AWS Serverless Introduction

AWS main serverless components

- AWS Lambda
- DynamoDB
- AWS Cognito
- AWS API Gateway
- Amazon S3
- AWS SNS & SQS
- AWS Kinesis Data Firehose
- Aurora Serverless
- Step Functions
- Fargate

What is Serverless?

1. Allows developers to **create** and **run** apps and services **without** having to worry about **infrastructure**.
2. Apps still **run** on **servers**, but **AWS handles** all of the server management.
3. AWS Serverless includes from **FaaS** (lambda) to **Databases** (dynamoDB), **Api Gateway**, **Queues**, **Storage** etc.
4. Developers can focus on building **product features** instead of managing servers.

AWS Lambda - Features (1)

Amazon Lambda: a **FaaS**, which allows developers to **run code without worrying** about infrastructure or servers.

1. **Pay** for the **consumed** time on computing.
2. Runs on **demand**.
3. Supports auto **concurrency** (up to 1000 concurrent executions) and **scaling** controls.
4. Integrates with **other AWS services**.

AWS Lambda - Features (2)

5. Supports many **programming** languages (Node.js, Python, Java, Golang, Custom Runtime API).

6. **Configure RAM** to increase compute power.

7. Supports **Docker** via Lambda Container Image.

8. Supports **File systems access.**

9. Supports **Lambda Layers** to externalize dependencies (e.g. npm packages) for reusability among other lambdas.

AWS Lambda - Interactions

API Gateway

Kinesis

DynamoDB

S3

CloudFront

CloudWatch Events
EventBridge

CloudWatch Logs

SNS

SQS

Cognito

AWS Lambda - Invocation Sync

1. We can choose to invoke a lambda **synchronously** or **asynchronously.**

2. With **synchronous** invocation, we **wait** for the function to process the event and return a **response** or it **times out**.

3. Following Amazon services support sync invocation: **ALB, Cognito, Lex, Alexa, API Gateway, CloudFront (Lambda@Edge), Kinesis Data Firehose**

AWS Lambda - Invocation Async

1. With **asynchronous** invocation, Lambda **queues** the event for processing and returns a **response immediately**.

2. Handles **retries** and can send invocation records to a destination.

3. We have to set the invocation type **parameter** to **Event**.

4. Following AWS services supports async invocation: **S3, SNS, SES, CloudFormation, CloudWatch Logs, CloudWatch Events, CodeCommit**

AWS Lambda - Event Source Mapping

1. **Polls** records from services (**Kinesis, SQS, DynamoDB Streams**) and **invokes** functions.

2. Batch of records are **pulled** from **poller**.

3. Event Source Mapping handles the polling and invokes lambda **synchronously**.

AWS S3 - Features

Amazon S3 is a cloud **object storage** service, which is used to store and protect any amount of data

1. **Scalability.**
2. Data **availability** (99.9%).
3. High **durability** (99.999999999%) of objects across multiple AZs.
4. **Security** / **Encryption**.
5. **Versioning**.
6. Files can have **size** from 0B to 5TB.
7. **Unlimited** storage.
8. **Account -> Bucket -> Object** (Files are stored in buckets as objects using a key)

AWS S3 - Storage Classes

1. **S3 Standard**: high durability, availability, and performance object storage for *frequently accessed data*.
   a. cloud applications, dynamic websites, content distribution.

2. **S3 Standard-Infrequent Access:** data that is accessed *less frequently*, *rapid access* when needed.
   a. long-term storage, backups, data store for disaster recovery files.

3. **Amazon Glacier:** Used for *data archiving*, highest performance, most retrieval flexibility
   a. lowest cost archive storage in the cloud.

4. **Amazon Glacier & Glacier Deep Archive:** lowest-cost storage class and supports *long-term retention*.
   a. use cases like access once or twice in a year

AWS S3 Storage Classes Comparison

	S3 Standard	S3 Intelligent-Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved

https://aws.amazon.com/s3/storage-classes/

AWS SQS - Features (1)

Amazon SQS is a **secure**, **durable hosted queue** that allows to integrate and **decouple** distributed software services.

1. Default SQS supports **at least once delivery** (Can have duplicate messages).
2. FIFO SQS, supports **exactly once delivery** (No duplicates).
3. **Security**: *Encryption* (In-flight encryption / At-rest encryption / Client-side encryption), *Access Controls* (IAM policies), *SQS Access Policies*.
4. **Availability**: highly-concurrent access to messages and high availability for messages.

AWS SQS - Features (2)

5. **Reliability**: locks messages during processing, multiple producers can send and multiple consumers can receive messages at the same time.

6. **Scalability**: scales transparently to handle any load increases.

7. **Holds** message until a **consumer deletes** it.

8. Message **retention up to 14** days.

9. **Message batches**: can reduce cost and increase throughput.

AWS SQS - Features (3)

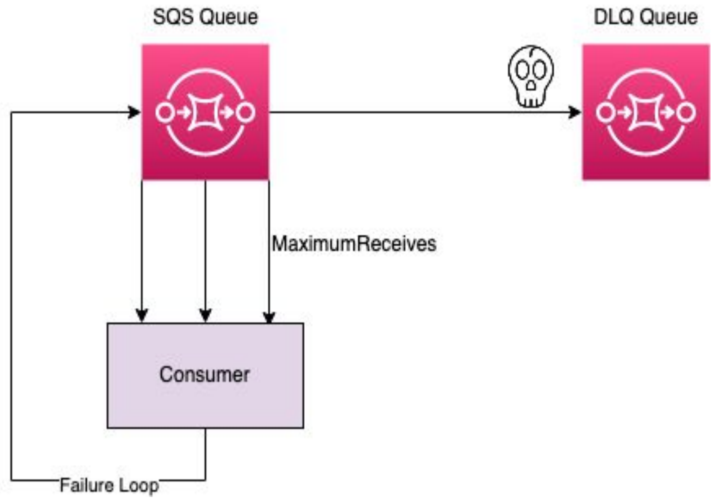
10. **Message Visibility Timeout:** When a message is polled, it becomes invisible to other consumers. Default visibility timeout is 30sec. After the message visibility timeout is over, the message is "visible" in SQS again.

11. Can **delay** messages to consumers up to 15 minutes (default is 0 seconds)

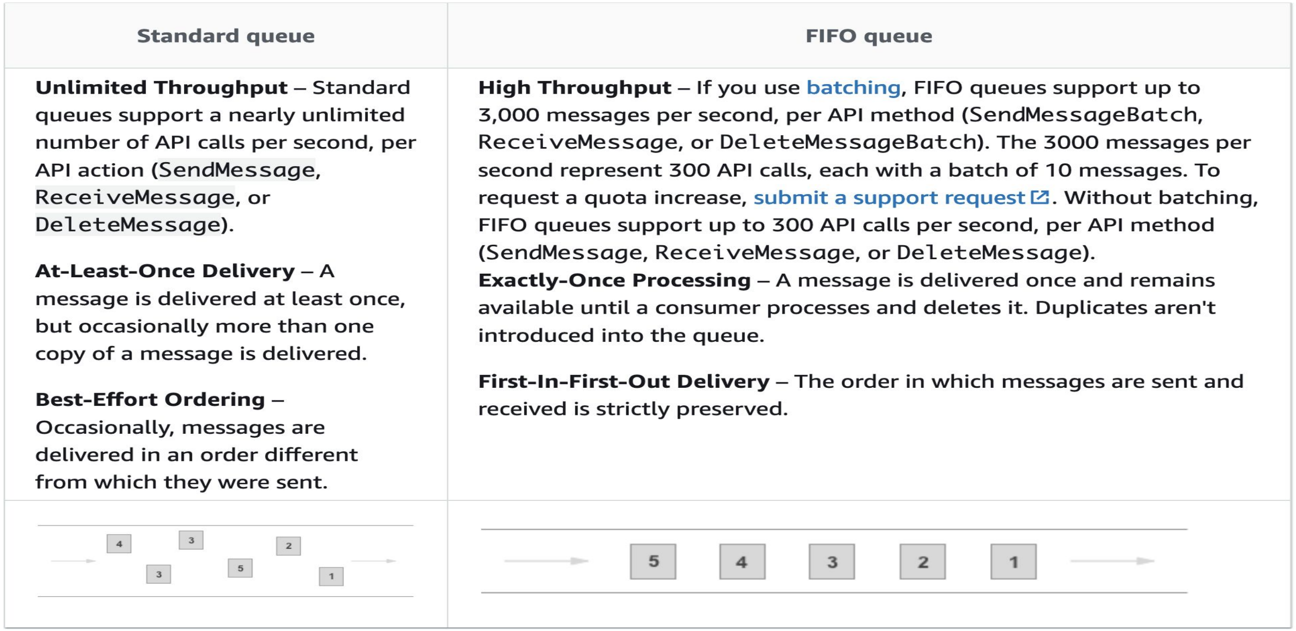
12. Supports **Long Polling,** as consumer can optionally "wait" for messages.

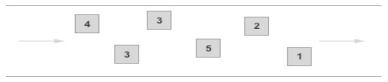
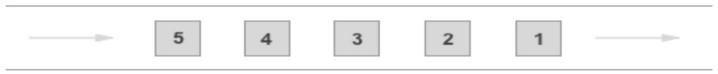
AWS SQS - Dead Letter Queue (DLQ)

- When a **consumer fails to process** a message before the *Visibility Timeout expires*, the message is returned to the queue.
- There is a configurable **limit** (**MaximumReceives**) on how **many times** a message can be **returned** to the queue.
- When we **pass the limit**, the message can be placed in the dead letter queue (**DLQ**)



AWS SQS - Standard vs FIFO



Standard queue	FIFO queue
Unlimited Throughput – Standard queues support a nearly unlimited number of API calls per second, per API action (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`). **At-Least-Once Delivery** – A message is delivered at least once, but occasionally more than one copy of a message is delivered. **Best-Effort Ordering** – Occasionally, messages are delivered in an order different from which they were sent.	**High Throughput** – If you use batching, FIFO queues support up to 3,000 messages per second, per API method (`SendMessageBatch`, `ReceiveMessage`, or `DeleteMessageBatch`). The 3000 messages per second represent 300 API calls, each with a batch of 10 messages. To request a quota increase, submit a support request. Without batching, FIFO queues support up to 300 API calls per second, per API method (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`). **Exactly-Once Processing** – A message is delivered once and remains available until a consumer processes and deletes it. Duplicates aren't introduced into the queue. **First-In-First-Out Delivery** – The order in which messages are sent and received is strictly preserved.
	

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html

AWS SNS - Features (1)

Amazon SNS is a managed service that provides **message delivery** from **publishers to subscribers** (pub-sub).

1. Sends messages to a **topic**.
2. Individual receiver / subscriber OR multiple receivers / subscribers (**Fan-Out**).
3. Supports **Standard** and **FIFO** topics. FIFO topic ensures strict message ordering, and prevent message duplication.

AWS SNS - Features (2)

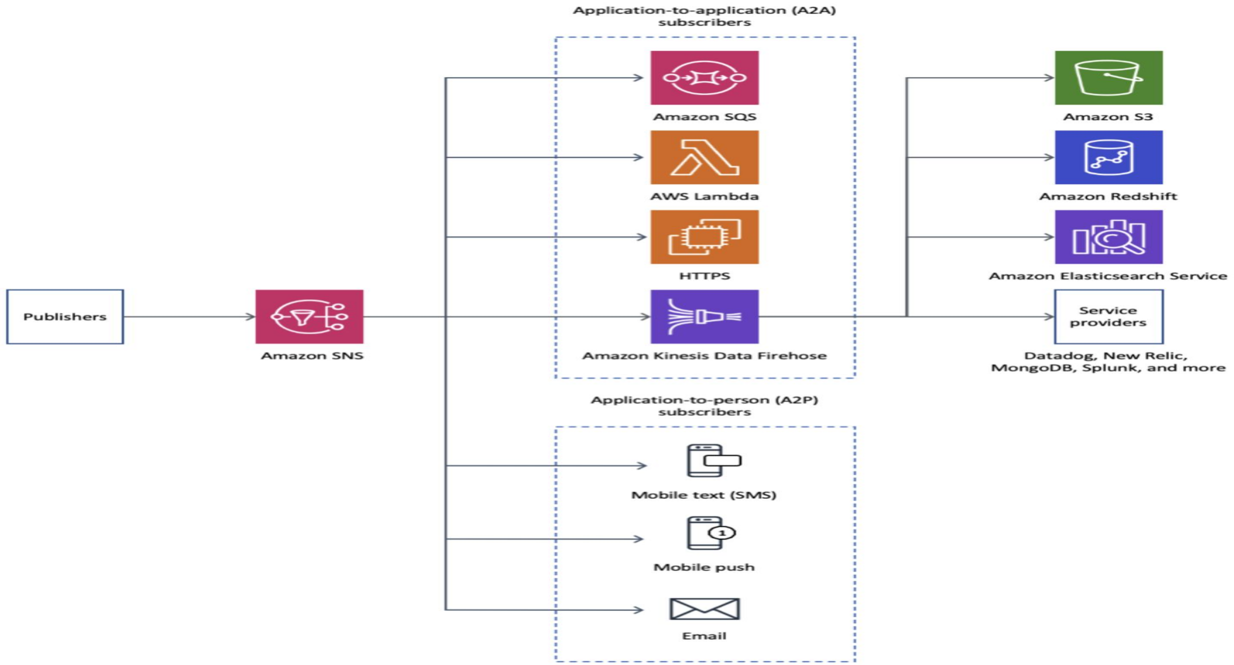
4. **Message filtering**. By default, each subscriber receives every message published to the topic. To receive a subset of the messages, a subscriber must assign a filter policy to the topic subscription

5. **Message durability,** supports **retry mechanism.**

6. **Application-to-application messaging**: SQS, HTTP / HTTPS, Lambda, Kinesis Data Firehose

7. **Application-to-person notifications**: Emails, SMS messages, Mobile Notifications

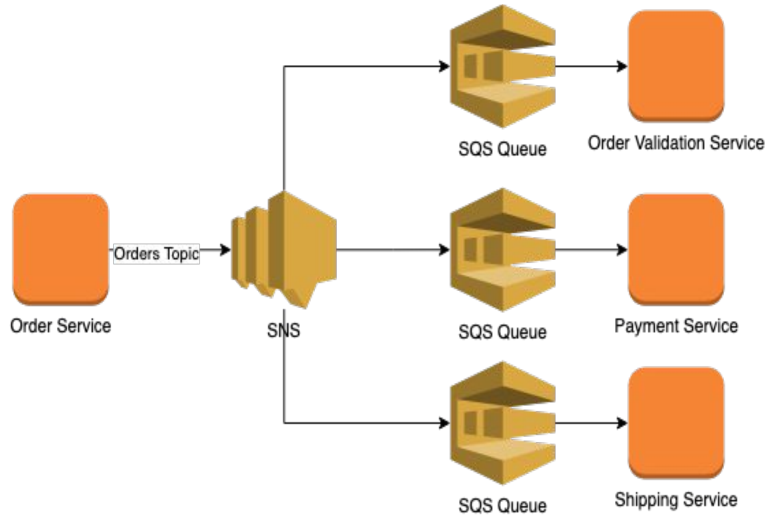
AWS SNS - Integrations


Application-to-application (A2A) subscribers
Publishers
Amazon SNS
Amazon SQS
AWS Lambda
HTTPS
Amazon Kinesis Data Firehose
Amazon S3
Amazon Redshift
Amazon Elasticsearch Service
Service providers
Datadog, New Relic, MongoDB, Splunk, and more
Application-to-person (A2P) subscribers
Mobile text (SMS)
Mobile push
Email

https://docs.aws.amazon.com/sns/latest/dg/welcome.html

AWS SNS + SQS: Fan Out

- Combine SNS and SQS to support **Fan-Out Pattern** (1 publisher / multiple subscribers).
- **Push** on specific SNS **topic**.
- **Multiple SQS queues** are subscribed to topic.
- Fully **decoupled**, **no** data **loss**.
- Take advantage of SQS features like **delays, retries**, **DLQ**



AWS API Gateway - What is?

- Amazon API Gateway allows developers to **create**, **publish**, maintain, monitor, secure **REST APIs, HTTP APIs** and **WebSocket APIs** at any scale.

- REST APIs in API Gateway are **HTTP-based**, **stateless** and support **standard HTTP methods** such as GET, POST, etc.

- For **WebSocket** APIs then, API Gateway enables **stateful**, route incoming messages based on message content.

AWS API Gateway - Features (1)

1. Stateless (**HTTP and REST**) APIs.
2. Stateful (**WebSocket**) APIs.
3. **Authentication** and **Authorization.**
4. API **versioning** (v1, v2...).
5. **Canary release deployments** to avoid breaking changes.
6. **Custom domain** names.

AWS API Gateway - Features (2)

7. Supports **logging** and **monitoring** of APIs using AWS CloudTrail and CloudWatch.
8. Handles **request throttling.**
9. **Aggregates** and **validates** requests and responses.
10. **Caches** responses.
11. Low cost and **efficient**.
12. **Performance** at any scale.

AWS API Gateway - Integration Types(1)

1. **MOCK**:
   a. Integrates the API method request with the API Gateway as a "loop-back" endpoint without invoking any backend.

2. **AWS**:
   a. Lets an API expose AWS service actions (e.g. Lambda).
   b. Must configure both the integration request and integration response.
   c. Must set up necessary data mappings among request response and the opposite.

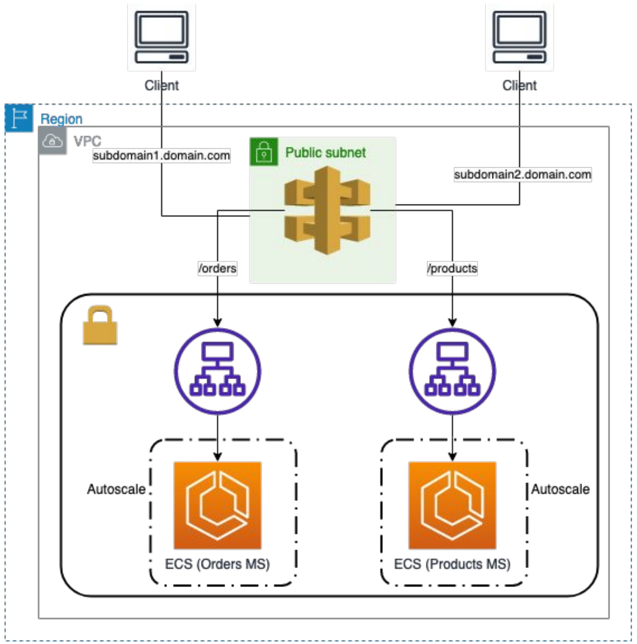
3. **HTTP**:
   a. Lets an API expose HTTP endpoints in the backend.
   b. Must configure both the integration request and integration response.
   c. Must set up necessary data mappings among request response and the opposite.

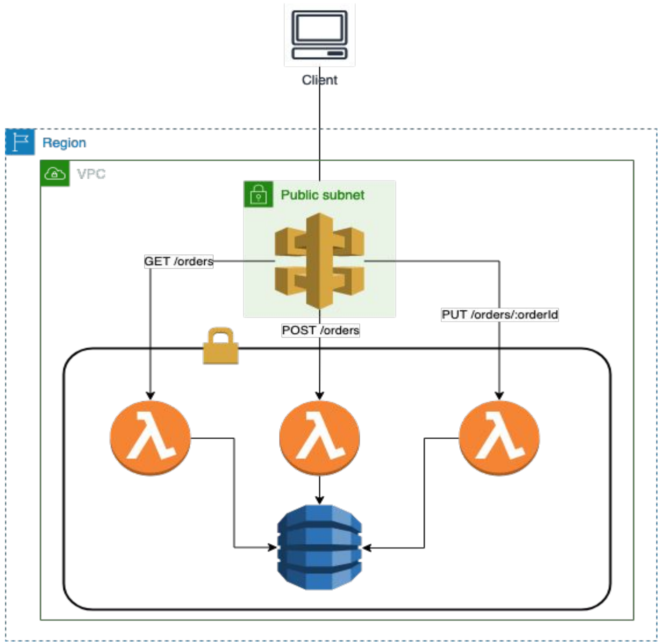
AWS API Gateway - Integration Types(2)

3. **AWS_PROXY** (Lambda Proxy)
   a. Integrate the API method request with the Lambda function-invoking action with the client request passed through as-is.
   b. No need to set the integration request or the integration response.
   c. FaaS is responsible for the logic of request / response
   d. No mapping template, headers, query string parameters... are passed as arguments

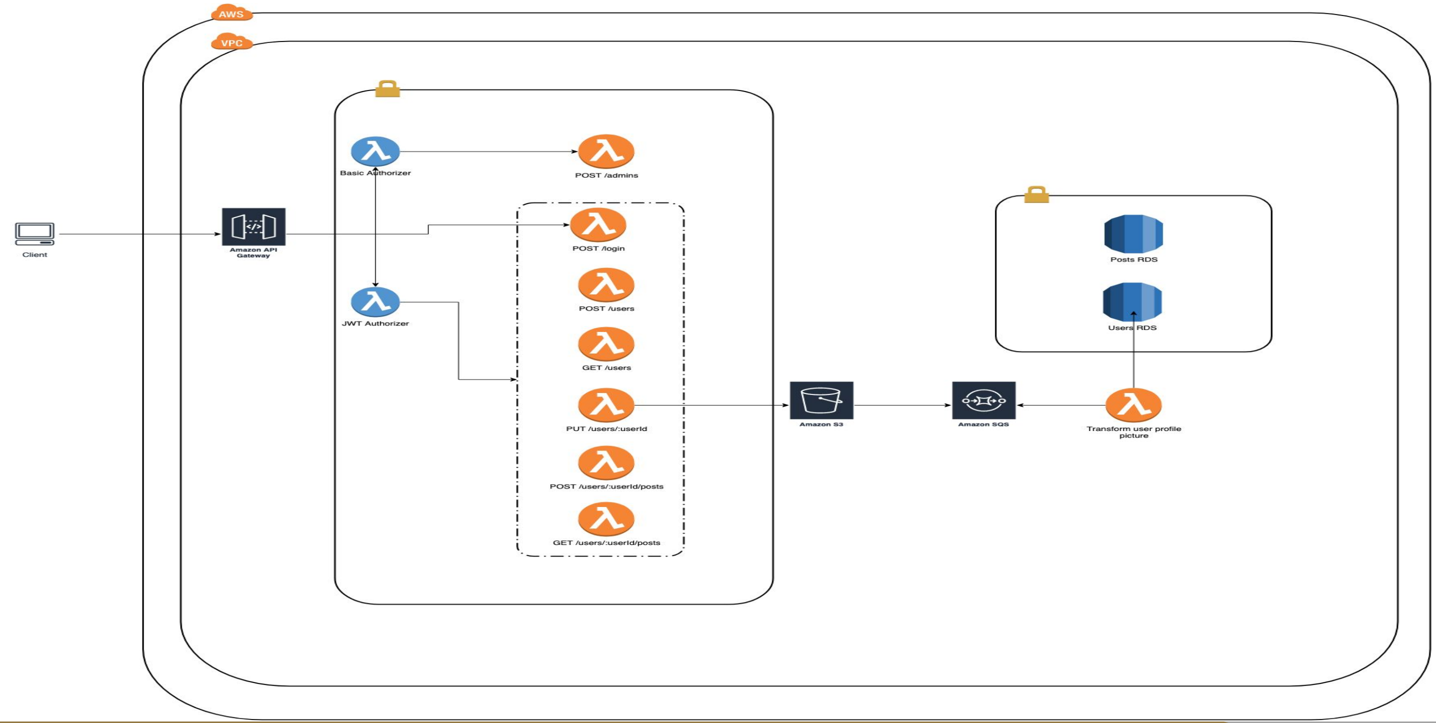
4. **HTTP_PROXY**
   a. Integrate the API method request with the HTTP endpoint with the client request passed through as-is.
   b. No mapping template .
   c. The HTTP request is passed to the backend.
   d. The HTTP response from the backend is forwarded by API Gateway.

API Gateway - Architecture Flow





Serverless (API-Gateway and Lambdas)



Let’s see some code snippets!

(https://github.com/arconsis/aws-network-microservices-warmup)

References

- https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
- https://docs.aws.amazon.com/sns/latest/dg/welcome.html
- https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html
- https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
- https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html
- https://www.udemy.com/course/aws-certified-solutions-architect-associate-saa-c02/

Thank you

Alexandros and Dimosthenis

arconsis