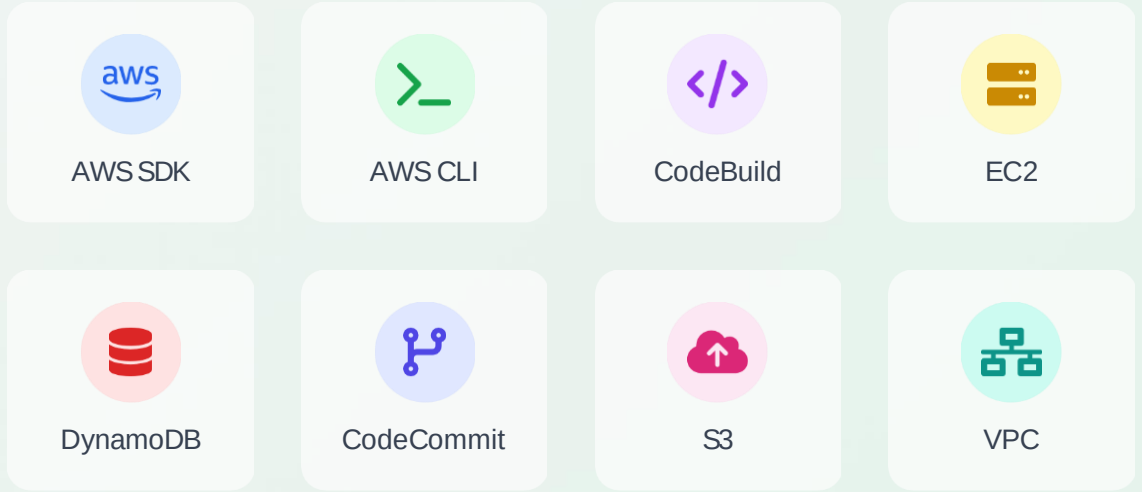
Truy cập Dịch vụ AWS theo Phương thức Lập trình

Hướng dẫn phát triển ứng dụng trên nền tảng đám mây hàng đầu



Report Generate Date: 2025-10-03

Bắt đầu Phát triển trên AWS

Mục Tiêu

- Nắm bắt cách truy cập các dịch vụ AWS theo cách lập trình (programmatically).
- Các mẫu lập trình (programmatic patterns) trong AWS SDKs.
- AWS Developer Tools (AWS Dev Tools / Developer Experience).

Truy cập dịch vụ thông qua SDK

Overview of interacting with AWS REST endpoint

Mỗi dịch vụ AWS (S3, DynamoDB,Ec2) đều có REST API entpoint. Bạn có thể gọi trực tiếp qua HTTP(S).

Ví dụ

```
GET /?list-type=2 HTTP/1.1
Host: my-bucket.s3.us-east-1.amazonaws.com
X-Amz-Date: 20251003T140000Z
Authorization: AWS4-HMAC-SHA256 Credential=... Signature=...
```

Đặc điểm:
Phải ký request bằng **Signature Version 4 (SigV4)**.
Yêu cầu gửi đúng **method (GET/PUT/POST/DELETE)**, query string, header, và body.
Response trả về ở dạng **JSON hoặc XML**.
Quản lý thủ công: pagination, error handling, retries.
☞ **Khó khăn**: Developer phải tự tính toán signature, quản lý request, parse response → dễ lỗi và mất nhiều thời gian.

AWS SDK

Tập hợp các thư viện cho phép nhà phát triển tương tác với các dịch vụ AWS bằng ngôn ngữ lập trình quen thuộc.

Ngôn ngữ được hỗ trợ:

Python (Boto3) | Java | Node.js | .NET | Go | PHP

```
import boto3
s3 = boto3.client('s3')
response = s3.list_buckets()
print(response)
```

Giá trị SDK mang lại:
✓ **Tự động ký request (SigV4)**
Bạn không cần tính toán thủ công Authorization header.
SDK tự dùng credentials từ IAM Role, env var, hay config file.

Cả AWS SDK và AWS CLI đều giúp nhà phát triển tương tác với AWS một cách hiệu quả và tự động hóa

AWS SDK & CLI - Công cụ Lập trình Cốt lõi

Kết Luận

- REST endpoints: cho bạn toàn quyền kiểm soát, nhưng phức tạp, tốn công.
- **AWS SDKs**: đơn giản hóa, tự động hóa phần khó (SigV4, parse response, pagination,error handling).

☞ Trong thực tế, developer **gần như luôn dùng SDK** (hoặc CLI) thay vì gọi REST trực tiếp.

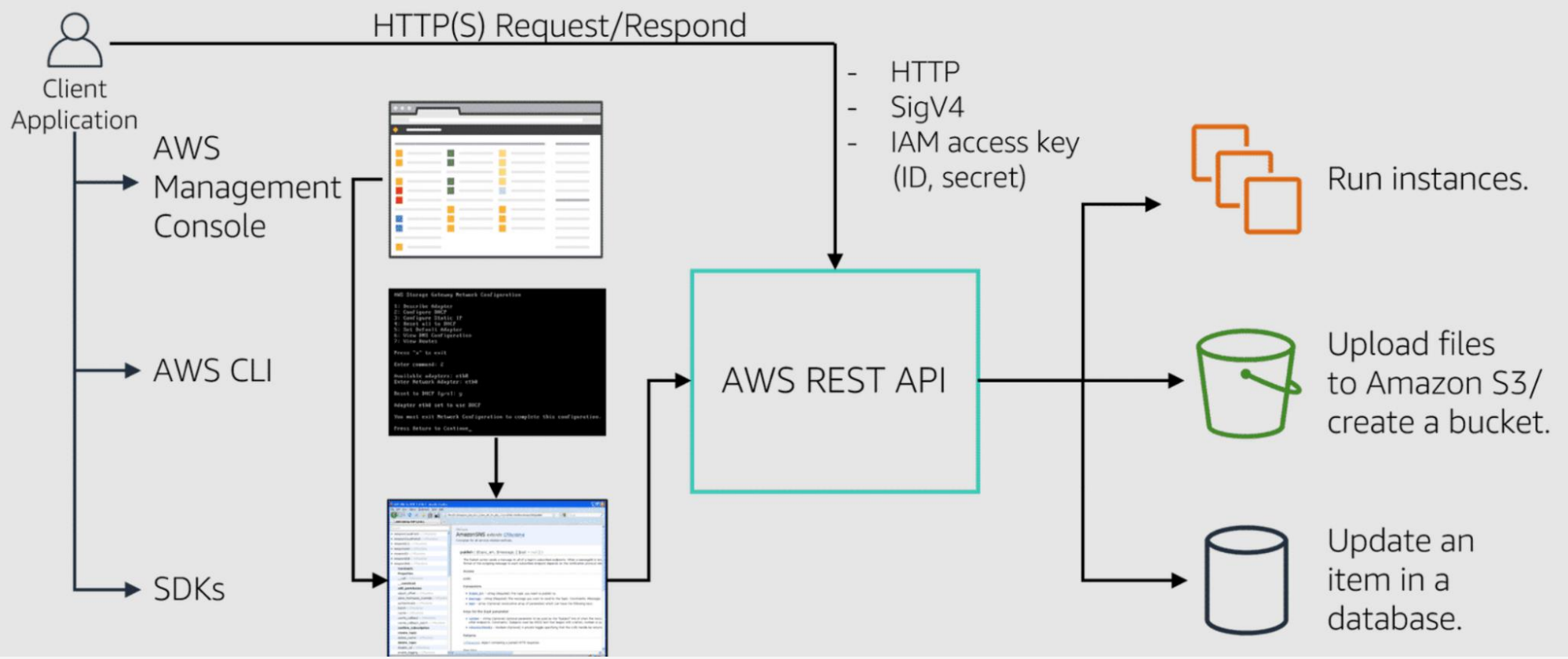
Response from REST

```
<ListAllMyBucketsResult>
  <Buckets>
    <Bucket>
      <Name>my-bucket-1</Name>
      <CreationDate>2025-10-01T12:00:00.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>my-bucket-2</Name>
      <CreationDate>2025-10-02T14:00:00.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

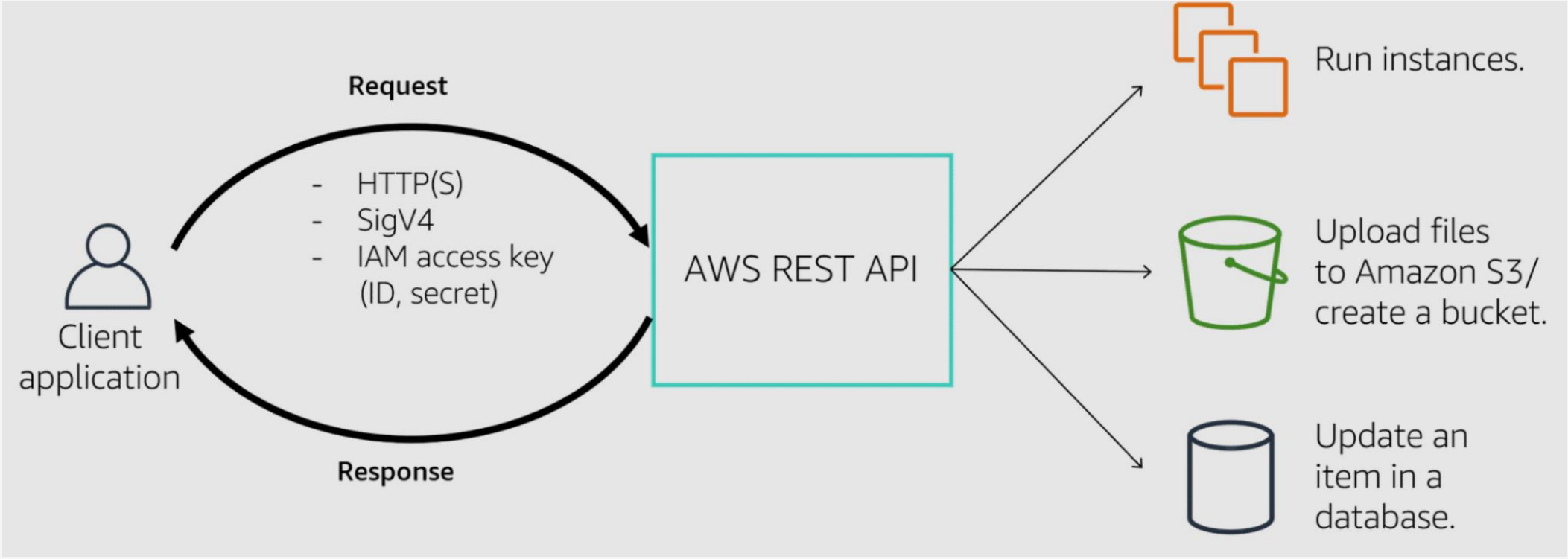
Response from SDK

```
{
  'Buckets': [
    {'Name': 'my-bucket-1', 'CreationDate': datetime(2025, 10, 1, 12, 0)},
    {'Name': 'my-bucket-2', 'CreationDate': datetime(2025, 10, 2, 14, 0)}
  ]
}
```

Truy cập AWS Services



Mỗi dịch vụ đều có API riêng



Quy trình phát triển trên AWS đảm bảo tính hiệu quả, khả năng mở rộng và độ tin cậy cho các ứng dụng

HTTP Request/Response AWS

Một request gửi đi bởi client (ứng dụng, AWS CLI, SDK) thường có 3 phần chính

- Start line
  - Gồm **HTTP method** (GET, POST, PUT, DELETE…) + **Path** + **HTTP version**

Ví dụ: . GET /mybucket/myfile.txt HTTP/1.1

- Headers
  - Chứa thông tin bổ sung (metadata), ví dụ:

Ví dụ:
Host: s3.amazonaws.com
Content-Type: application/json
Authorization: AWS4-HMAC-SHA256 Credential=..

- Body (Payload)
  - Chứa dữ liệu cần gửi (chỉ có trong một số method như POST, PUT).
  - Ví dụ khi upload file lên S3: body chính là nội dung file.

HTTP Request/Response AWS

HTTP Response (từ server → client)

- Start line
  - Gồm **HTTP version + Status code + Status text**

  Ví dụ: . HTTP/1.1 200 OK

- Headers
  - Metadata về response (kích thước dữ liệu, loại dữ liệu, cache, v.v).

    Ví dụ:
    Content-Type: application/json
    Content-Length: 123
    ..

- Body (Payload)
  - Chứa dữ liệu trả về (hoặc thông báo lỗi)..
  - Ví dụ với AWS S3 khi GET Object, body chính là nội dung file.

HTTP Request/Response AWS

Process khi gọi AWS API

- **HTTPS protocol**
  - Toàn bộ request/response tới AWS đều đi qua **HTTPS (TLS)** → đảm bảo dữ liệu không bị nghe lén (confidentiality) và không bị chỉnh sửa giữa đường (integrity).=> bảo vệ đường truyền.
- **SigV4 Authentication**
  - AWS dùng Signature Version 4 (SigV4) để xác thực.
  - Cơ chế này dựa trên IAM-based identity (Access Key ID + Secret Access Key).
  - Client sẽ ký số request (method, headers, payload) để chứng minh request là từ identity hợp lệ.
  - AWS server xác minh lại chữ ký trước khi thực thi.

  => bảo vệ danh tính và tính toàn vẹn của request.

◆ Request Flow to AWS REST API

Plow steps:

- Request (e.g., turn on an EC2 instance)
- Signature Version 4 (SigV4) for authorization
- Route to specific AWS service (EC2, S3, DynamoDB)

AWS Response Structure

AWS Response Elements

1. HTTP Status Codes

- Cho biết kết quả tổng quát của request:
  - 200 OK → thành công.
  - 201 Created → tài nguyên mới được tạo (ví dụ tạo bucket S3).
  - 400 Bad Request → lỗi tham số đầu vào.
  - 403 Forbidden → thiếu quyền IAM.
  - 404 Not Found → tài nguyên không tồn tại.
  - 429 Too Many Requests → vượt hạn mức (throttling).
  - 500 / 503 → lỗi hệ thống AWS.

AWS Response Structure

AWS Response Elements

2. Request ID và Amazon ID (Diagnostics Metadata)

- Mỗi request tới AWS đều sinh ra **ID duy nhất**, giúp debug và trace log:
- Thường có trong header response:
  - x-amzn-RequestId → ID duy nhất cho request.
  - x-amz-id-2 (S3) → Amazon internal ID để truy vết sâu hơn.

  - Example:

```
x-amzn-RequestId: 5c2d8f8a-8f2b-11ee-b9d1-0242ac120002
x-amz-id-2: abcdXYZExampleID1234567890
```

AWS Response Structure

AWS Response Elements

3. **Payload (Response Body)**

- **Thành công** → chứa chi tiết tài nguyên hoặc dữ liệu mà bạn request.
  - Ví dụ: S3 → trả về danh sách bucket, EC2 → chi tiết instance.
  - JSON hoặc XML tùy service.
  - Ví dụ lỗi DynamoDB:

```
{
  "__type": "com.amazonaws.dynamodb.v20120810#AccessDeniedException",
  "message": "User is not authorized to perform: dynamodb:PutItem on resource: Table/Users"
}
```

Tools for AWS Interaction

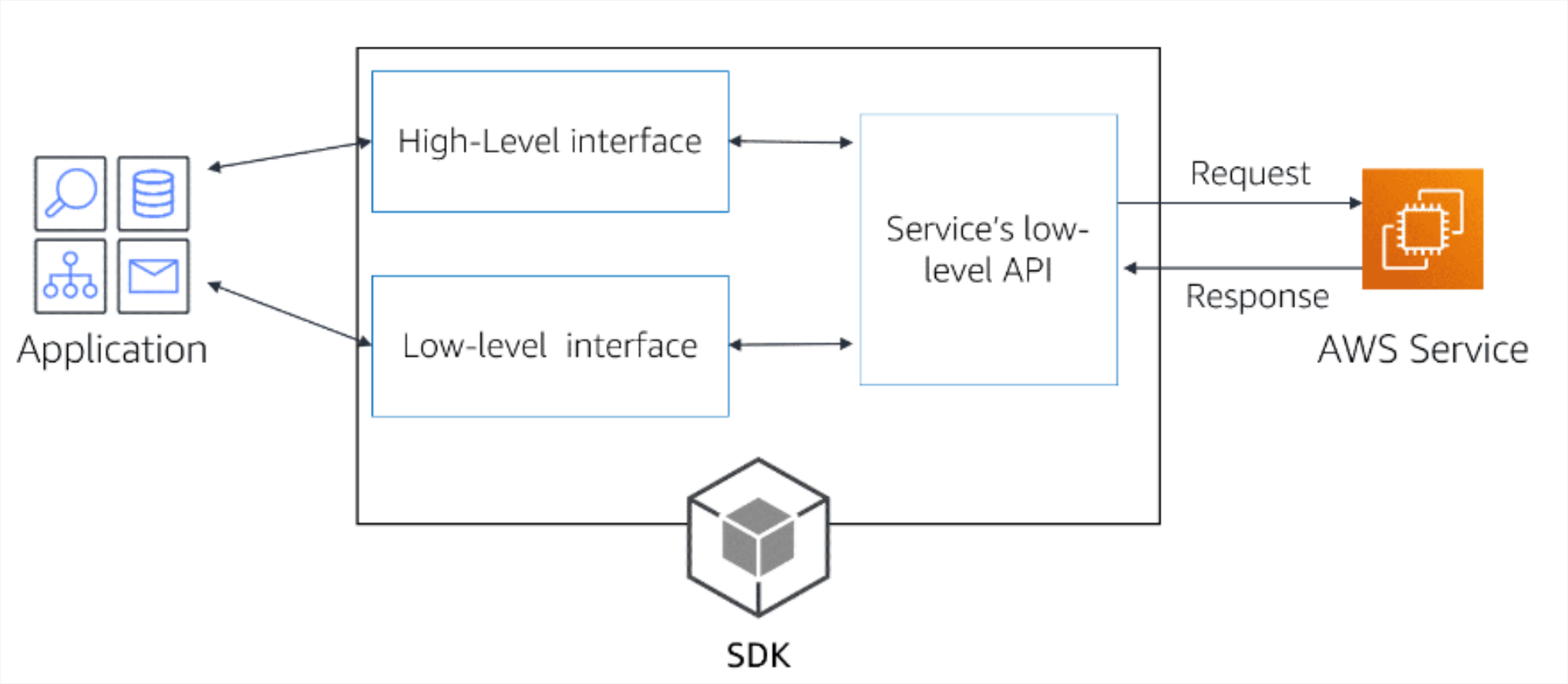
- AWS Management Console:
  - GUI for viewing and managing AWS resources
- AWS CLI:
  - Command-line for scripting and automation
- AWS SDKs:
  - SDKs for various programming languages for application integration

AWS SDK

- **Ngôn ngữ lập trình cụ thể**
  - AWS SDKs có cho nhiều ngôn ngữ: Python (boto3), Node.js (aws-sdk), Java, .NET, Go, PHP...
  - Giúp bạn dùng **code quen thuộc** thay vì phải viết request HTTP thủ công.
- **Tự động xây dựng request**
  - SDK sẽ **tạo HTTP(S) request** chuẩn theo yêu cầu của dịch vụ.
  - Nó sẽ thêm các phần quan trọng:
    - X-Amz-Date
    - Authorization (chứa chữ ký SigV4)
    - Host và các headers khác

    => Bạn chỉ cần gọi hàm trong SDK, không cần lo cách viết request.
- **Gửi đến endpoint dịch vụ**
  - SDK tự động xác định endpoint (ví dụ: S3, EC2, DynamoDB trong region của bạn).
- **Xử lý response**
  - Nếu thành công → trả về data (thường dưới dạng object, dictionary, hoặc JSON).
  - Nếu thất bại → trả về **HTTP error code** (403, 404, 500…) kèm error message rõ ràng.

AWS SDK


High-Level interface
Low-level interface
Service's low-level API
Request
Response
AWS Service
Application
SDK

API command: **Low-level API** và **High-level API**.

◆ 1. Low-level API

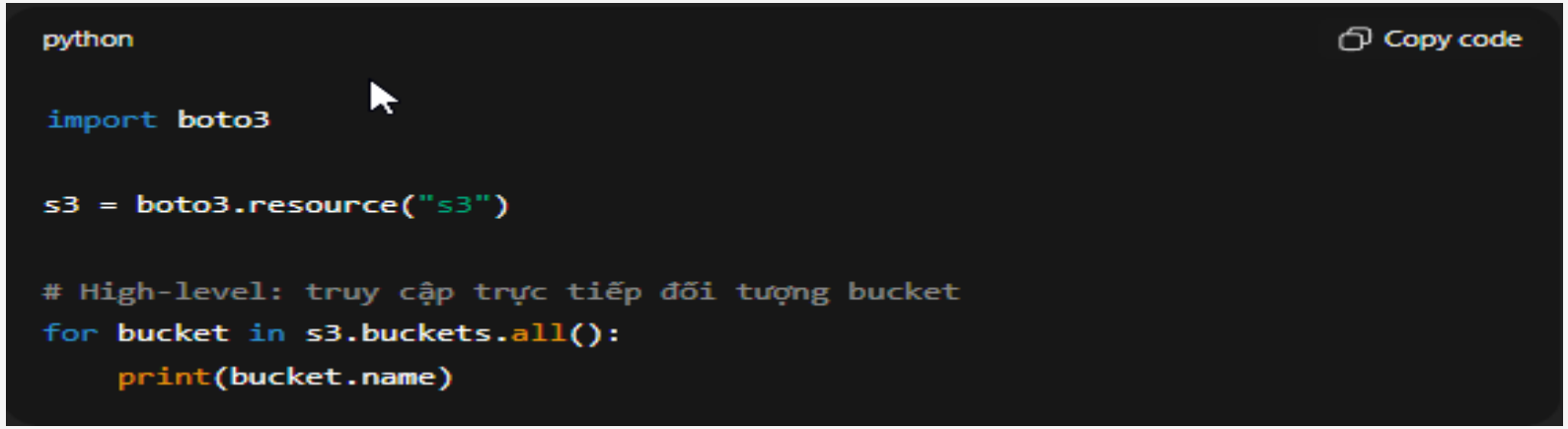
- Gần như là một-một (1:1) với AWS Service API gốc (REST/Query API).
- Bạn phải cung cấp **nhiều chi tiết** (tham số, cấu hình).
- Linh hoạt hơn nhưng **dễ dài dòng**.
- Thường dùng khi bạn cần toàn quyền kiểm soát.
- Example:

```python
import boto3

s3 = boto3.client("s3")

# Gọi API ListBuckets (tương ứng REST API)
response = s3.list_buckets()

for bucket in response["Buckets"]:
    print(bucket["Name"])
```

API command: **Low-level API** và **High-level API**.

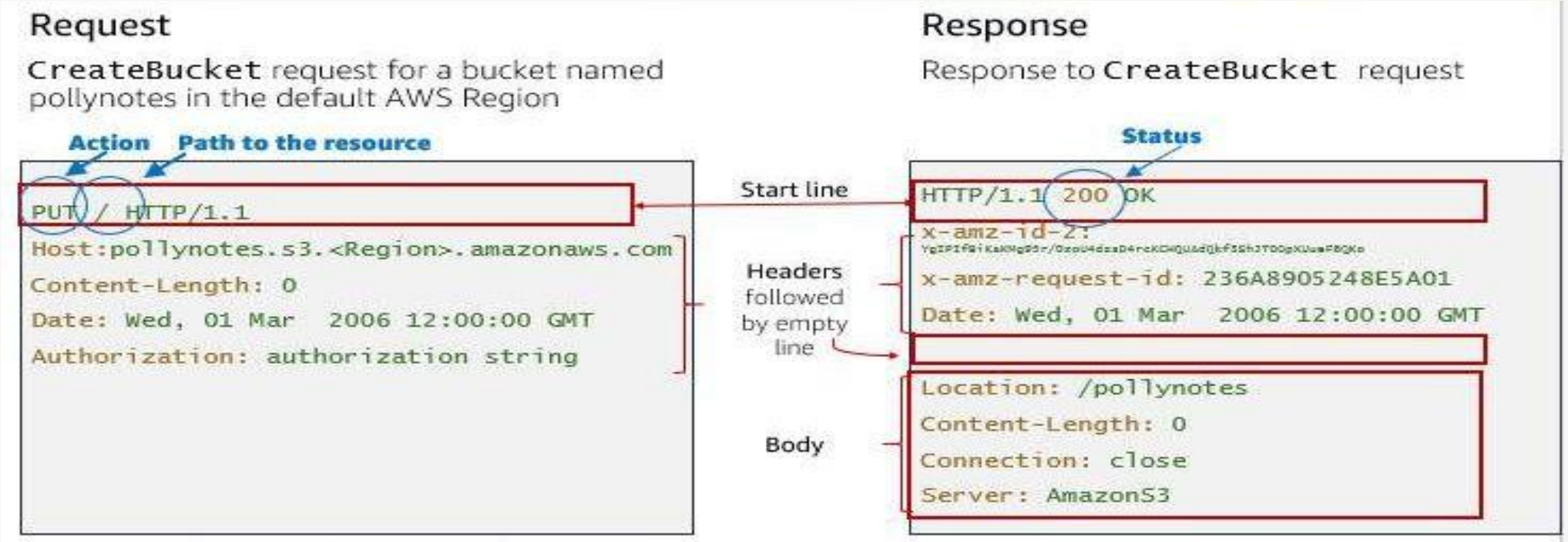
◆ 1. High-level API

- Được SDK xây dựng sẵn **trên nền low-level** để **đơn giản hóa** công việc thường dùng.
- Ẩn bớt chi tiết (request/response), giúp code ngắn gọn hơn.
- Thường được cung cấp qua **Resource API** hoặc **Wrapper functions**.
- Example:



AWS Command Line Interface (AWS CLI)

Example: HTTP request/response for Amazon S3 API to create a bucket



AWS Command Line Interface (AWS CLI)

Running the AWS Command Line Interface (CLI)

You can use AWS services by running the AWS Command Line Interface (AWS CLI) commands from your terminal program.

- Invoke locally:
  - Linux shells or macOS – Run commands through common shell programs such as bash, zsh, and tcsh
  - Windows command line – Run commands at the Windows command prompt or in PowerShell
- Invoke remotely:
  - Connect to Amazon Elastic Compute Cloud (Amazon EC2) instances through a remote terminal program such as PuTTY or Secure Shell (SSH)
  - Run AWS CLI commands from AWS Cloud9, AWS CloudShell or AWS Systems Manager Session

AWS Command Line Interface (AWS CLI)

AWS CLI Command structure

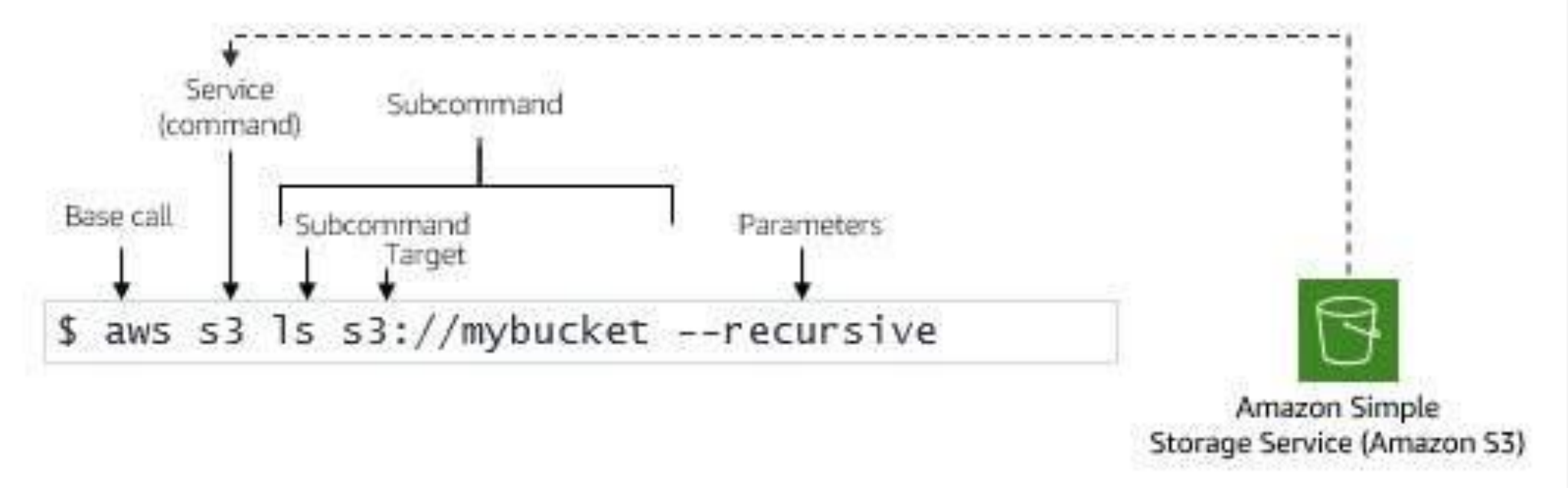
```
aws <command> <subcommand> [options and parameters]
```

AWS CLI commands are formatted in this order:
•aws - base call program
•<command> - AWS service supported by the CLI
•<subcommand> - operation to be performed
•[options and parameters] - information required by the operation

AWS Command Line Interface (AWS CLI)

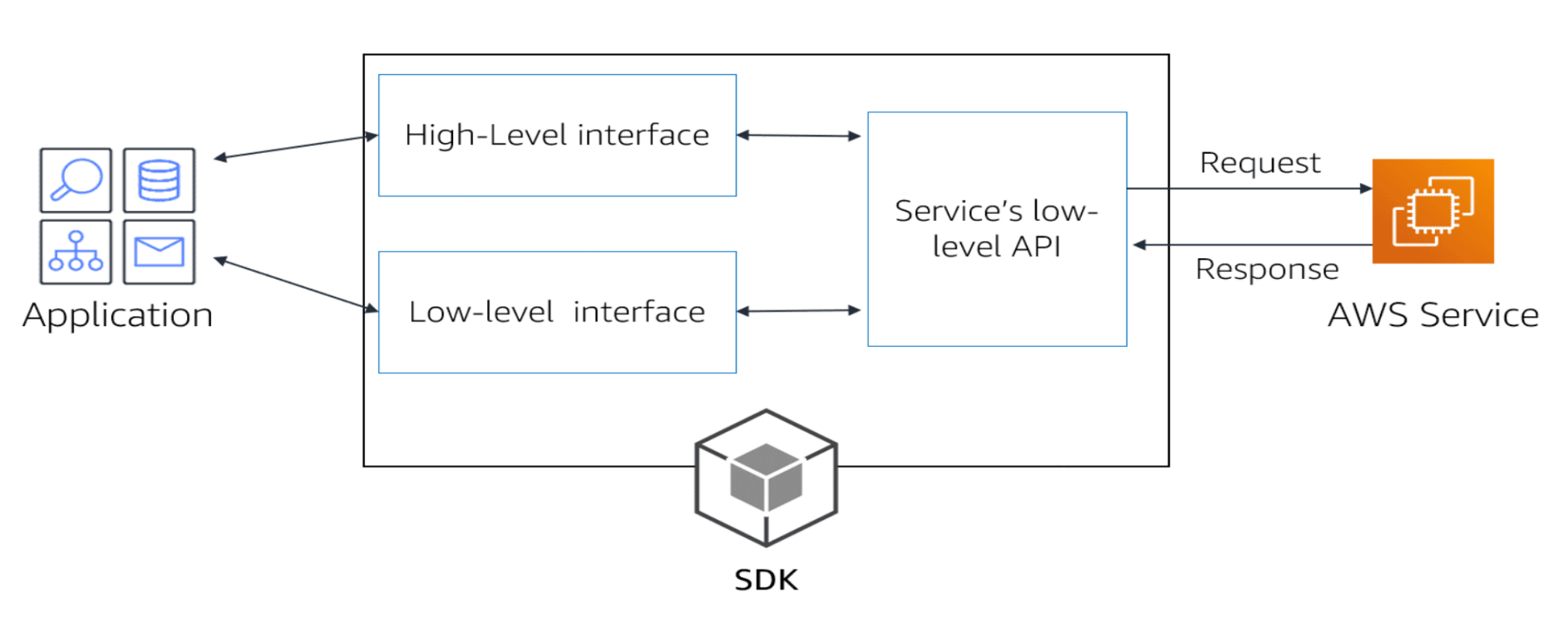
Example: Recursively list all all the contents of an S3 bucket

```
aws s3 ls s3://mybucket --recursive
```



AWS SDK and Programming Patterns

Synchronous vs Asynchronous Invocation



AWS SDK and Programming Patterns

◆ Đặc điểm của synchronous interactions

- Blocking: Client chờ cho đến khi service xử lý xong và trả về kết quả.
- Độ trễ phụ thuộc: Thời gian phản hồi = network latency + thời gian xử lý service.
- Đơn giản: Dễ lập trình và dễ debug.
- Hạn chế: Không phù hợp với các tác vụ dài, có thể gây timeout.

◆ Use case

- **Ứng dụng web/mobile**: cần phản hồi ngay (login, lấy dữ liệu).
- **Batch nhỏ, query nhanh**: DynamoDB getItem, S3 getObject.
- **Lambda gọi Lambda đồng bộ** khi cần kết quả ngay để xử lý tiếp.

AWS SDK and Programming Patterns

◆ **Đặc điểm** Asynchronous interactions

- **Asynchronous interactions** nghĩa là **client gửi request nhưng không chờ kết quả ngay lập tức**.
- Request được **đưa vào hàng đợi (queue)** hoặc **gửi sự kiện (event)**.
- AWS service sẽ xử lý sau.
- Kết quả có thể được lưu (ví dụ: S3, DynamoDB) hoặc gửi thông báo (SNS, EventBridge, SQS).

👉 Điều này giúp **decouple các service**, tăng **scalability** và **resilience**

◆ **Use case**

- **Xử lý background job**: Ví dụ resize ảnh, video transcoding.
- **Event-driven microservices**: Order → Payment → Shipping.
- **High-throughput ingestion**: Logs, IoT events gửi qua Kinesis hoặc SQS.
- **Workflow dài**: ML training, ETL jobs.

AWS SDK and Programming Patterns

◆ **Waiters trong AWS SDK for** asynchronous operations.

- Waiter = built-in utility trong AWS SDK (Python boto3, Java, .NET, Node.js...)
- Nó sẽ poll API theo chu kỳ cho đến khi trạng thái resource đạt điều kiện mong muốn hoặc timeout.
- Giúp code asynchronous nhưng developer chỉ cần viết code “chờ cho xong”.

python Copy code

```python
import boto3

ec2 = boto3.client("ec2")

# Start an EC2 instance
ec2.start_instances(InstanceIds=['i-1234567890abcdef0'])

# Use waiter to wait until it's running
waiter = ec2.get_waiter('instance_running')
waiter.wait(InstanceIds=['i-1234567890abcdef0'])

print("✅ EC2 is now running")
```

Gaining Insight into your Application