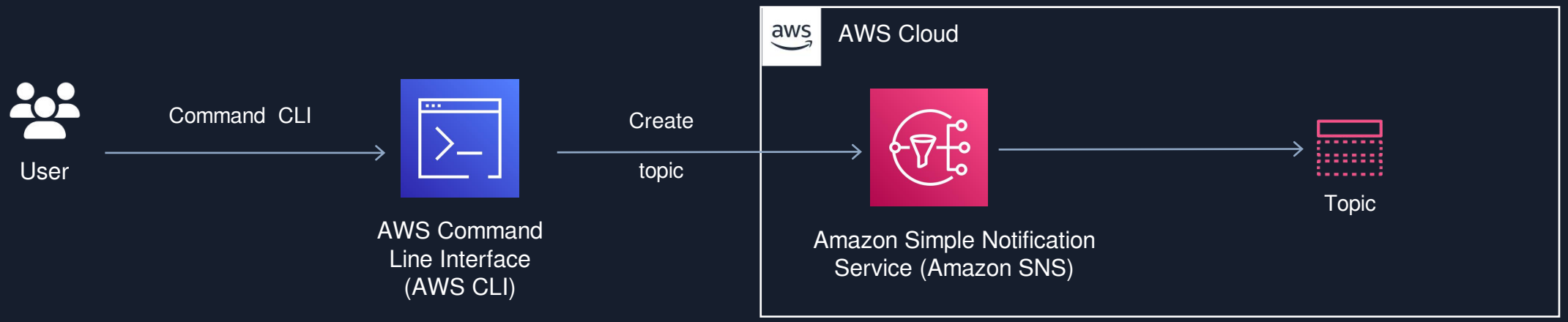
Why Serverless Simplifies Cloud Solutions ?

Why Serverless Simplifies Cloud Solutions ?

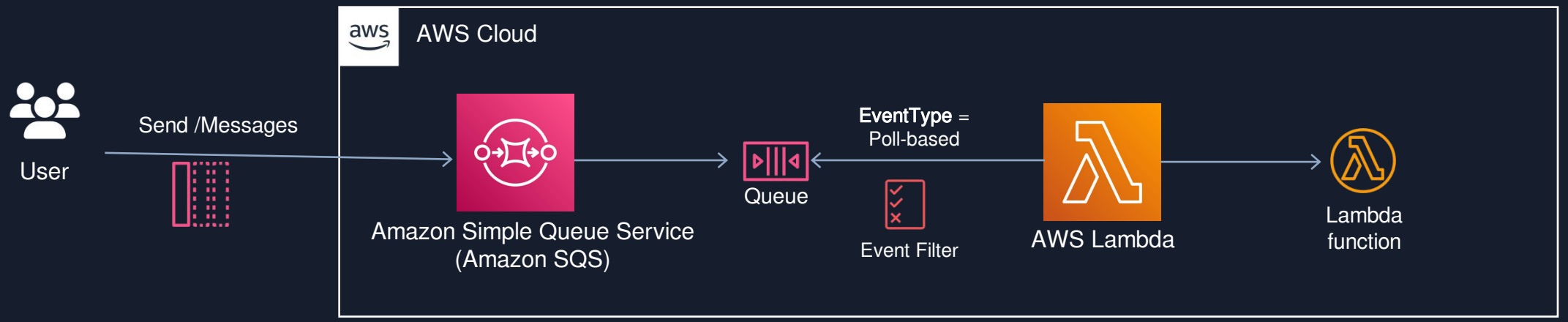
- Reduced Infrastructure Management
- Automatic Scalability
- Consumption-Based Billing
- Accelerated Development
- Less Operational Overhead
- Flexibility in Language and Tool Choice
- Simple Deployment
- Easy Integration
- Resiliency and High Availability
- Focus on business logic
- Reduced Time to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

Architecture Serverless

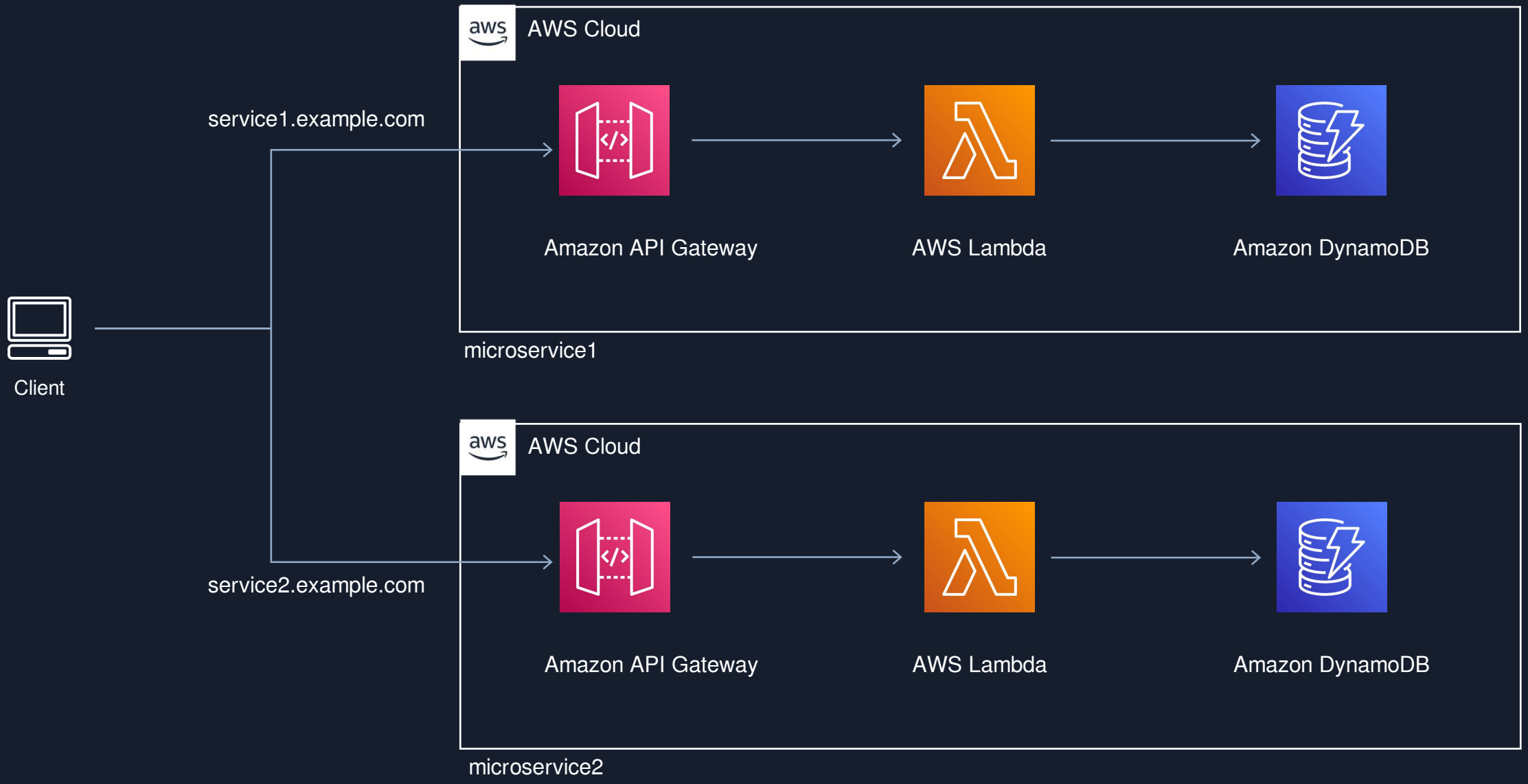
Amazon SNS with AWS Management Console



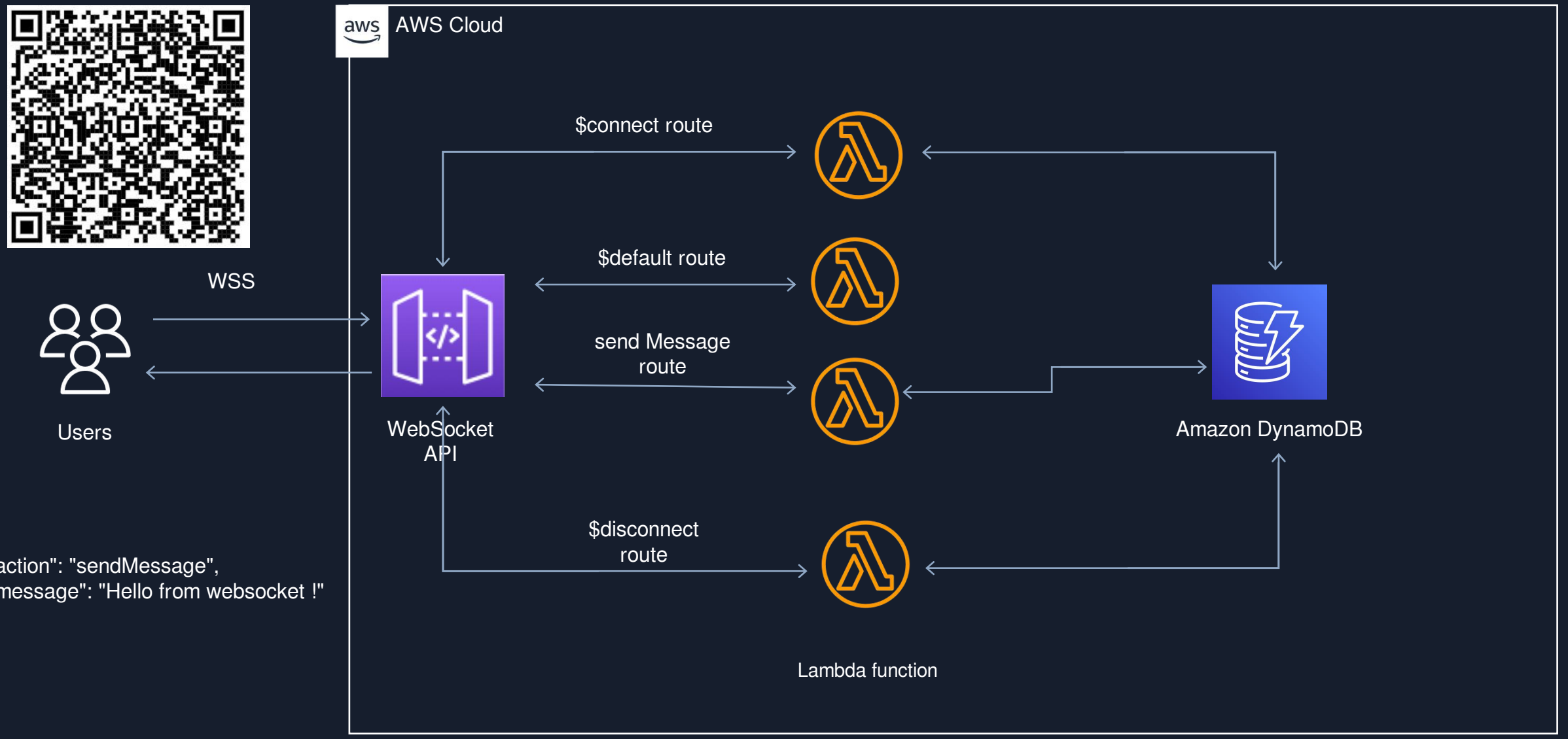
Amazon SQS Queue Polling From AWS Lambda



Microservices “REST”



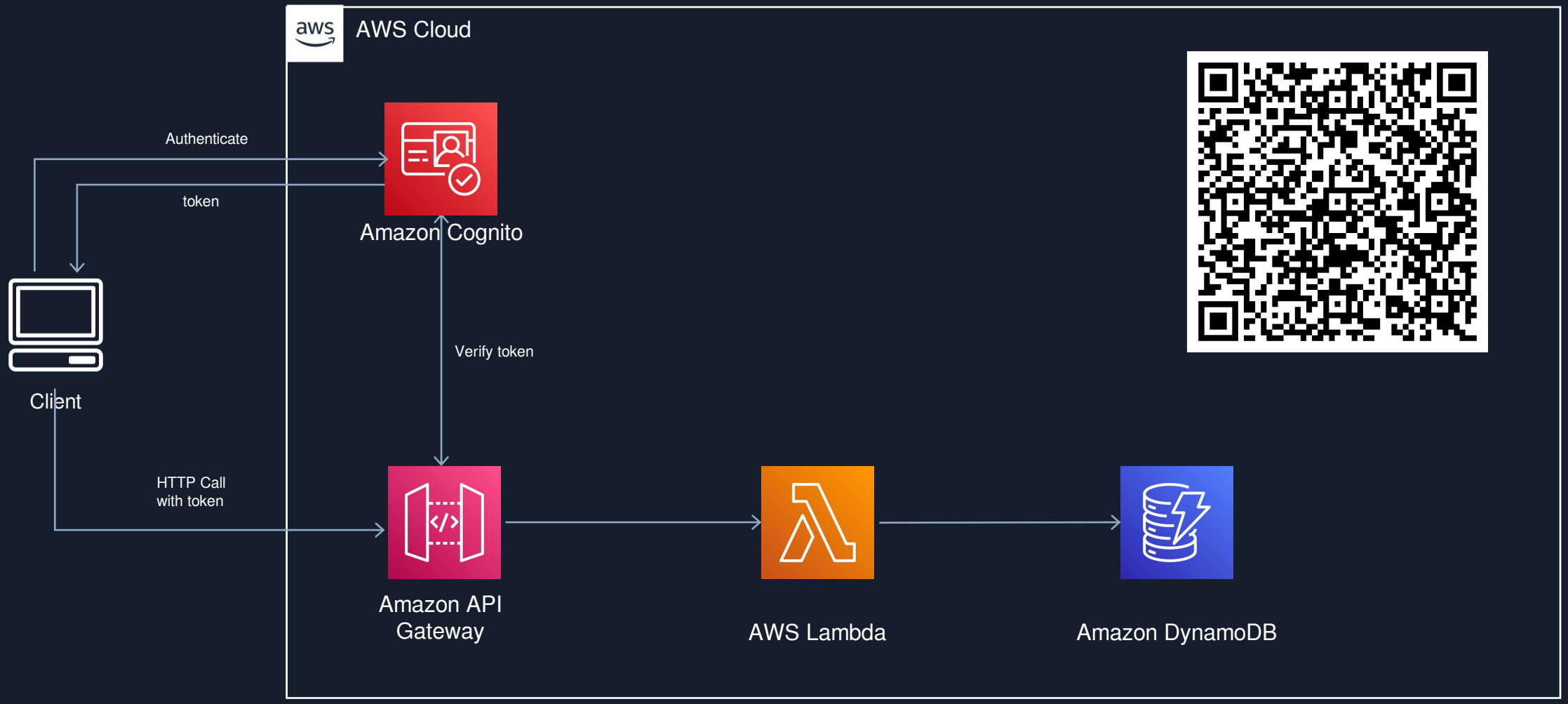
Build Serverless Chat App with a WebSocket API and Lambda



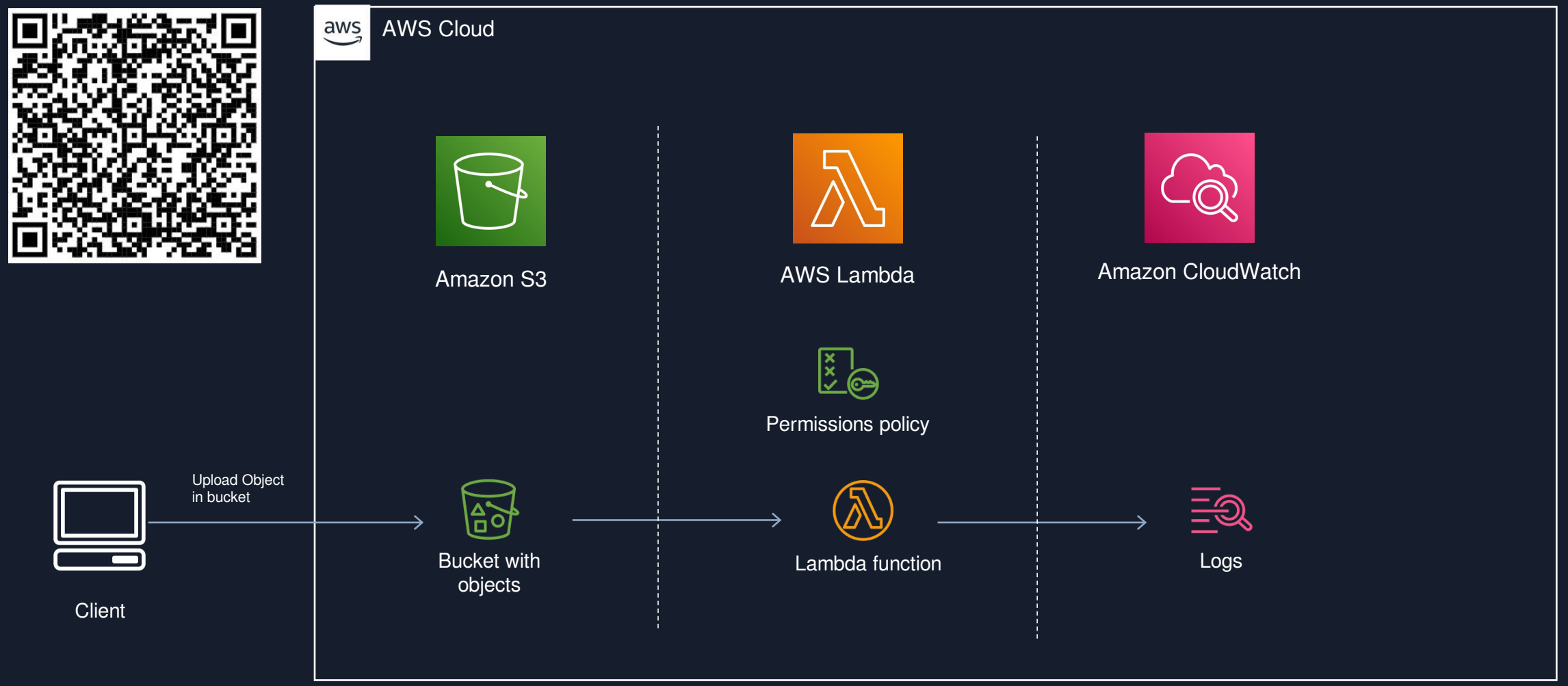

```
{
  "action": "sendMessage",
  "message": "Hello from websocket !"
}
```

https://docs.aws.amazon.com/apigateway/latest/developerguide/websocket-api-chat-app.html

Secure your API Gateway with Amazon Cognito User Pools

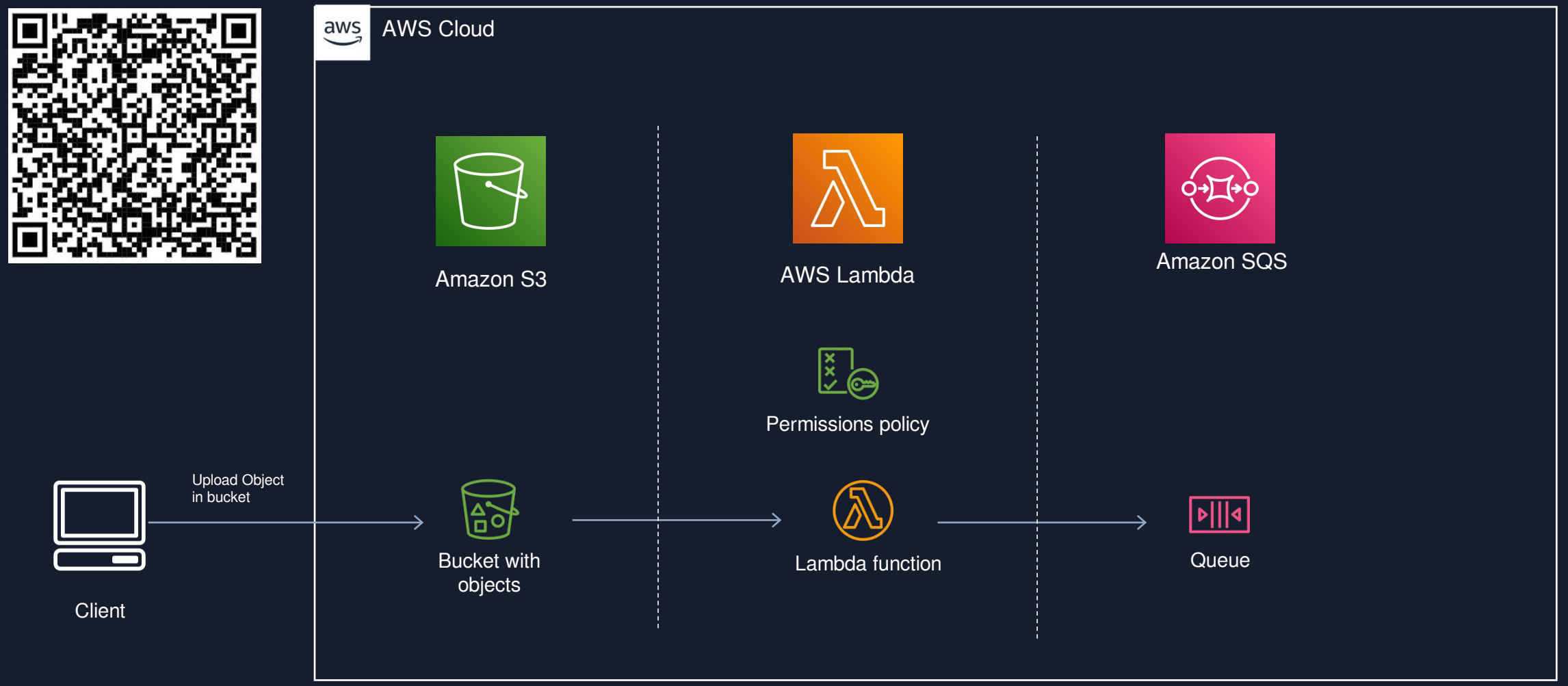


Using an Amazon S3 trigger to invoke a Lambda function



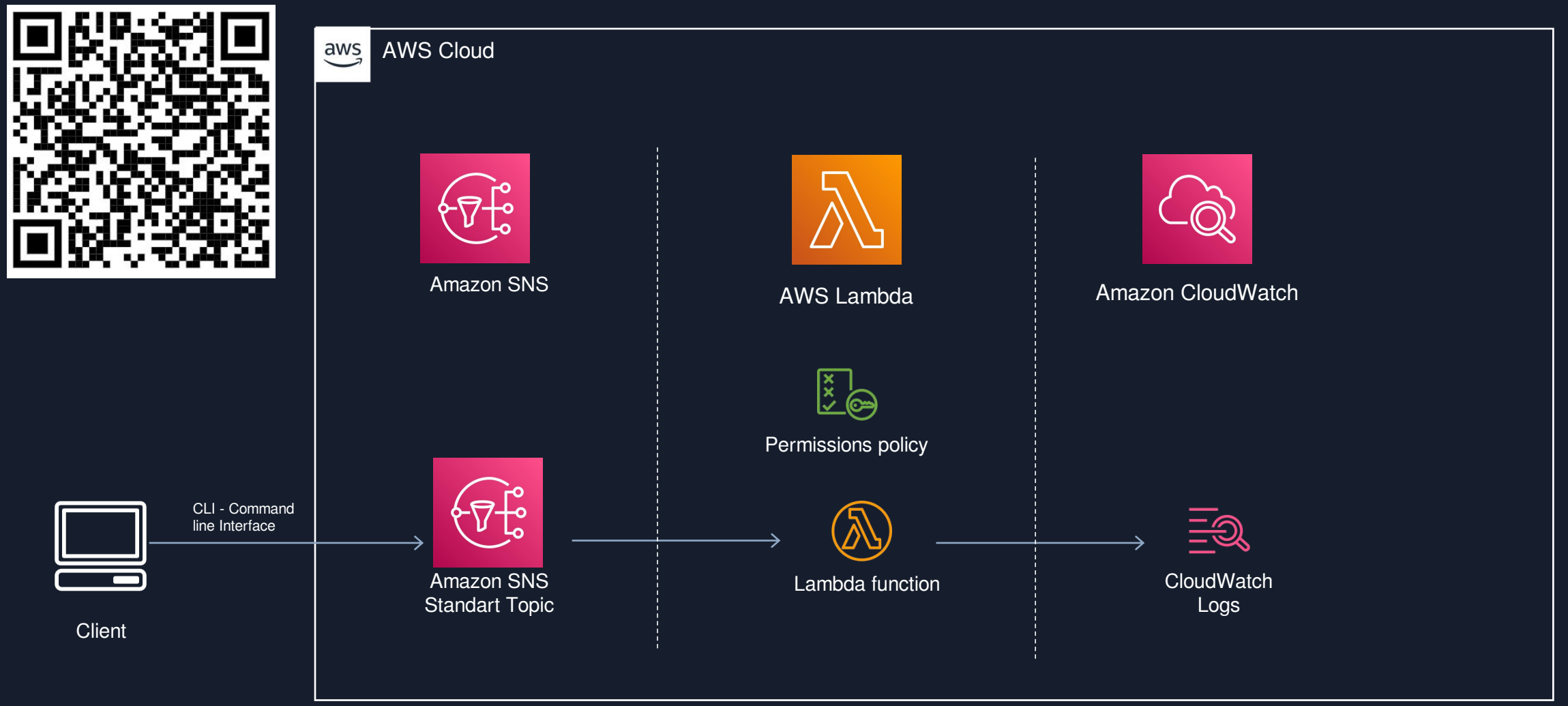
https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html

AWS Lambda Destination to SQS - DLQ Case



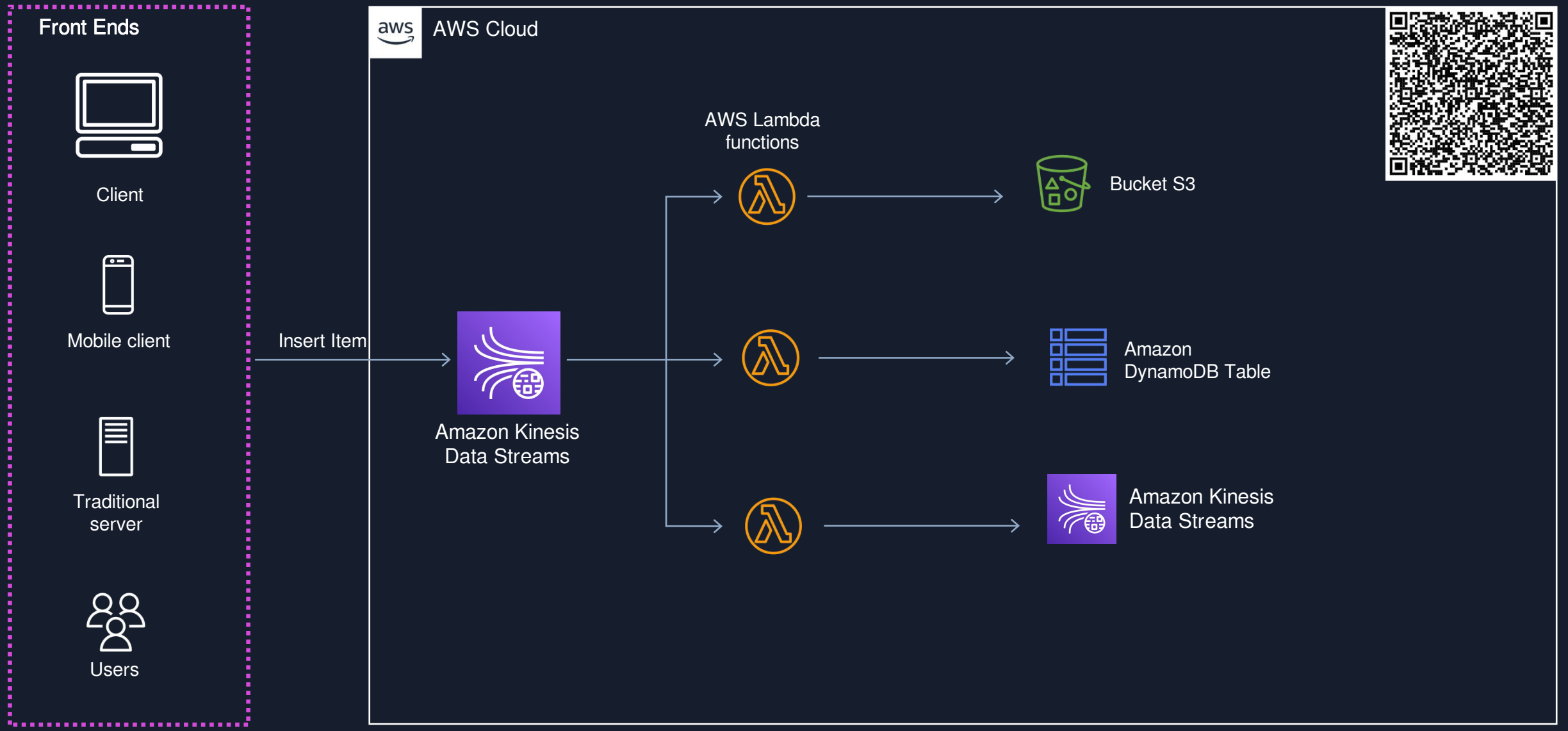
https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html

Using an Amazon SNS to send message to Lambda function



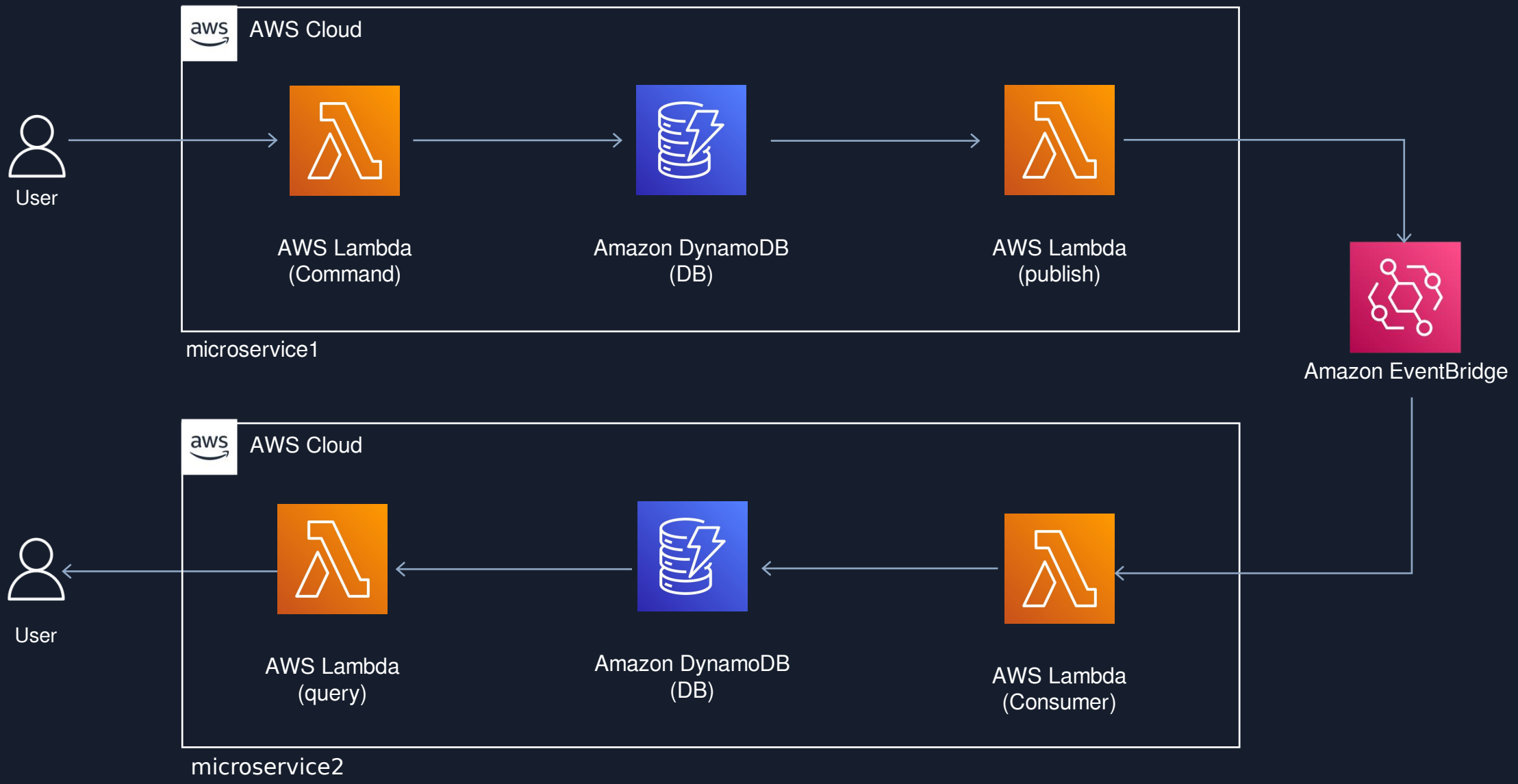
https://docs.aws.amazon.com/lambda/latest/dg/with-sns-example.html

Processing performance with Amazon Kinesis Data Streams and AWS Lambda

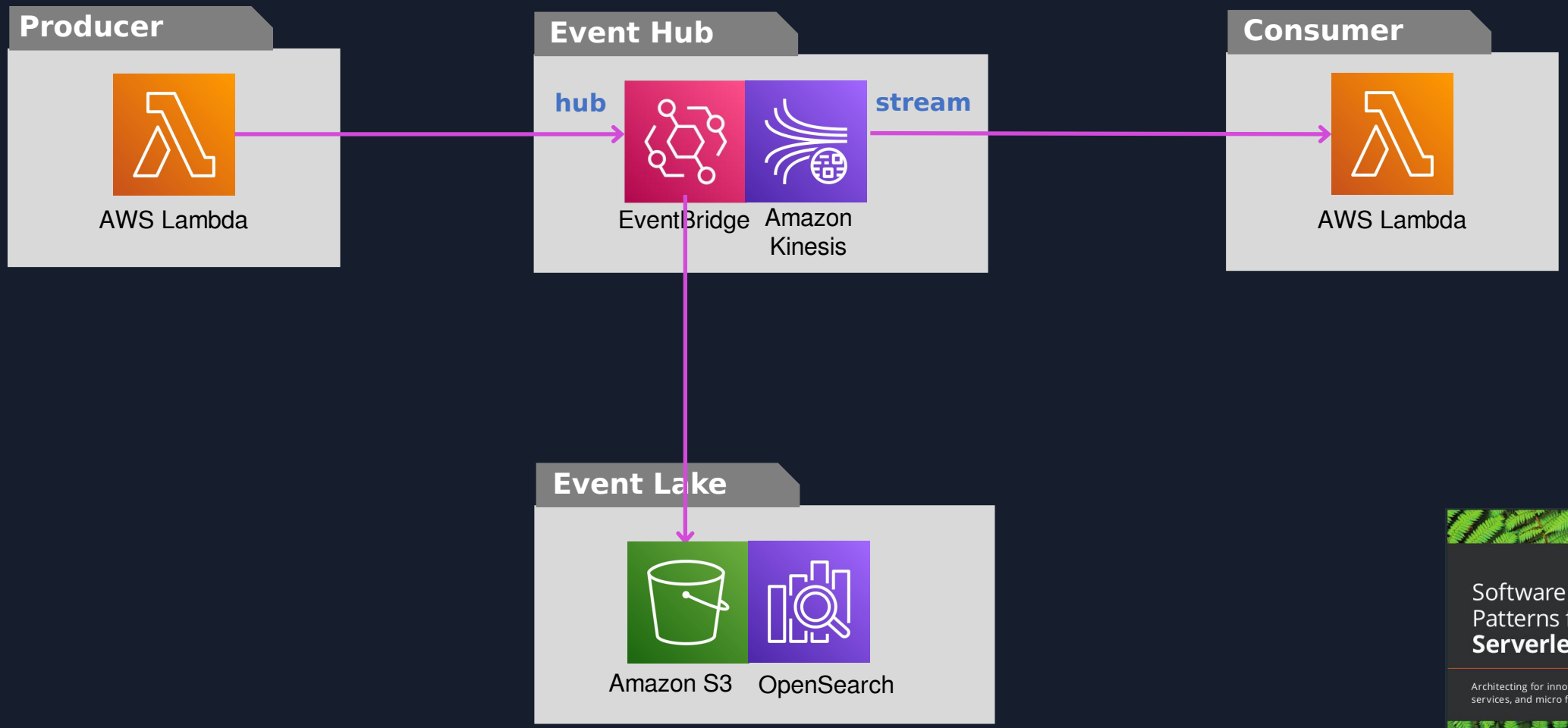


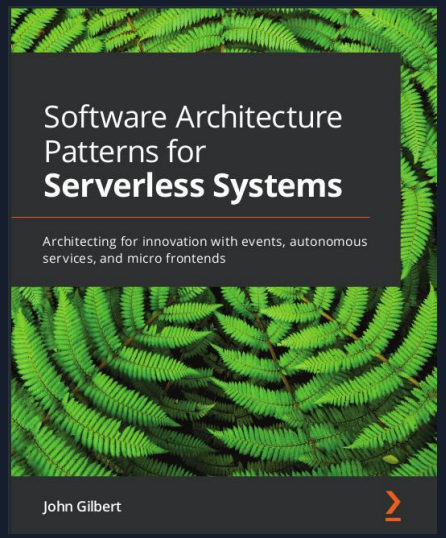
https://aws.amazon.com/blogs/compute/increasing-real-time-stream-processing-performance-with-amazon-kinesis-data-streams-enhanced-fan-out-and-aws-lambda/

Command, Publish, Consume, Query (CPCQ)

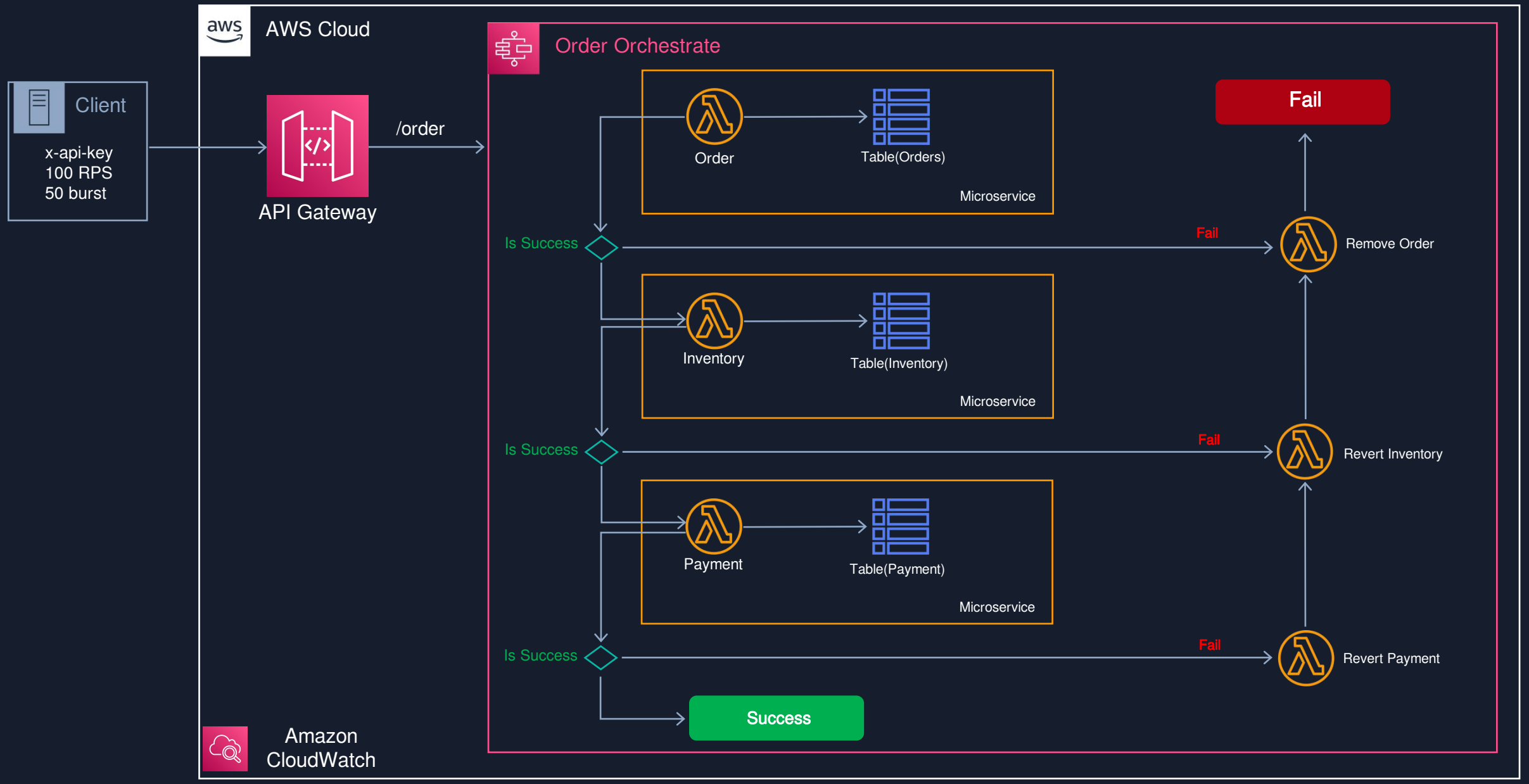


Event hub

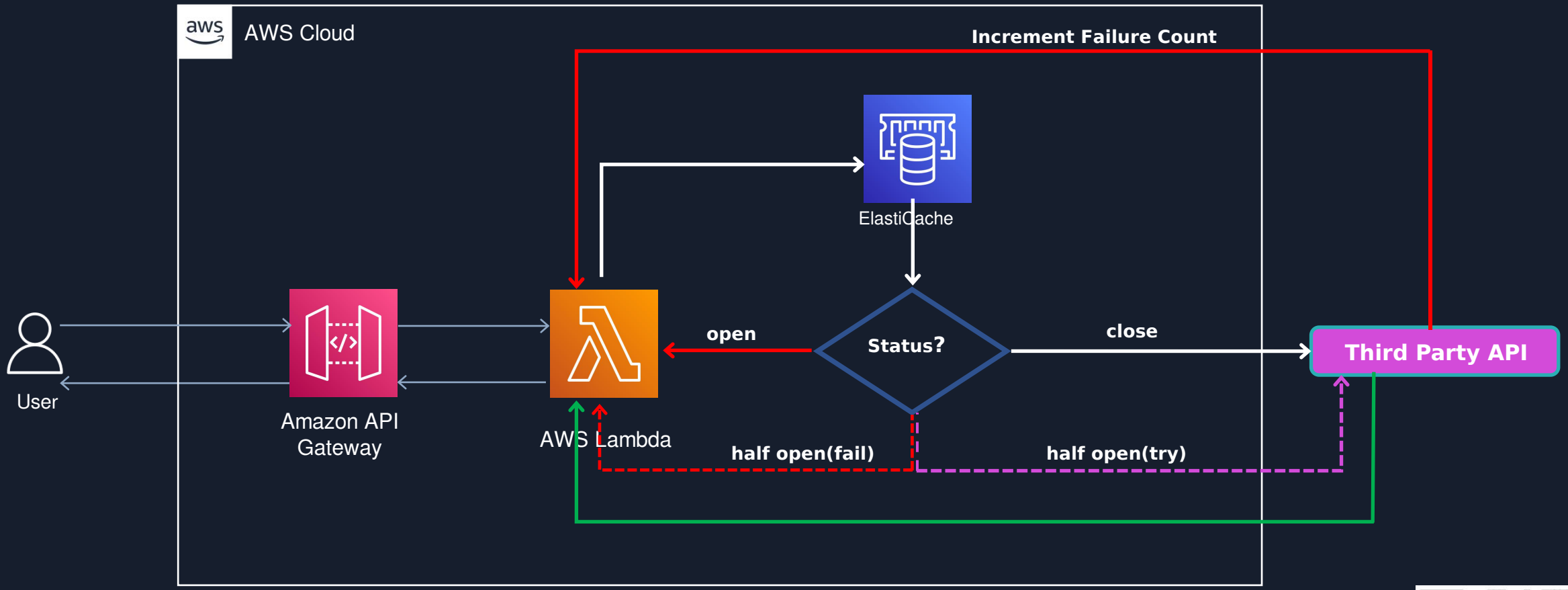

Producer
AWS Lambda
Event Hub
hub
EventBridge
Amazon Kinesis
stream
Consumer
AWS Lambda
Event Lake
Amazon S3
OpenSearch


Software Architecture Patterns for Serverless Systems
Architecting for innovation with events, autonomous services, and micro frontends
John Gilbert

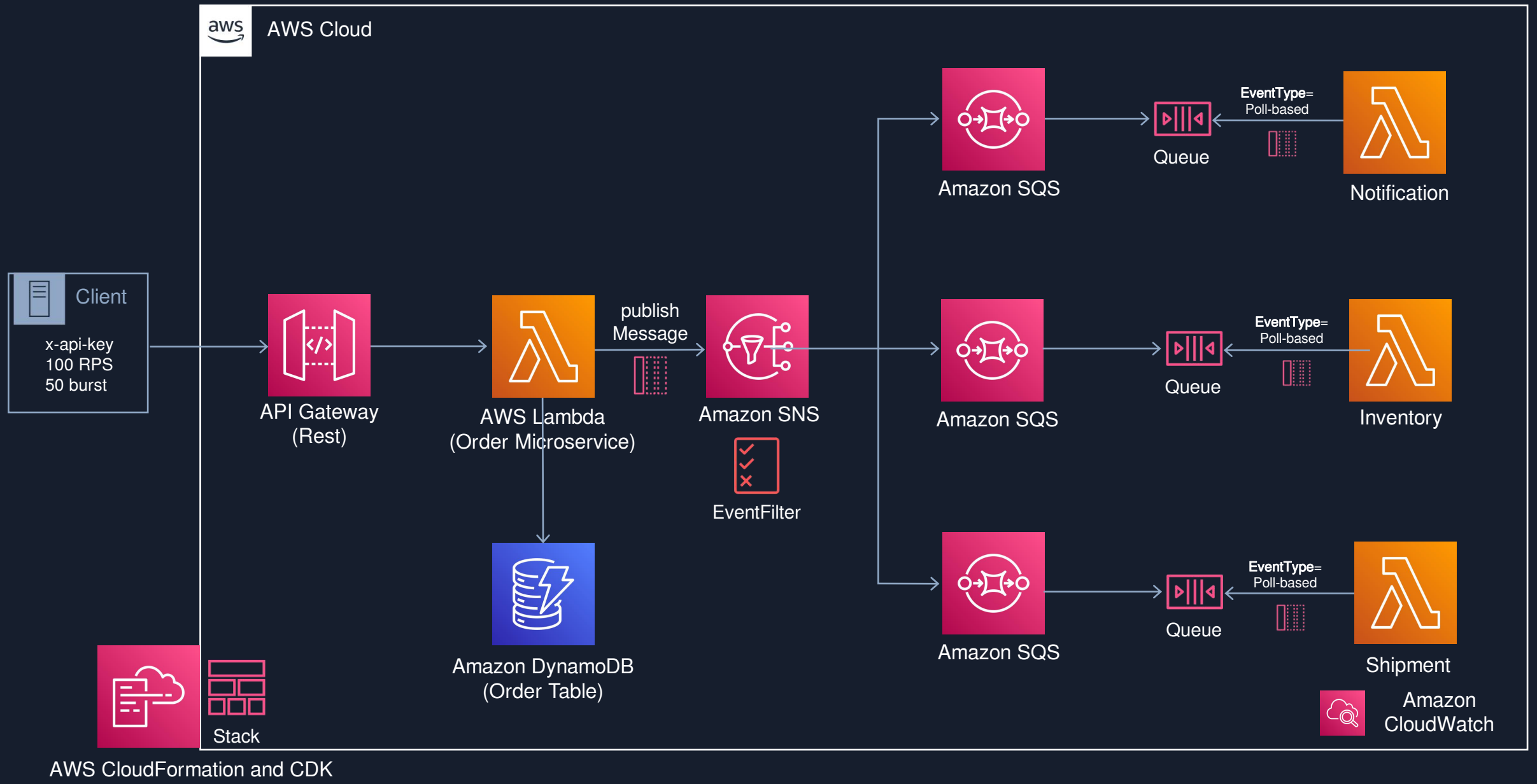
Saga Pattern For Orchestrate Distributed Transaction (AWS StepFunctions)



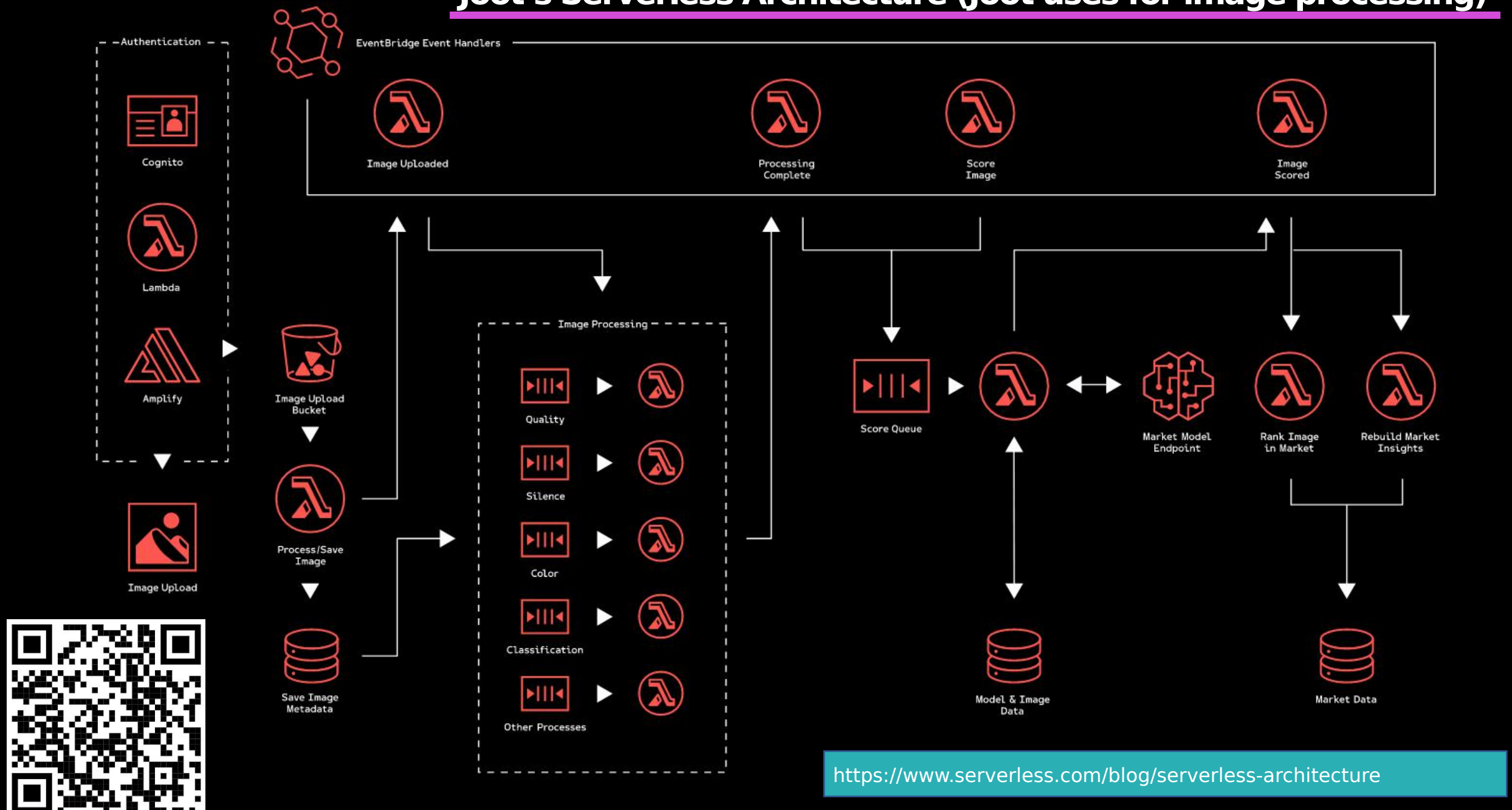
Circuit Breaker and Many More by Jeremy Daly



Fan-Out Serverless Architectures Using SNS, SQS and Lambda



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



https://www.serverless.com/blog/serverless-architecture

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.

https://aws.amazon.com/serverless/getting-started/?nc=sn&loc=2&serverless.sort-by=item.additionalFields.createdDate&serverless.sort-order=desc

How to build, run and deploy?

serverless

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

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

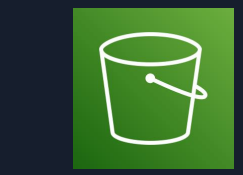
Architect

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

CLAUDIA.JS

Claudia.js makes it easy to deploy Node.js projects to AWS Lambda and API Gateway.

https://aws.amazon.com/serverless/getting-started/?nc=sn&loc=2&serverless.sort by=item.additionalFields.createdDate&serverless.sort-order=desc



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

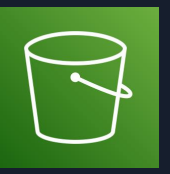
```
aws s3api put-bucket-versioning --bucket  s3-join-community-2023-files-us-ea-2 --versioning-configuration Status=Susp
```

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

https://github.com/weder96/joinCommunity2023/blob/main/learning/09_S3/README.md

AWS Database

DynamoDB Primary Key, Partition Key and Sort Key

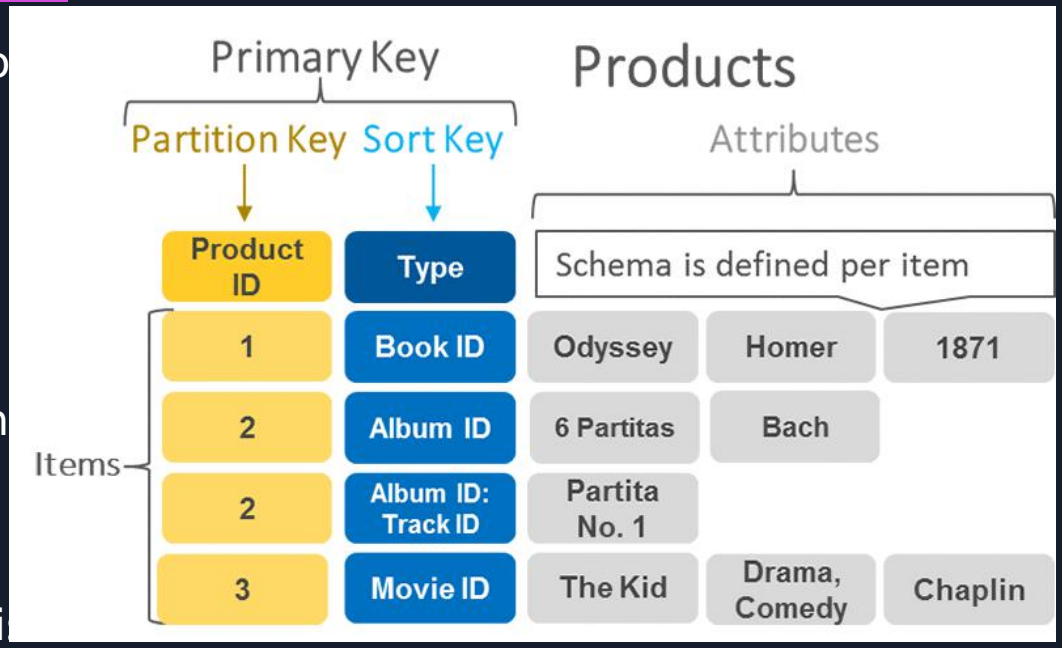
A primary key **uniquely identifies** each item in the table, so two items can have the same key. DynamoDB supports two different kinds of primary keys:

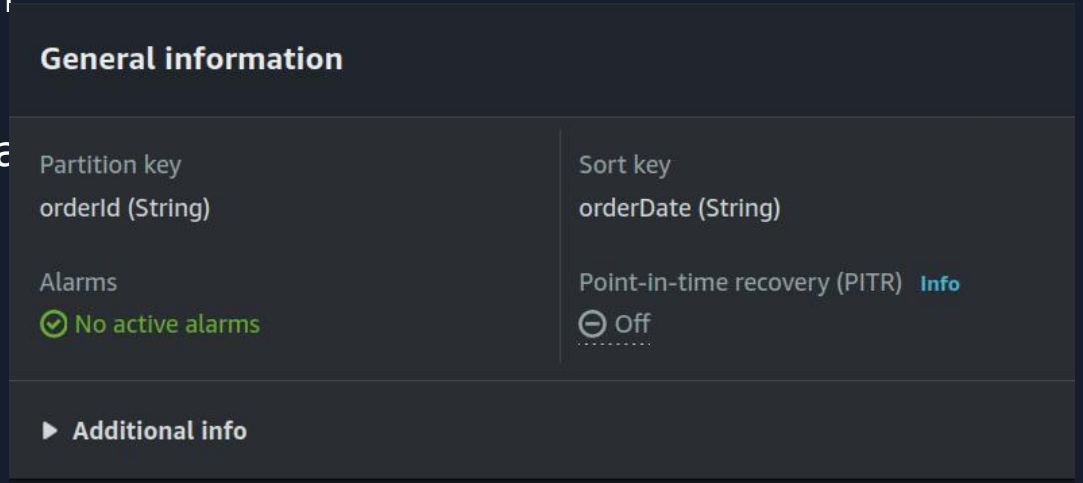
- Partition key
- Partition key and sort key

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

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

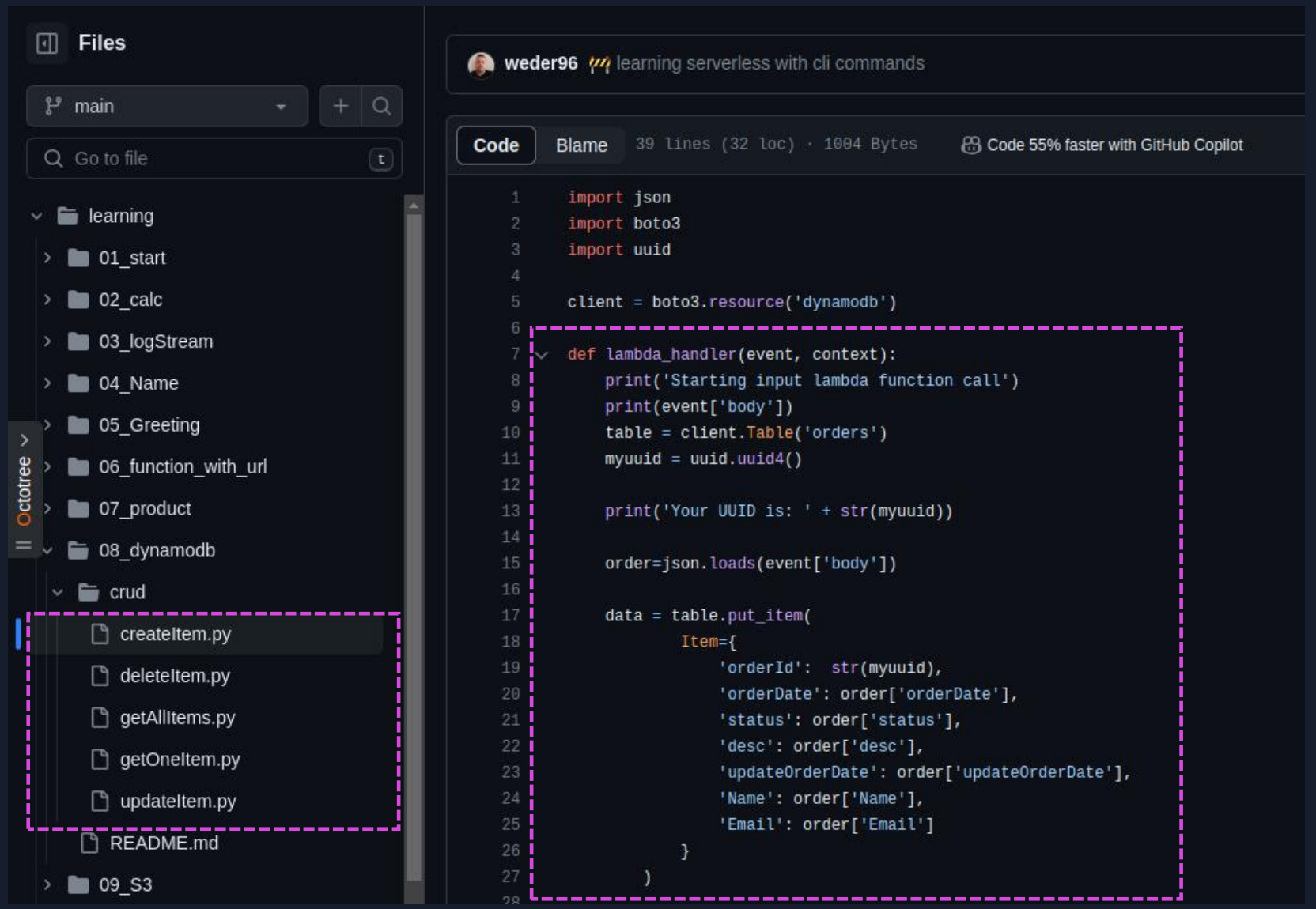
DynamoDB uses the partition key value as input to an interna hash function. A composite primary key gives you additional flexibility when **querying data**





https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

DynamoDB Python(Boto3)



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

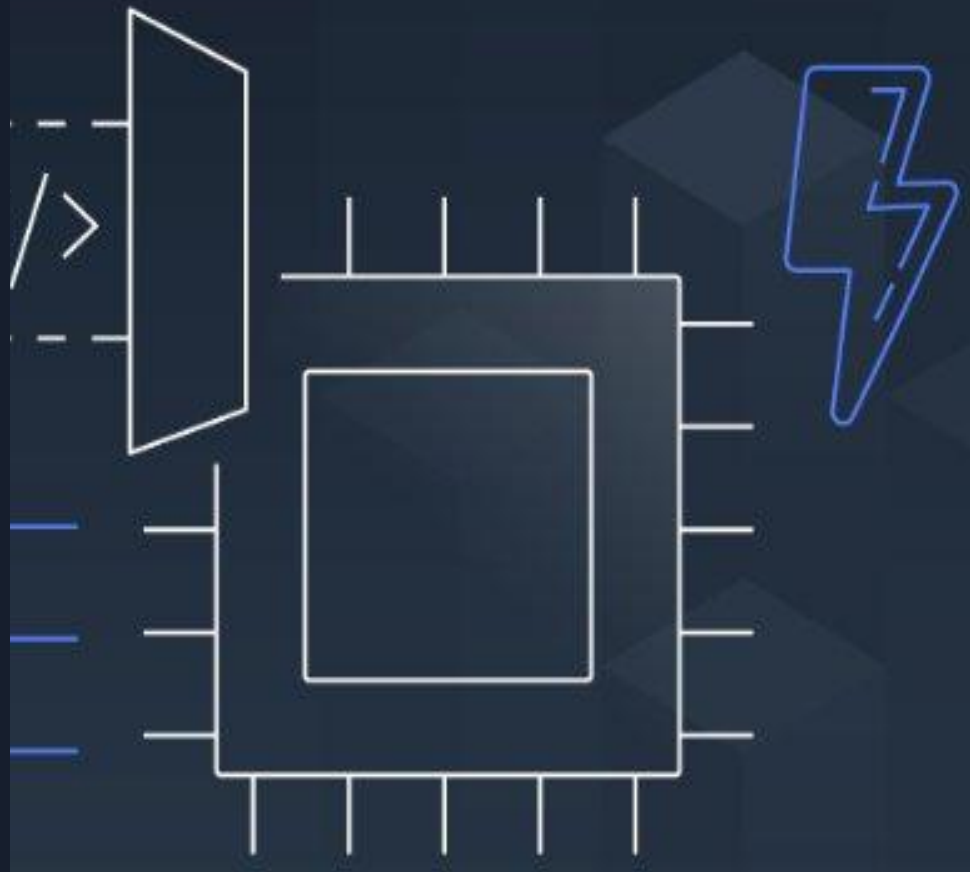
```
aws dynamodb create-table \
    --table-name orders \
    --attribute-definitions \
        AttributeName=id,AttributeType=S \
        AttributeName=status,AttributeType=S \
    --key-schema \
        AttributeName=id,KeyType=HASH \
        AttributeName=status,KeyType=RANGE \
    --provisioned-throughput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

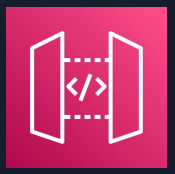
Model – Data schema specifying the data structure of a request or response payload.

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.

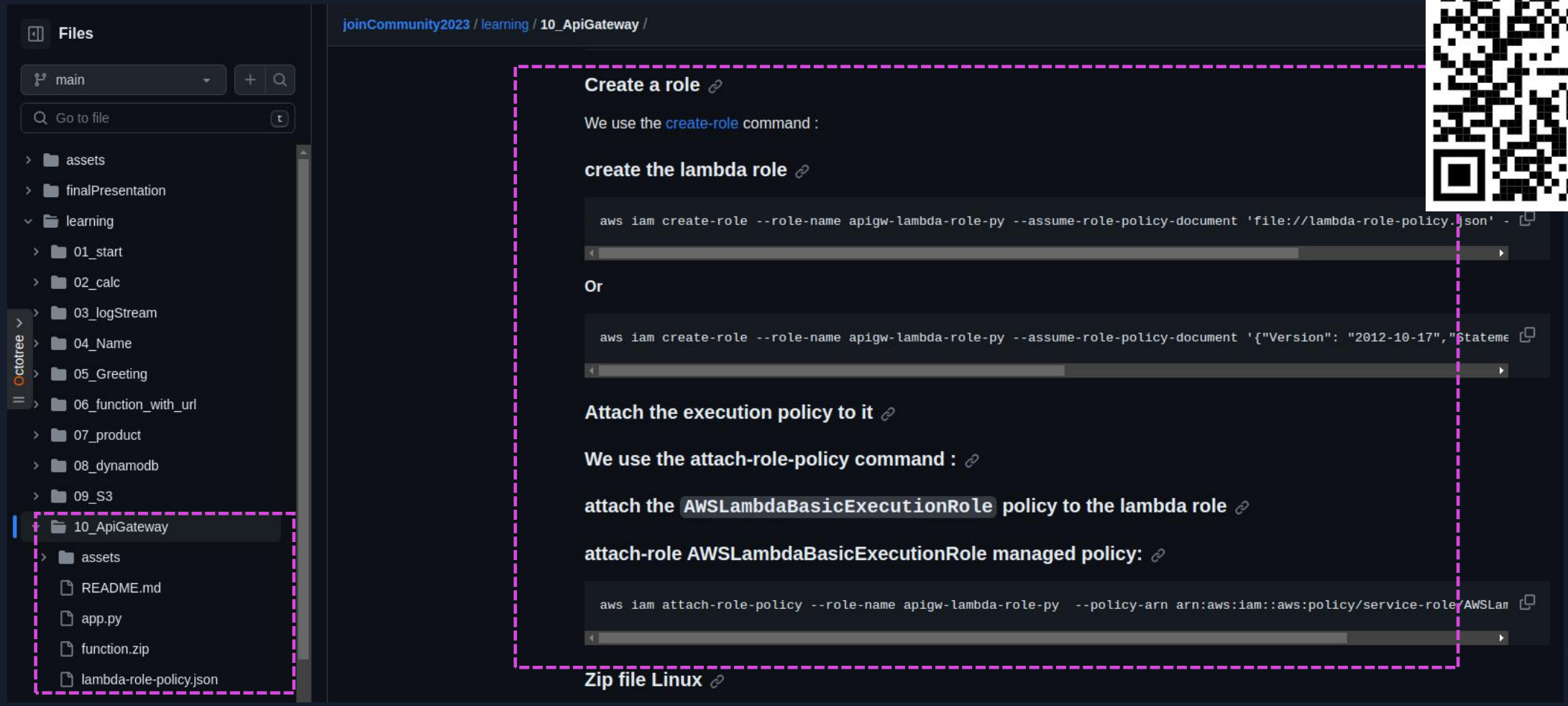
Usage plan – Provides selected API clients with access to one or more deployed APIs. You can use a usage plan to configure throttling and quota limits, which are enforced on individual client API keys.



Amazon
API Gateway

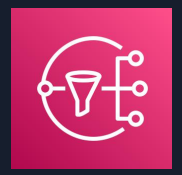
https://github.com/weder96/aws-certification-learning/tree/main/module-5#section-10

Amazon API Gateway CLI Commands

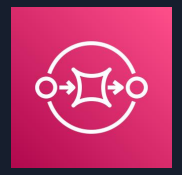


Amazon API Gateway

https://github.com/weder96/joinCommunity2023/tree/main/learning/10_ApiGateway



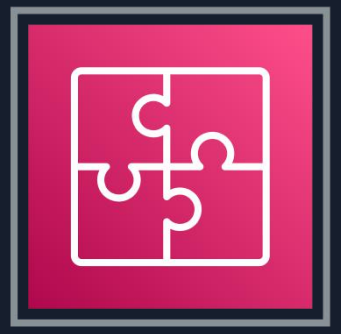
SNS



SQS

EVENTBRIDGE

AWS Application Integration




Amazon
SNS
aws

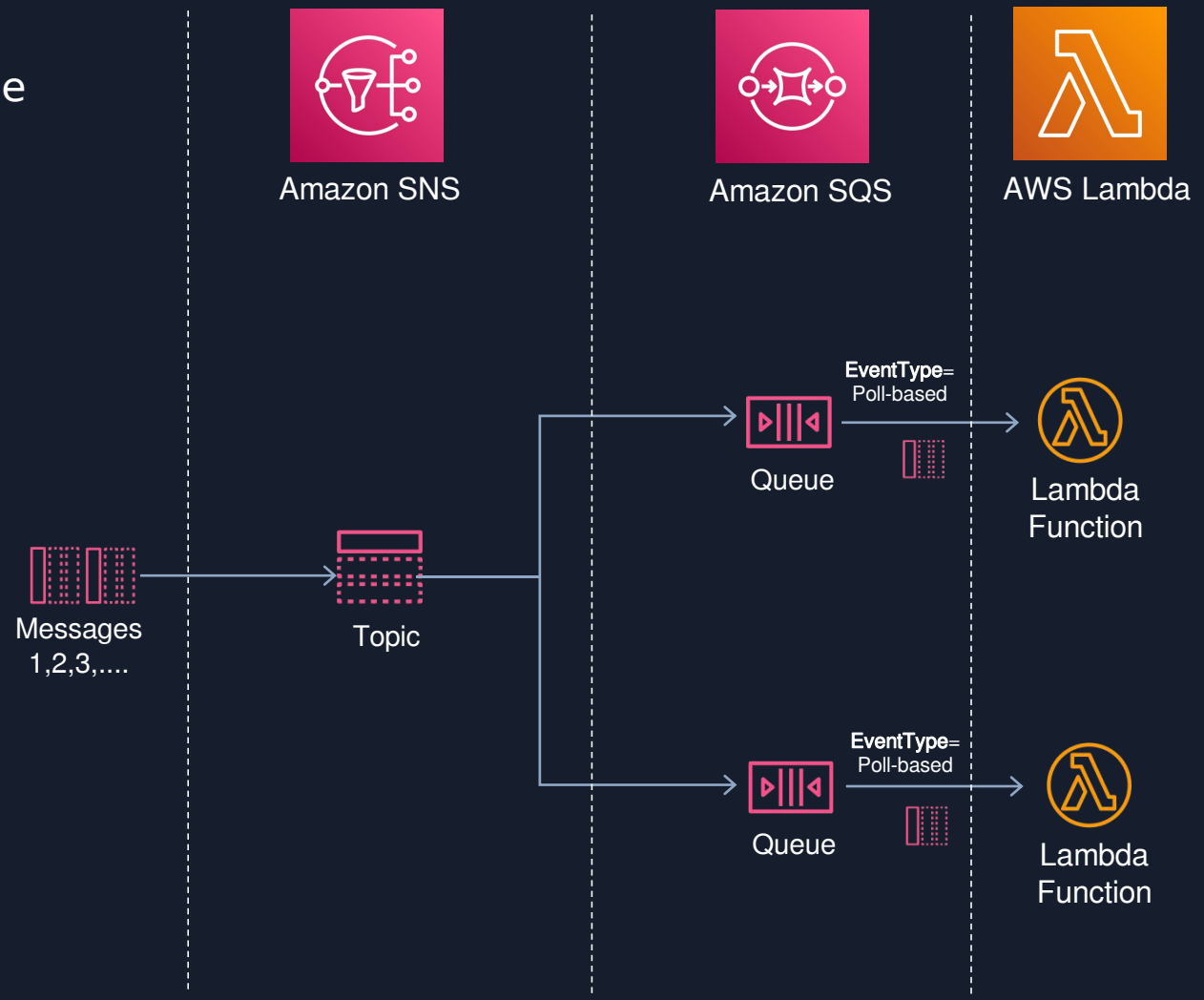
Amazon SNS: Fully Managed Pub/Sub Messaging

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 Code 55% faster with GitHub Copilot

Lambda and SNS with Python

Zip file Linux

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

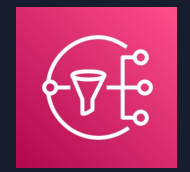
Create Lambda Function

```
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

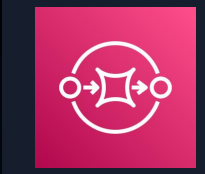
To create a topic, use the sns create-topic command and specify the name to assign to the topic.

```
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.

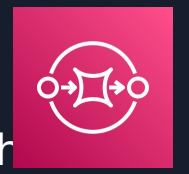
Enables you to **decouple** and **scale microservices**, distributed systems, and serverless applications.

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.



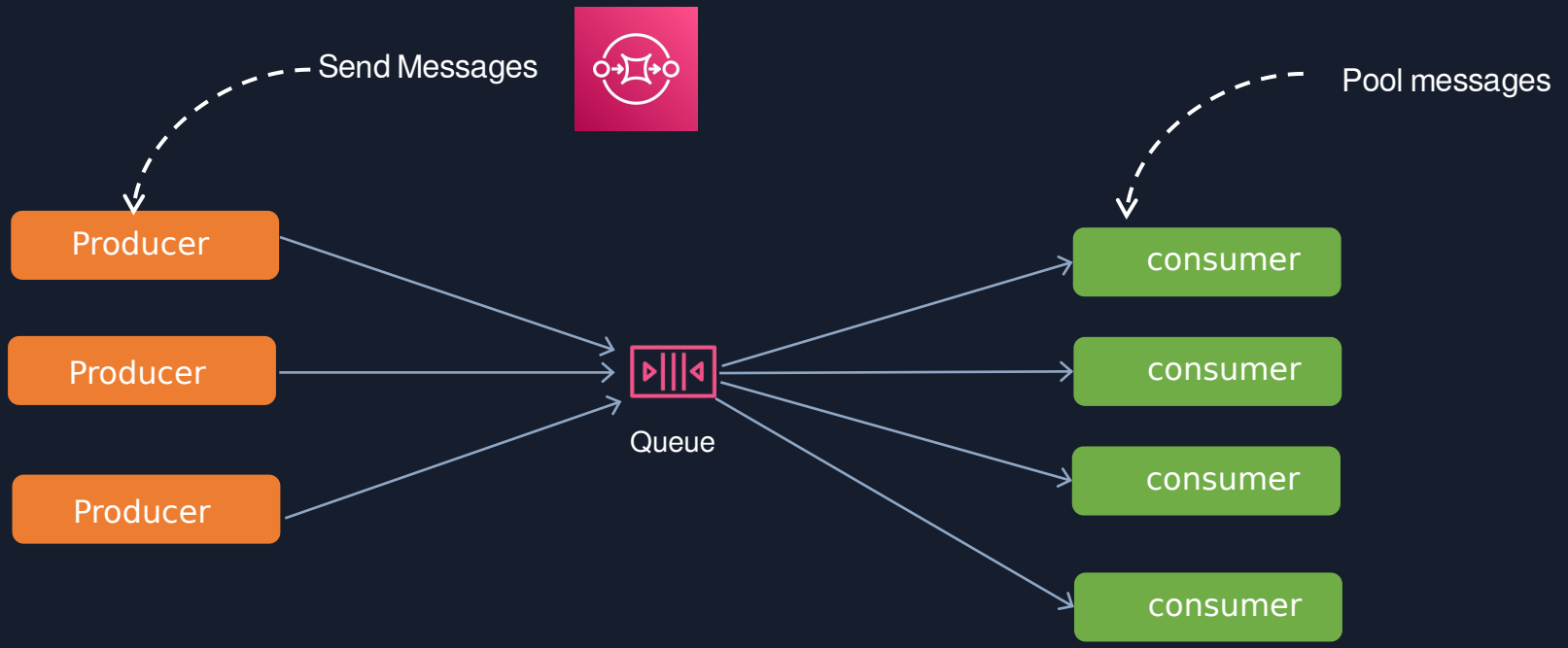
Amazon SQS

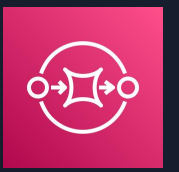
Provides a **generic web services API** that you can access using any programming language that th
AWS
SDK supports.

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-att
```

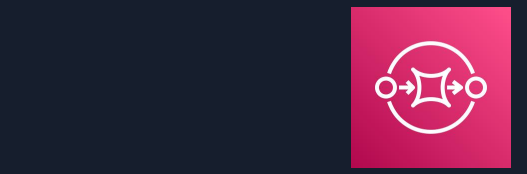
Deletion of a Processed Message from the AWS SQS Queue

```
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



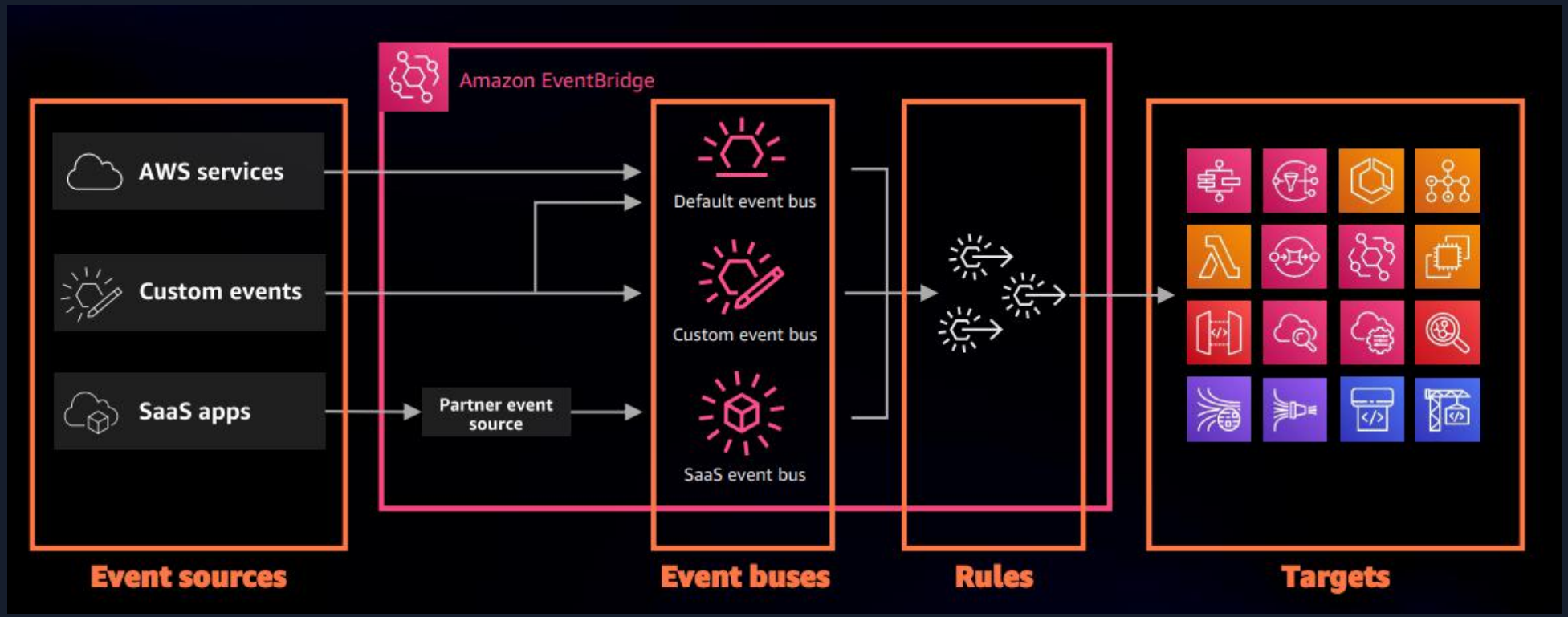
Amazon SQS

https://github.com/weder96/joinCommunity2023/blob/main/learning/16_lambda_sqs/README.md


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**



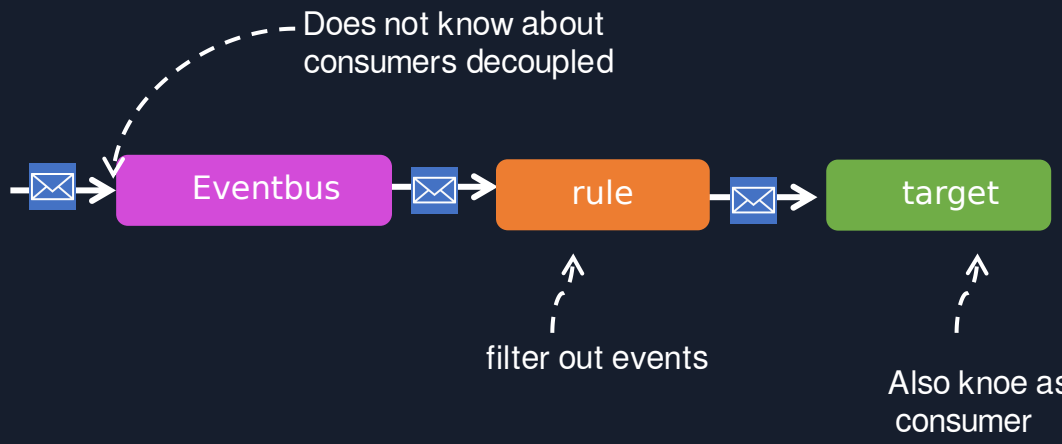
https://serverlessland.com/patterns
https://da-public-assets.s3.amazonaws.com/serverlessland/pdf/2021+-+Serverlesspresso+exhibit+-+PDF.pdf

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

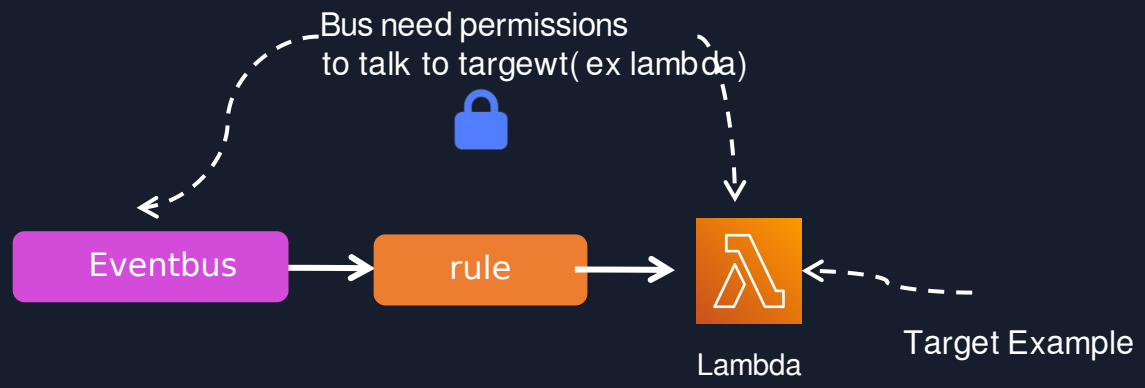
https://da-public-assets.s3.amazonaws.com/serverlessland/pdf/2021+-+Serverlesspresso+exhibit+-+PDF.pdf

Understanding EventBridge target



What are target ?

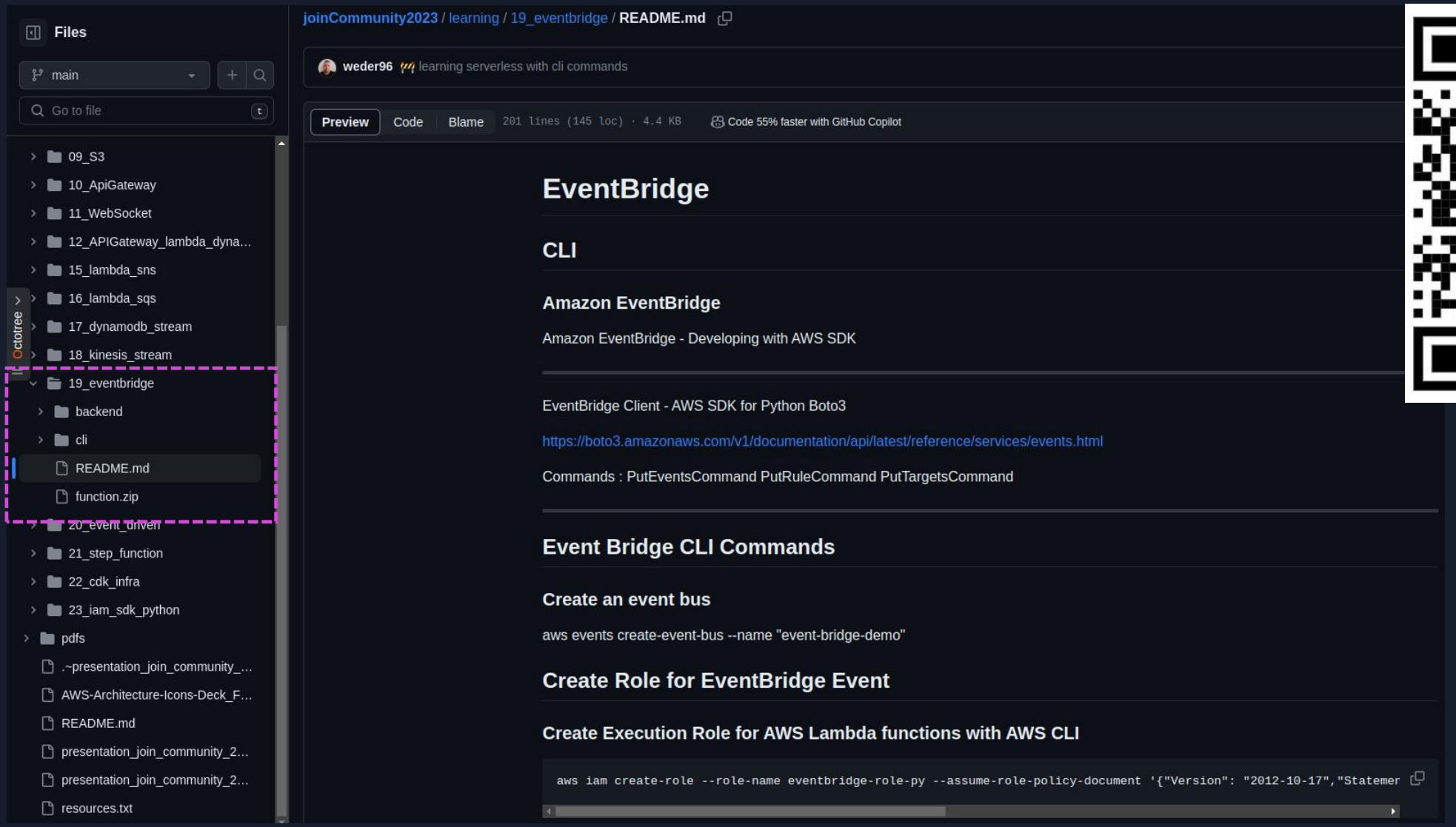
Targets are consumer of you events
use rules to process events before reaching targets



Target Permissions

Your eventBridge bus need permissions
to send events to targets

Amazon EventBridge CLI Commands and (Python) Boto3



Amazon EventBridge

https://github.com/weder96/joinCommunity2023/blob/main/learning/19_eventbridge/README.md

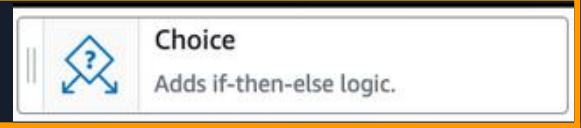
STEP FUNCTIONS

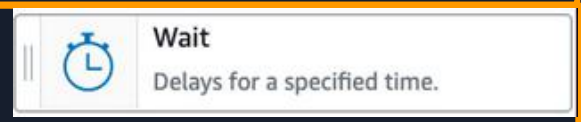
AWS Application Orchestration

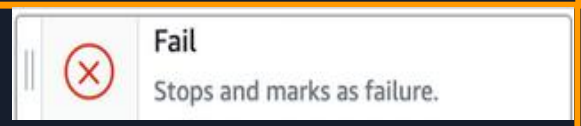

AWS
Step Functions

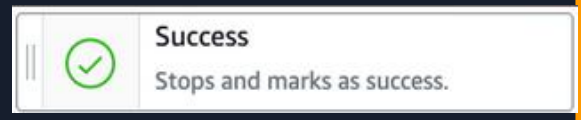
State types

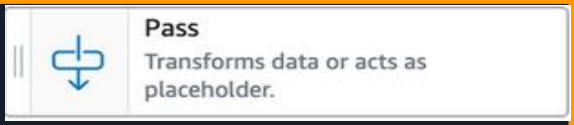
Task	A single unit of work
Choice	Adds branching logic
Parallel	Fork and join the data across tasks
Wait	Delay for a specified time
Fail	Stops an execution and marks it as a failure
Succeed	Stops an execution successfully
Pass	Passes its input to its output
Map	Loop and Foreach

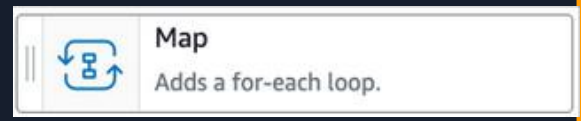


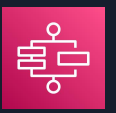




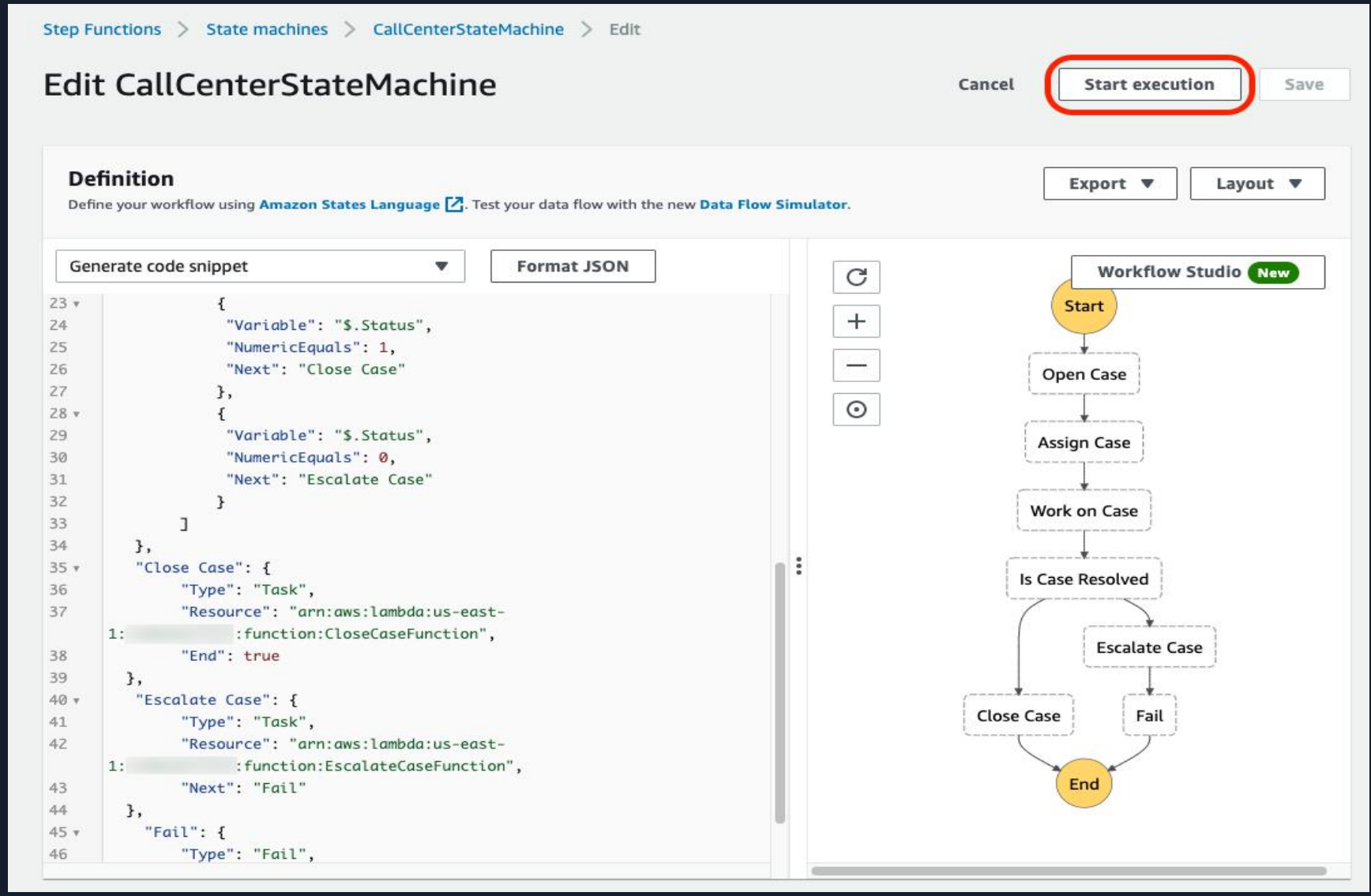






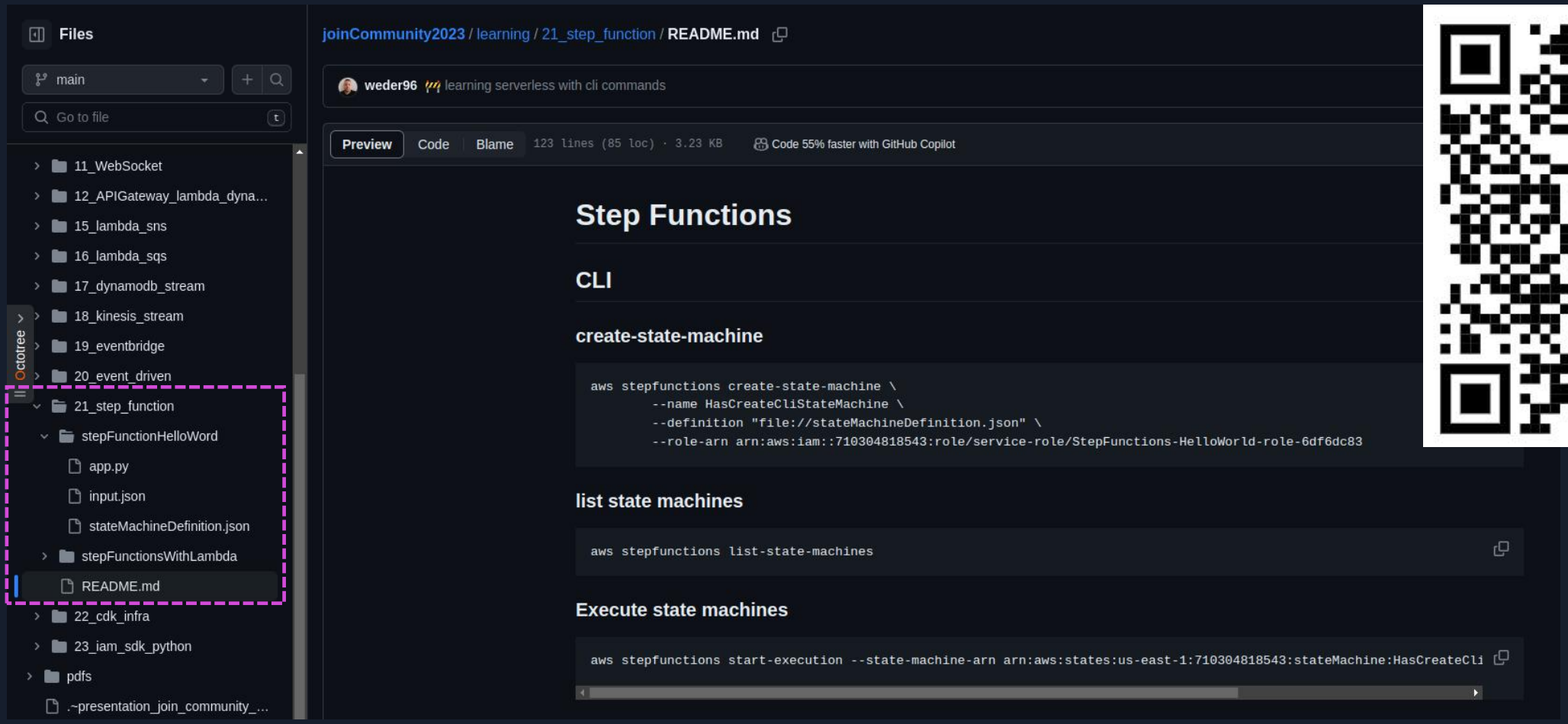


Create a Serverless Workflow with AWS Step Functions and AWS Lambda



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

Amazon Step Functions CLI Commands and (Python) Boto3



AWS Step Functions

https://github.com/weder96/joinCommunity2023/blob/main/learning/21_step_function/README.md

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

https://aws.amazon.com/blogs/architecture/understanding-the-different-ways-to-invoke-lambda-functions/

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

lambda > 01_start > app.py > ...

```python
import json

def lambda_handler(event, context):
    print('Starting input lambda function call')
    print(event)

    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!!!')
    }

```

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"
    }
  ]
}
```

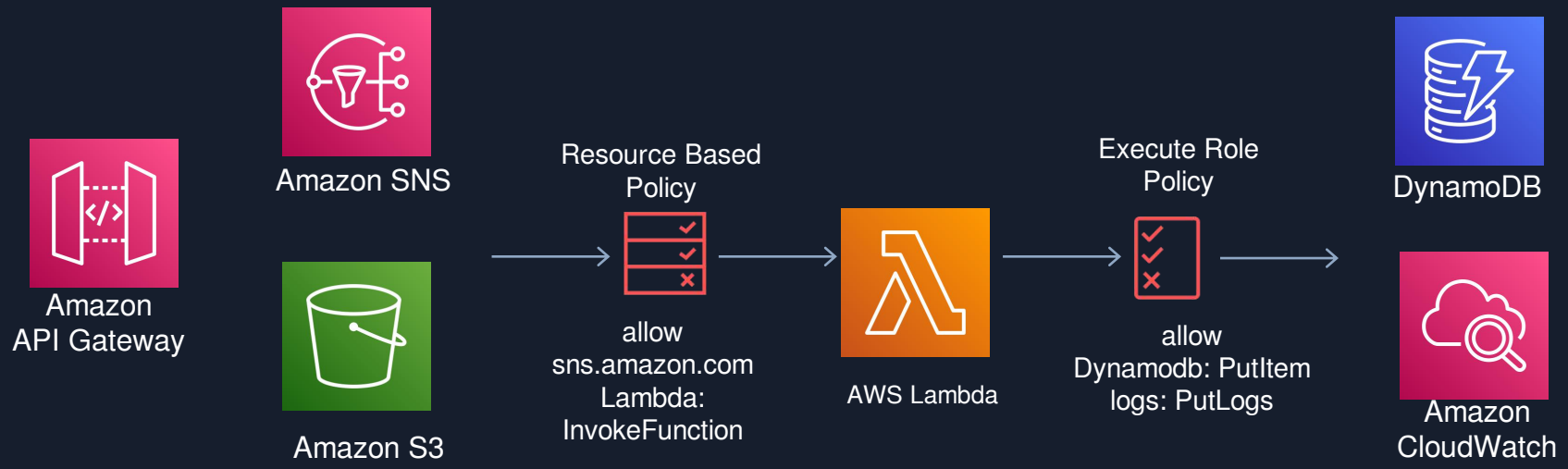
https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.html

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.

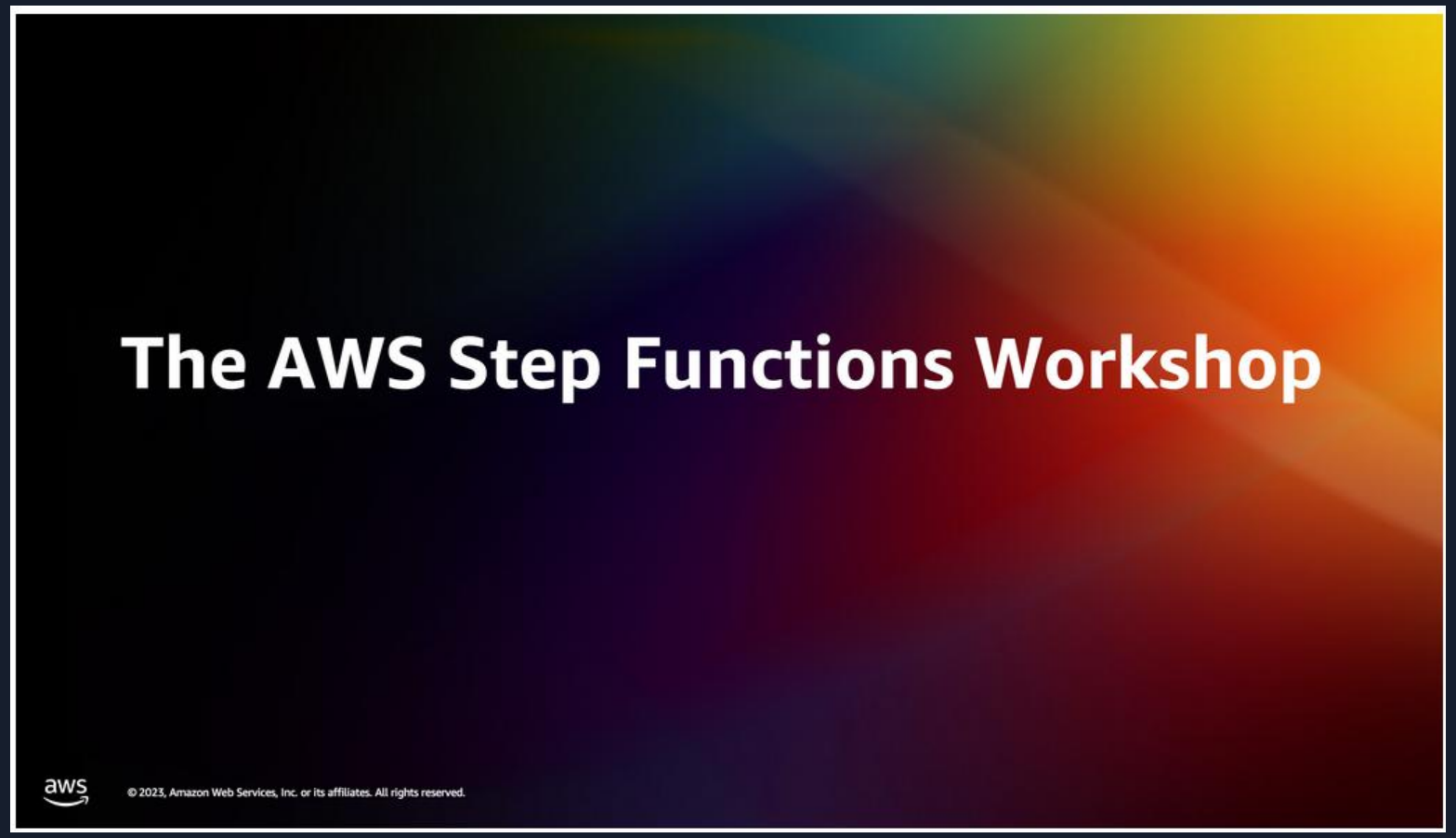
Amazon S3 upload event triggers to lambda function asyncrhously, so we should also add Resource-based policy into our Lambda function grants S3 invocation.



https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.html

Next Steps

The AWS Step Functions Workshop



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

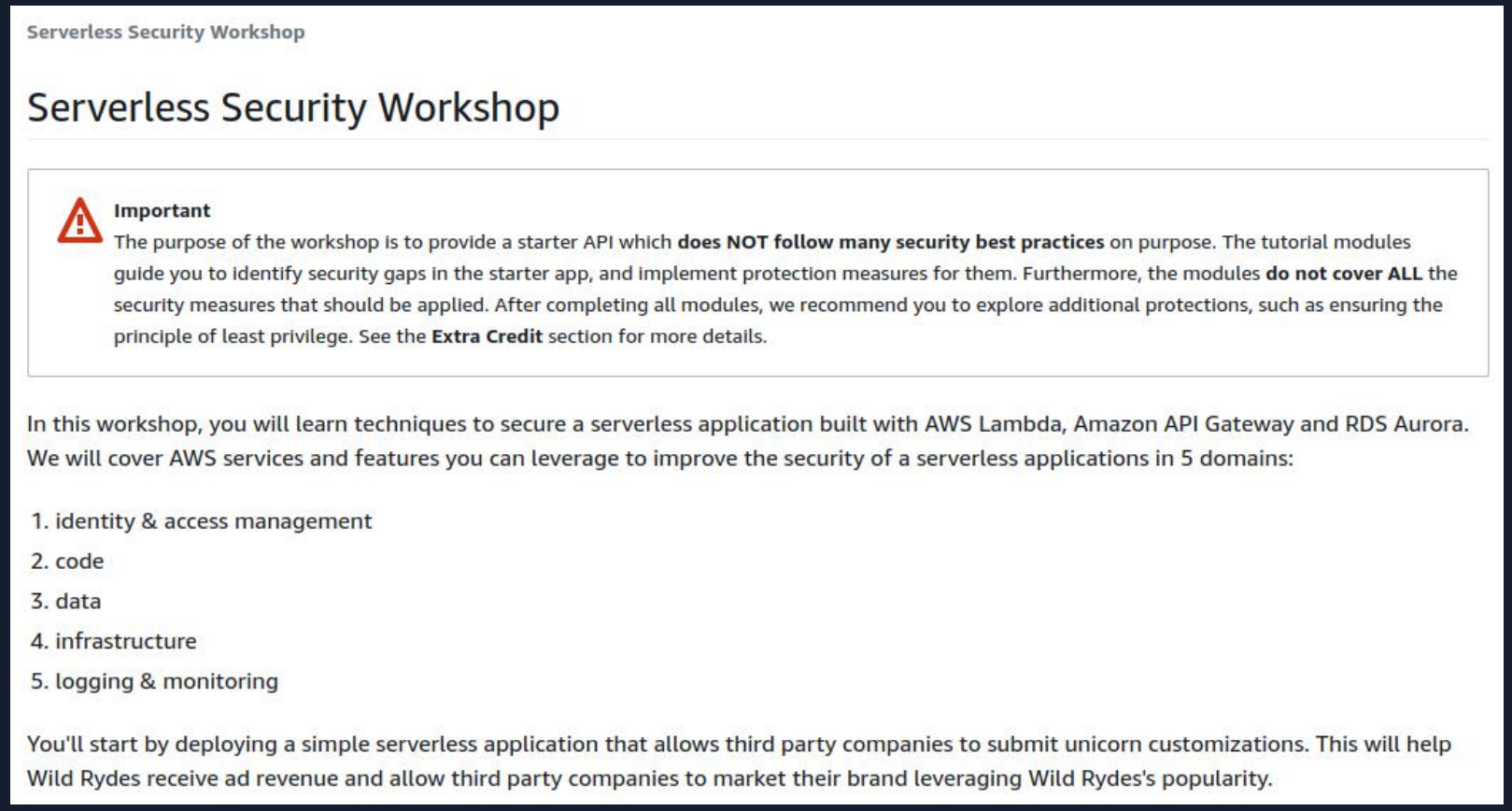
AWS Serverless Digital Learning Badges

AWS Skill Builder

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

Want to Try?

Serverless Security Workshop



https://catalog.us-east-1.prod.workshops.aws/workshops/026f84fd-f589-4a59-a4d1-81dc543fcd30/en-US

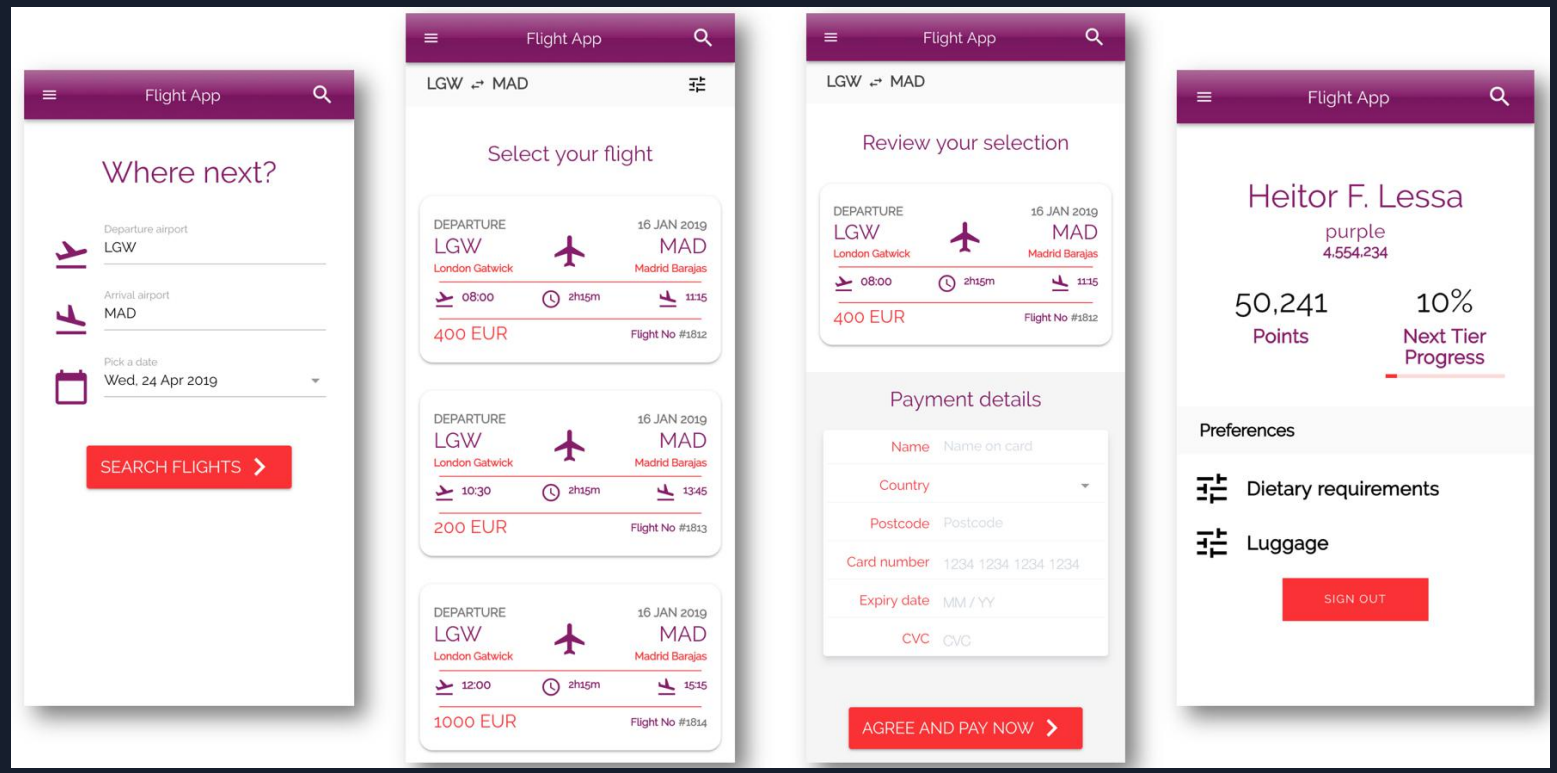
Want to Try?

Serverland AWS Lambda Fundamentals

https://s12d.com/lambda-fundamentals

Want to Try?

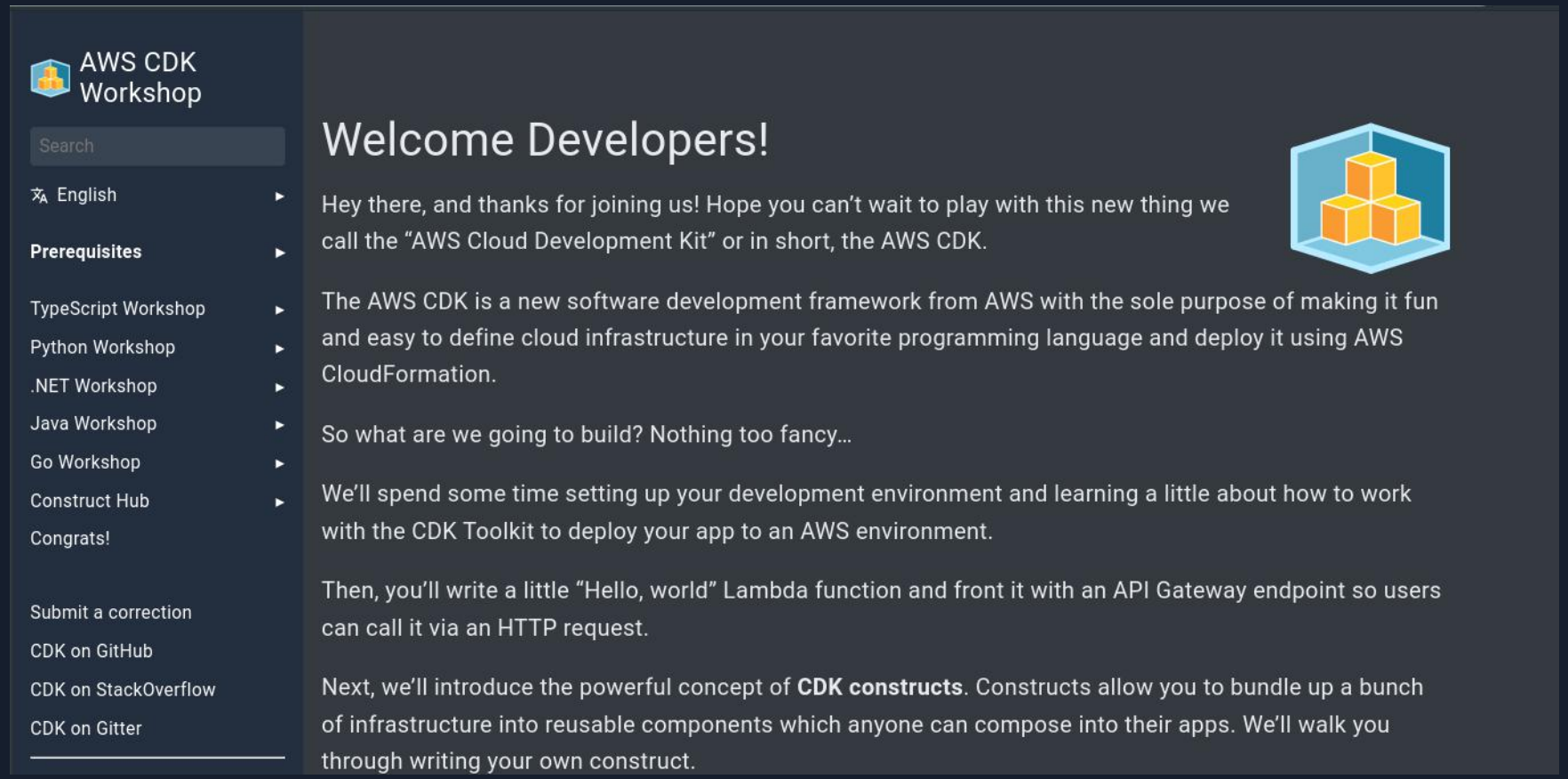
Serverless airline - Multiple patterns/practices



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

Want to Try?

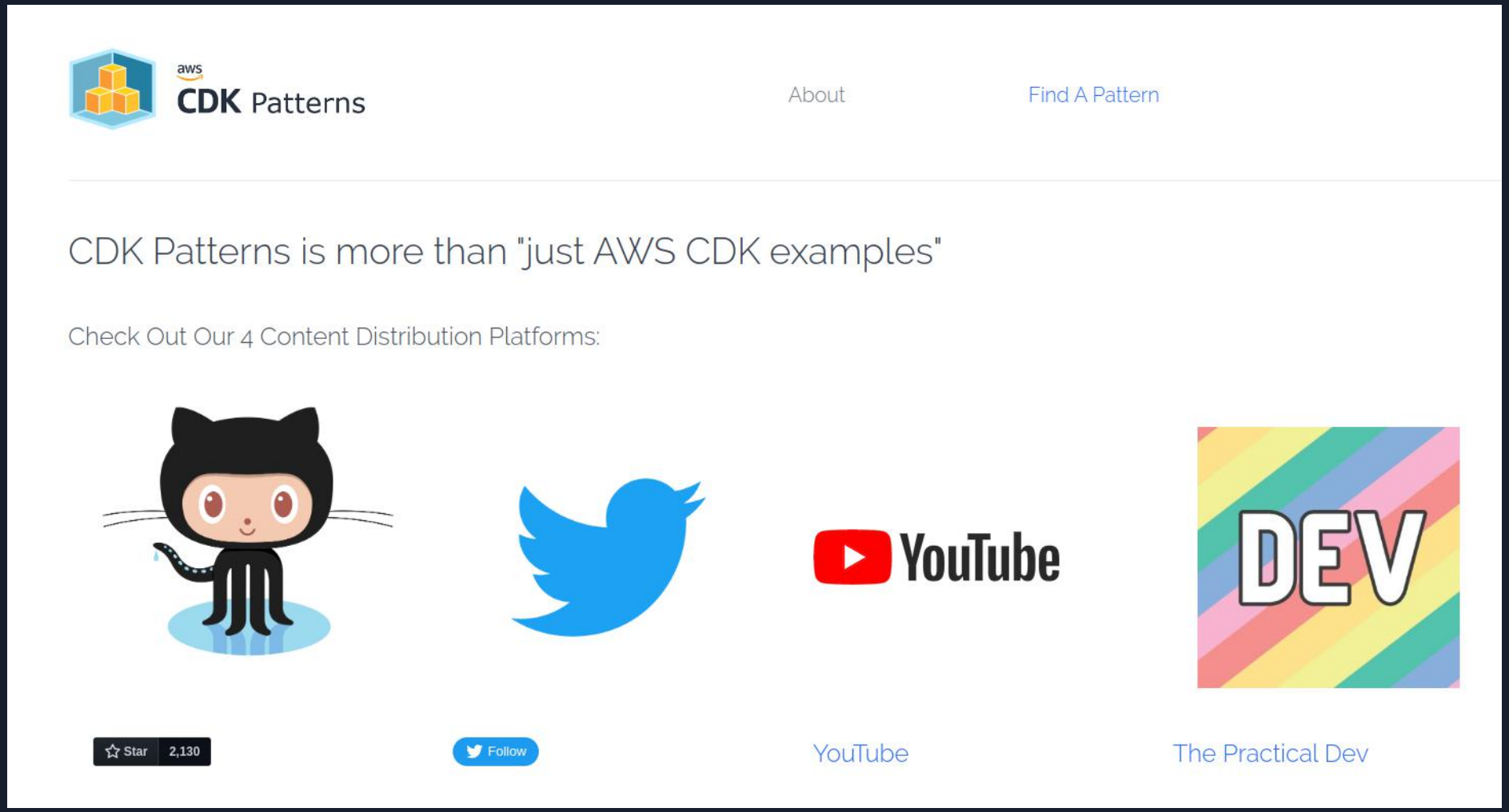
CDK - Workshop



https://cdkworkshop.com/

Want to Try?

CDK Patterns



https://cdkpatterns.com/

Want to Try?

handson.cloud

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

Q & A

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

https://www.linkedin.com/in/wedermarianodesousa/

https://github.com/weder96

https://twitter.com/weder96

https://dev.to/weder96

aws certified Cloud Practitioner FOUNDATIONAL

aws certified Solutions Architect ASSOCIATE

aws certified Developer ASSOCIATE

AWS community builder Serverless

aws EDUCATE Getting Started with Storage

aws Cloud Economics Accredited PARTNER

aws Technical Accredited PARTNER

aws Business Accredited PARTNER

aws TECHNOLOGY File Storage SPECIALIZED

aws TECHNOLOGY Storage Data Migration SPECIALIZED

aws TECHNOLOGY Storage Data Protection & Disaster Recovery SPECIALIZED

aws Well-Architected PROFICIENT

aws TECHNOLOGY Amazon Braket

aws TECHNOLOGY Architecting

aws TECHNOLOGY Migration Foundations

aws TECHNOLOGY Amazon EKS

aws TECHNOLOGY Networking Core

aws TECHNOLOGY Serverless

aws TECHNOLOGY Events and Workflows

aws TECHNOLOGY Compute

aws TECHNOLOGY Cloud Essentials

AWS
User Groups
Anápolis, Brazil

THANK YOU