Developer Associate Certification - Thoughts

Francisco Javier Moreno Diaz – GenAI
Assisted on Slides 2-4 and 15-18
fmorenod@gmail.com

Comparison with SAA-C03

- DVA-C02 goes deeper into:
- *Serverless development*
- *Service integration*
- *Deployment automation*
- *Observability & optimization*
- Assumes you already know: IAM, VPC, S3, RDS, high-availability
- Shift the angle: **less architecture**, **more code + integration patterns**

Dimension	SAA-C03	DVA-C02
Focus	Architecture, design, resilience	Development, integration, deployment
Typical questions	High availability, networking, storage, security	Lambda patterns, API Gateway, DynamoDB, CI/CD
Technical level	Broad, generalist	Deep in serverless & runtime
Skills required	Infrastructure + cloud design	Code, events, SDKs, IaC
Key services	EC2, ALB/NLB, RDS, VPC	Lambda, API Gateway, DynamoDB, SQS/SNS, EventBridge
Mindset	How to design it	How to build, integrate and operate it

Conclusion:
→ SAA = *how to design the system*
→ DVA = *how to build and integrate the system*

Services you must master

- 3 page of services, really ?
  - AWS Lambda
  - API Gateway (REST + HTTP + WebSocket)
  - DynamoDB (partition key, sort key, GSIs, LSI, streams)
  - SQS / SNS
  - EventBridge
  - CloudWatch / X-Ray
  - IaC tools: SAM / CloudFormation

DAV-C02 Study Guide

The exam also validates a candidate's ability to complete the following tasks:

- Develop and optimize applications on AWS.
- Package and deploy by using continuous integration and continuous delivery (CI/CD) workflows.
- Secure application code and data.
- Identify and resolve application issues.

The exam has the following content domains and weightings:

- Content Domain 1: Development with AWS Services (32% of scored content)
- Content Domain 2: Security (26% of scored content)
- Content Domain 3: Deployment (24% of scored content)
- Content Domain 4: Troubleshooting and Optimization (18% of scored content)

Developing on AWS - Course

Course objectives

In this course, you will learn to:

- Set up the AWS SDK and developer credentials for Java, C#/.NET, and Python
- Interact with AWS services and develop solutions by using the AWS SDK
- Use AWS Identity and Access Management (IAM) for service authentication
- Use Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB as data stores
- Integrate applications and data by using AWS Lambda, Amazon API Gateway, Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), and AWS Step Functions
- Use Amazon Cognito for user authentication
- Use Amazon ElastiCache to improve application scalability
- Leverage the CI/CD pipeline to deploy applications on AWS

Developing on AWS - Outline

Day 1

Module 0: Course Overview

Module 1: Introduction to AWS: Infrastructure overview, Introduction to AWS foundation services

Module 2: Introduction to Developing on AWS: Introduction to developer tools, management tools

Module 3: Introduction to AWS Identity and Access Management: IAM, AuthN&AuthZ

Module 4: Introduction to the Lab Environment

Module 5: Developing Storage Solutions with S3: Troubleshooting

Day 2

Module 6: Developing Flexible NoSQL Solutions with Amazon DynamoDB: Best practices, Troubleshooting

Module 7: Developing Event-Driven Solutions with AWS Lambda: How Lambda works, Use cases, Best practices

Module 8: Developing Solutions with Amazon API Gateway: Developing with API Gateway, Best practices, Introduction to SAM

Module 9: Developing Solutions with AWS Step Functions: Use cases

Day 3

Module 10: Developing Solutions with SQS and SNS: Developing SQS, SNS; Developing with Amazon MQ

Module 11: Caching Information with Amazon ElastiCache: Caching strategies

Module 12: Developing Secure Applications

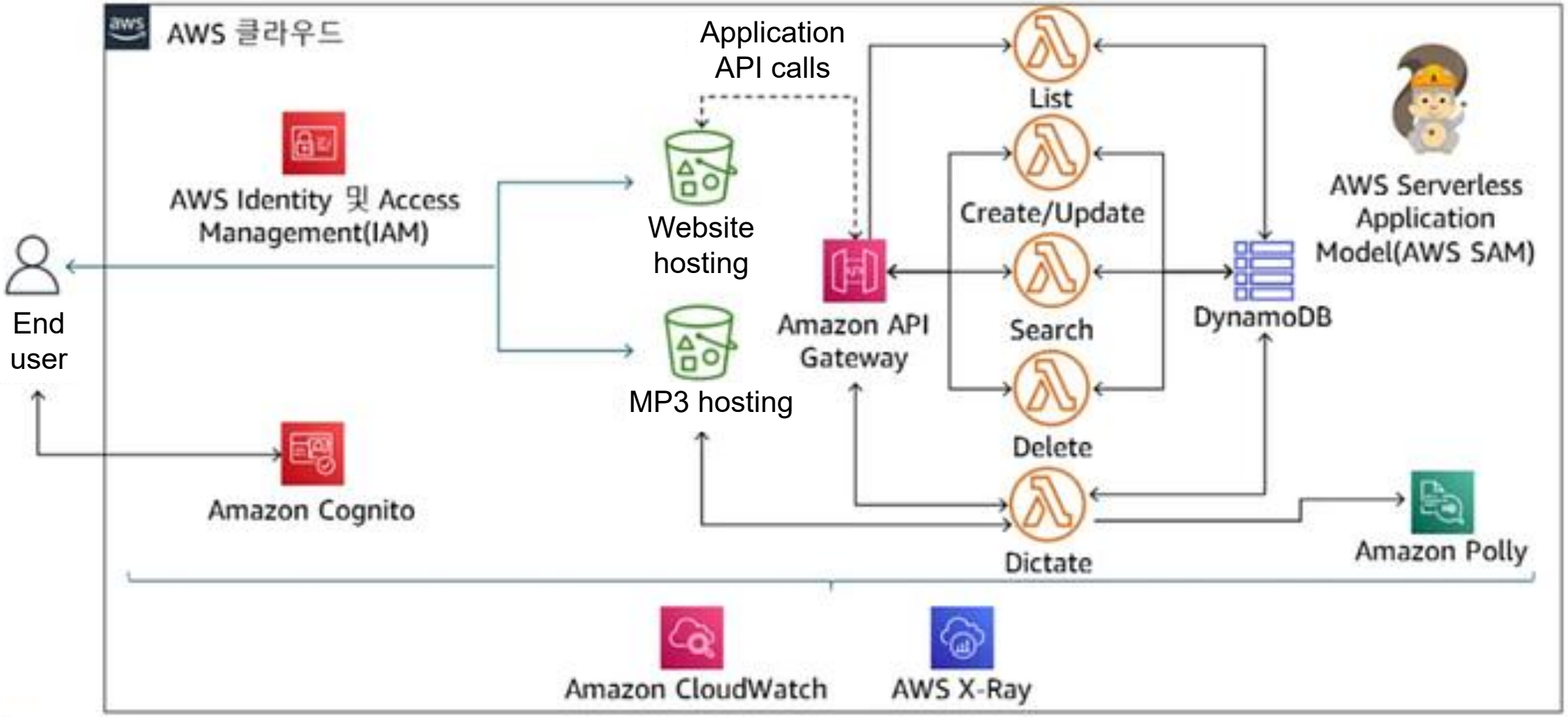
Module 13: Deploying Applications: Devops, Deployment and testing strategies, AWS Elastic Beanstalk

Module 14: Course wrap-up

Frequent Patterns

- API Gateway → Lambda → DynamoDB
- Event-driven architectures: SQS, SNS, EventBridge
- Retries, Dead Letter Queues (DLQs)
- Deployment strategies: Lambda versions & aliases, blue/green
- Observability: Logs, metrics, tracing
- Serverless best practices (for DVA-C02)

Proposed Lab

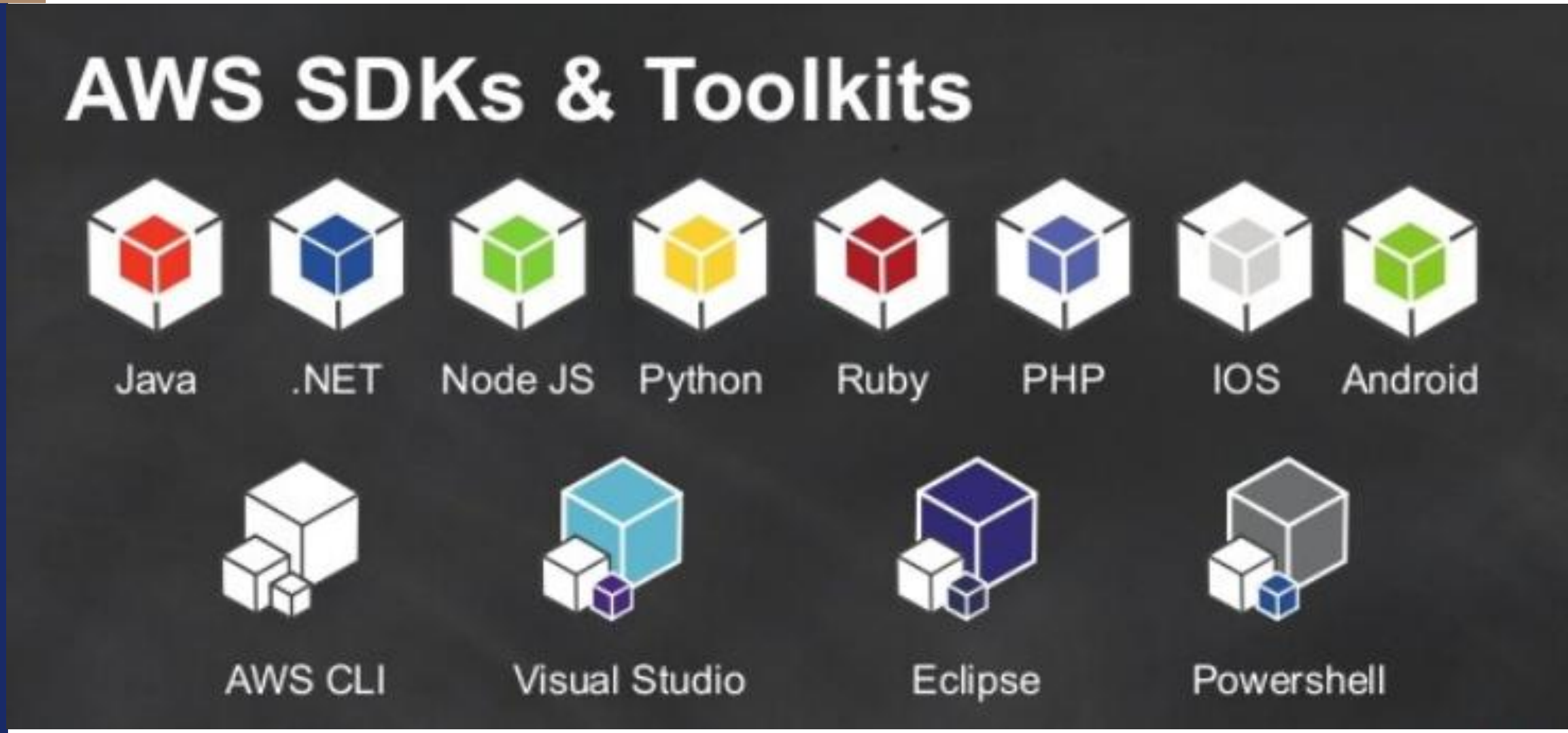


Types of questions & how to answer them

- Scenario based: “A developer has to…”
- Focus on:
- Managed services vs self-managed
- Serverless vs servers
- Scalability & high availability
- Approach:
- Read the question fully before scanning options
- Eliminate answers that involve managing servers when a serverless service fits
- Think in terms of best practice within AWS ecosystem
- Choosing EC2 when serverless is appropriate
- Confusing IAM roles vs policies
- Overlooking idempotency in functions
- Ignoring DLQ / retry logic
- Poor design in DynamoDB: hot partitions, inefficient queries

Study Plan

Week 1: Dive into Lambda, API Gateway, DynamoDB — hands-on labs
Week 2: Infrastructure as Code, CI/CD, SAM
Week 3: Practice with mock exams, identify weaknesses
Week 4: Final review, simulated exam conditions


AWS SDKs & Toolkits
Java
.NET
Node JS
Python
Ruby
PHP
IOS
Android
AWS CLI
Visual Studio
Eclipse
Powershell

Which API SDK have to use

SDK types

- Low-level API
- High-level API

Example of a low-level API: **Get a list of S3 buckets.**

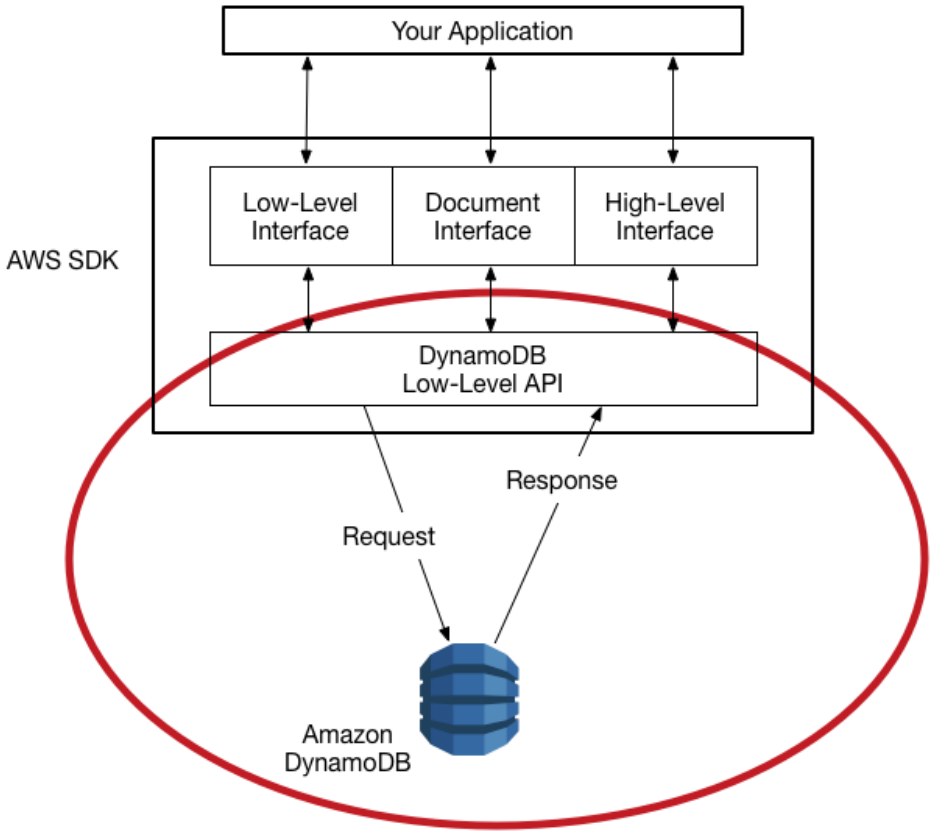
```
# Retrieve the list of existing buckets
s3 = boto3.client('s3')
response = s3.list_buckets_v2(Bucket='mybucket')

for content in response['Contents']:
    print(f'  {content["Key"]} : {content["LastModified"]}')
```

Example of a high-level API: Retrieving a list of S3 buckets (same functionality as above). It can be used more concisely through abstraction, and can be accessed and used like a Method/Member rather than accessing it directly as a structure (JSON).

```
# Retrieve the list of existing buckets
s3 = boto3.resource('s3')
bucket = s3.Bucket('mybucket')

for obj in bucket.objects.all():
    print(f'  {obj.key} : {obj.last_modified')
```



Resources

- AWS Skill Builder — free courses
- AWS Free Tier Labs
- Serverless Land patterns repository
- Tutorial Dojo free question sets
- Official AWS documentation & whitepapers

Question 1

Una función Lambda procesa eventos desde una cola SQS Standard. Algunos mensajes fallan repetidamente por datos corruptos. Se requiere evitar bloqueos y garantizar que los mensajes fallidos no se reprocesen indefinidamente.
¿Qué solución es la *más adecuada*?

A. Configurar un *visibility timeout* más largo
B. Configurar una DLQ en SQS y un máximo de reintentos
C. Aumentar el timeout de Lambda para que procese mensajes más grandes
D. Crear una nueva función Lambda que valide mensajes antes del procesamiento

Question 1 - Answer

Una función Lambda procesa eventos desde una cola **SQS Standard**. Algunos mensajes fallan repetidamente por datos corruptos. Se requiere evitar bloqueos y garantizar que los mensajes fallidos no se reprocesen indefinidamente.
¿Qué solución es la *más adecuada*?

A. Configurar un *visibility timeout* más largo
B. Configurar una DLQ en SQS y un máximo de reintentos
C. Aumentar el timeout de Lambda para que procese mensajes más grandes
D. Crear una nueva función Lambda que valide mensajes antes del procesamiento

DLQ + máximo de reintentos evita loops infinitos y permite aislar mensajes corruptos. Es el patrón estándar para SQS + Lambda.

Question 2

Una función Lambda debe conectarse a una base de datos en RDS dentro de una VPC privada.
Se ha notado que las primeras invocaciones tardan varios segundos. ¿Cuál es la causa más probable?

A. Falta de memoria asignada
B. Falta de permisos IAM para RDS
C. Latencia causada por la creación de ENI (Elastic Network Interface) durante el cold start
D. IAM Role muy grande

Question 2

Una función Lambda debe conectarse a una base de datos en RDS dentro de una VPC privada.
Se ha notado que las primeras invocaciones tardan varios segundos. ¿Cuál es la causa más probable?

A. Falta de memoria asignada
B. Falta de permisos IAM para RDS
C. Latencia causada por la creación de ENI (Elastic Network Interface) durante el cold start
D. IAM Role muy grande

Cuando Lambda está dentro de una VPC, el cold start implica crear ENIs, lo que genera latencia inicial. Es uno de los escenarios clave del examen.