AWS SQS & SNS

CLARUSWAY
WAY TO REINVENT YOURSELF

Table of Contents

1

SQS

SQS

What is a Queue?

A message queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures. Messages are stored on the queue until they are processed and deleted. Each message is processed only once, by a single consumer. Message queues can be used to decouple heavyweight processing, to buffer or batch work, and to smooth spiky workloads.

SQS

What is a Message Queue?

A message queue is fundamentally any technology that acts as a buffer of messages — it accepts messages and lines them up in the order they arrive. When these messages need to be processed, they are again taken out in the order they arrive.

A message is any data or instruction added to the message queue. Going with our example, a message would be the details of an order that could be added to the message queue and then later processed by the payment service.

The architecture of a message queue is simple — applications called the **producers** create messages and add them to the message queue. Other applications called the **consumers** pick up these messages and process them. Some examples of message queues are Apache Kafka, RabbitMQ, and LavinMQ, among others.

SQS

What is a Decoupling?

- Decoupling refers to components remaining autonomous and unaware of each other as they complete their work for some greater output.
- This decoupling can be used to describe components that make up a simple application, or the term even applies on a broad scale.
- For example, many point to the fact that in public cloud computing, the architecture is decoupled in that someone like Amazon will completely handle the physical infrastructure of the network for us, while we work with the data and applications hosted on this hardware.
- We are not really sure what they are up to, while they are not entirely positive what we are doing!

SQS

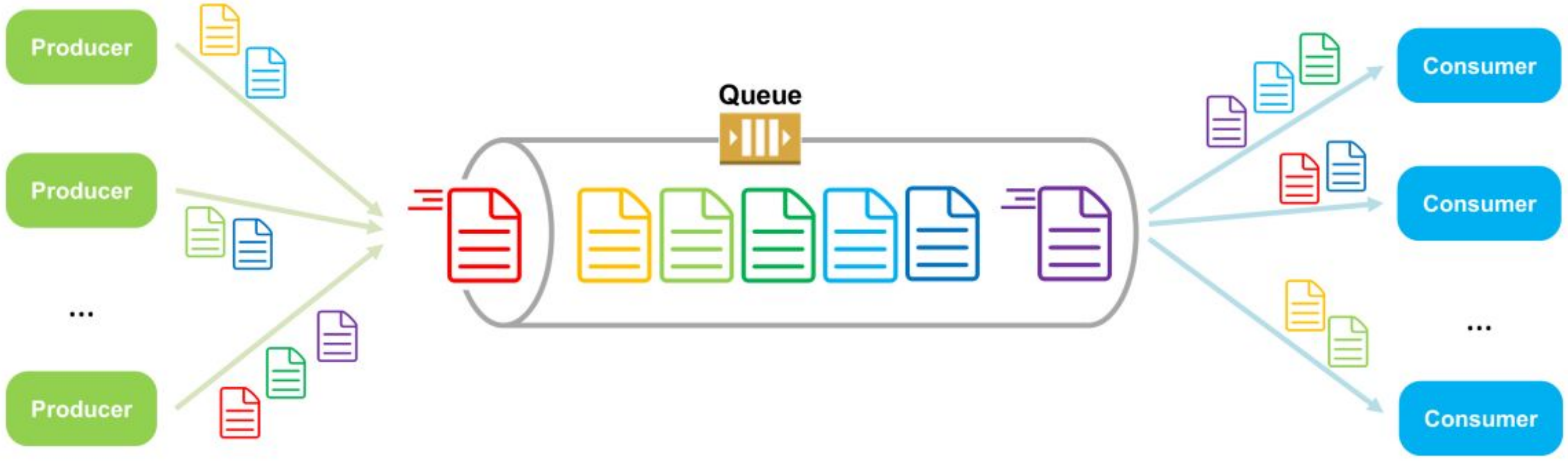
What is SQS?

amazon
SQS

- Amazon **Simple Queue Service (SQS)** is a fully managed **message queuing service** that enables you to decouple and scale microservices, distributed systems, and serverless applications.

SQS

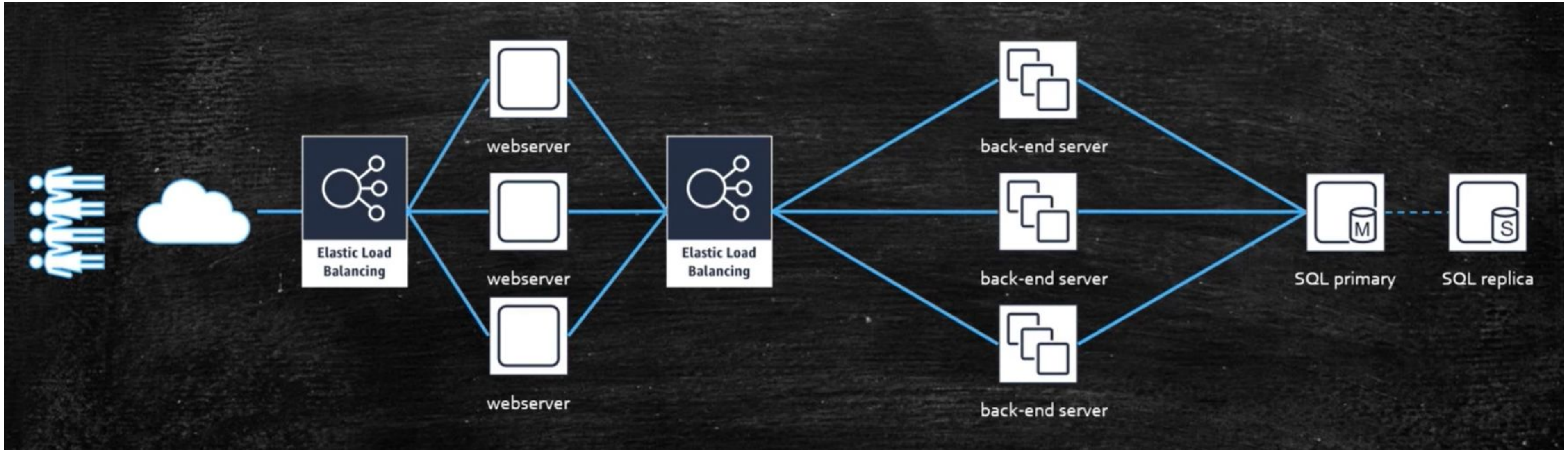
What is SQS?



CLARUSWAY
WAY TO REINVENT YOURSELF

SQS

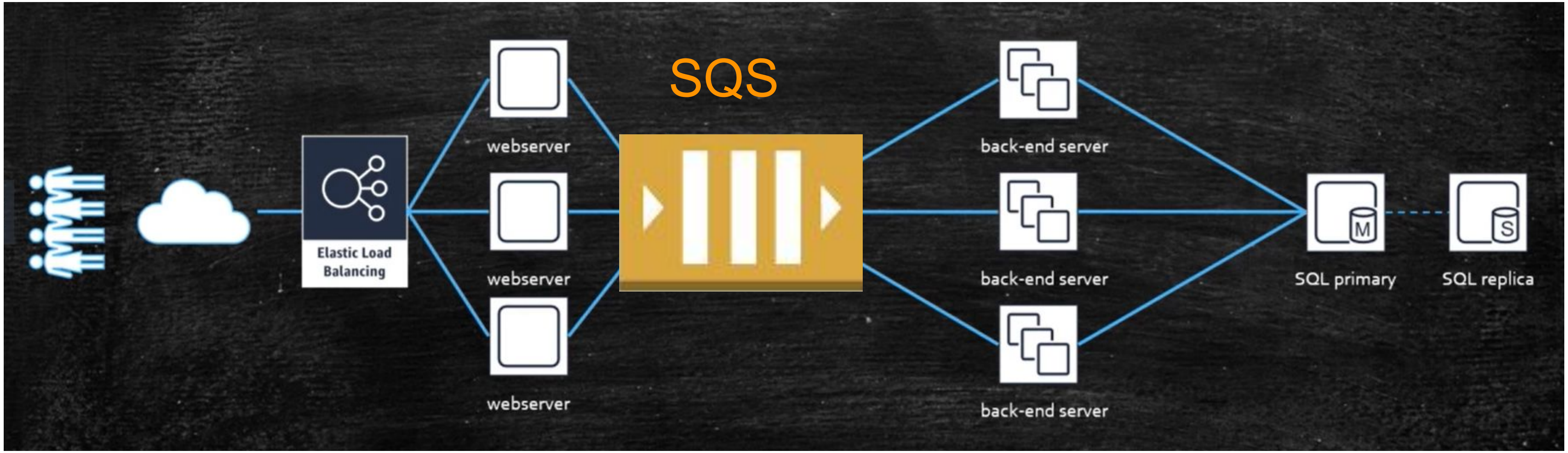
What is SQS?



SQS

What is SQS?

Decoupling

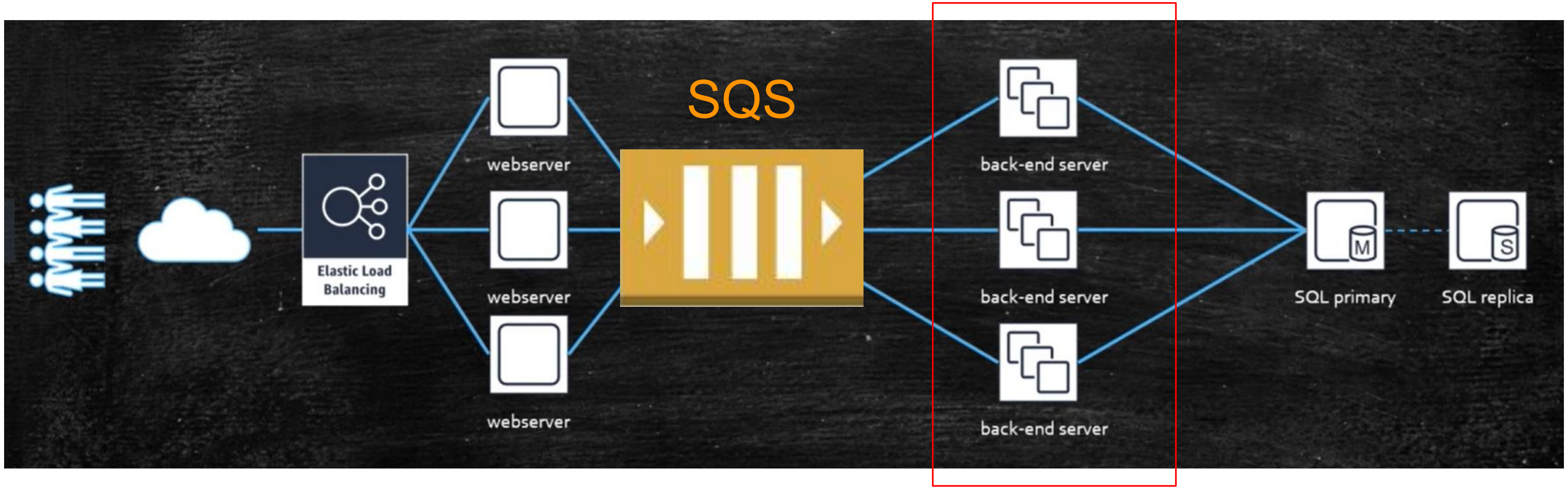


SQS

What is SQS?

CloudWatch Metric – Queue Length
ApproximateNumberOfMessages

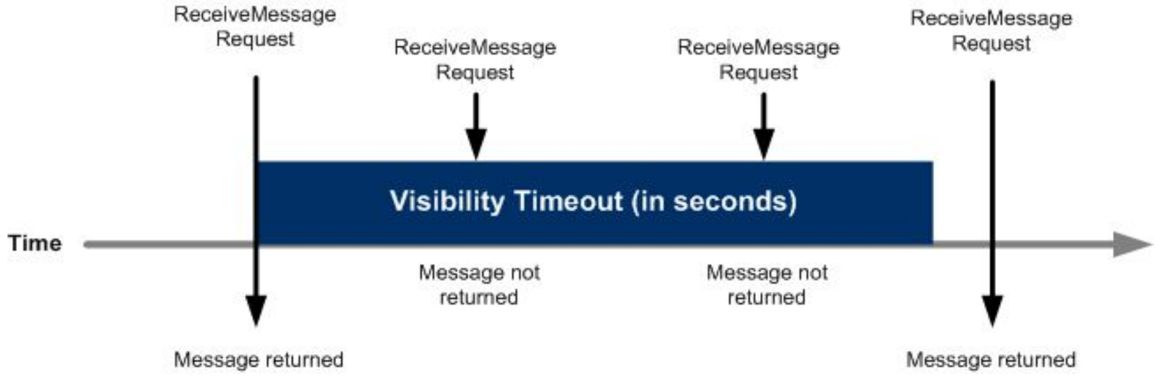
ASG



SQS

Message Visibility Timeout

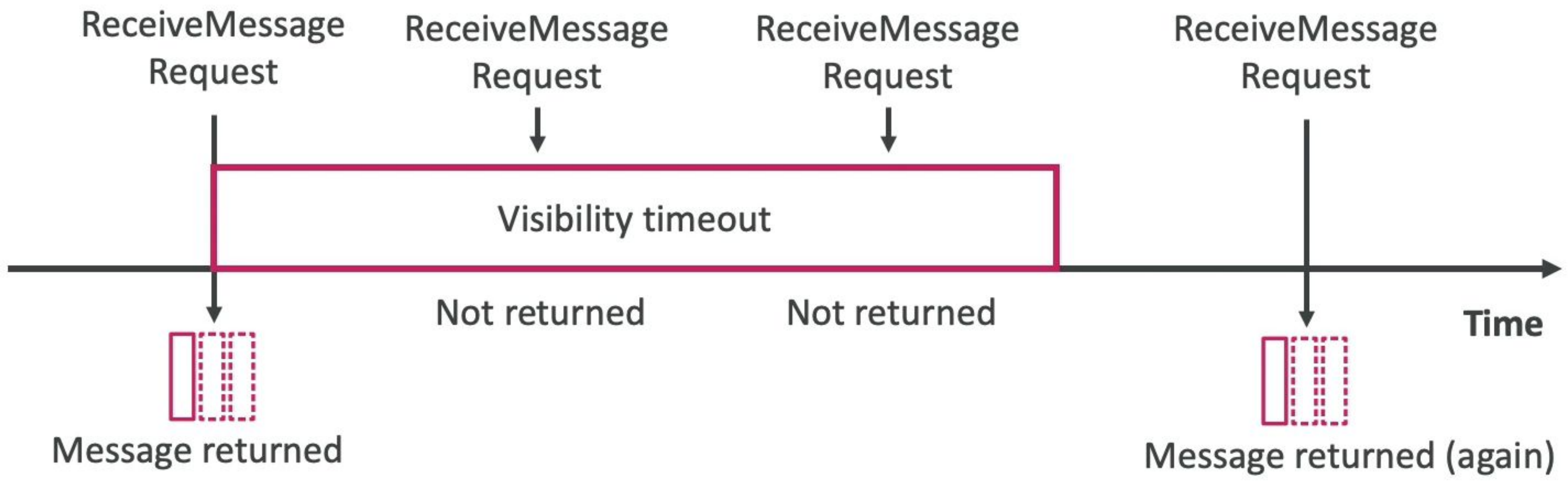
When a consumer receives and processes a message from a queue, the message remains in the queue. Amazon SQS doesn't automatically delete the message. Because Amazon SQS is a distributed system, there's no guarantee that the consumer actually receives the message (for example, due to a connectivity issue, or due to an issue in the consumer application). Thus, the consumer must delete the message from the queue after receiving and processing it.



Immediately after a message is received, it remains in the queue. To prevent other consumers from processing the message again, Amazon SQS sets a *visibility timeout*, a period of time during which Amazon SQS prevents other consumers from receiving and processing the message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours.

SQS

Message Visibility Timeout



SQS

Dead Letter Queue (DLQ)

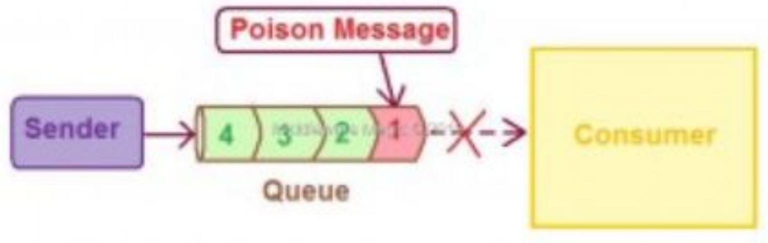
The dead-letter queue (or undelivered-message queue) is the queue to which messages are sent if they cannot be routed to their correct destination. Each queue manager typically has a dead-letter queue.

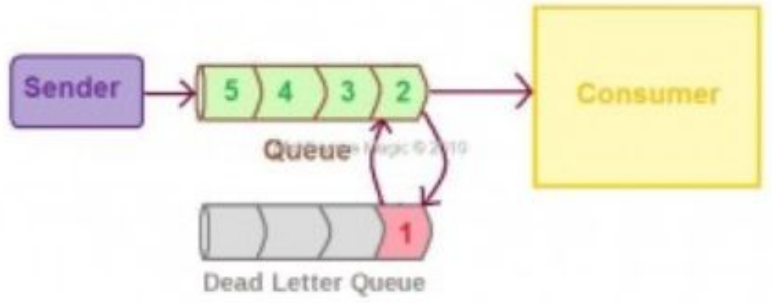
A *dead-letter queue* (DLQ), sometimes referred to as an *undelivered-message queue*, is a holding queue for messages that cannot be delivered to their destination queues, for example because the consumer failed to process it, or because it is full. Dead-letter queues are also used at the sending end of a channel, for data-conversion errors.

Any queue can be turned as a dead-letter queue so that messages that cannot be delivered to their correct destination can be stored for later retrieval.

SQS

Dead Letter Queue (DLQ)





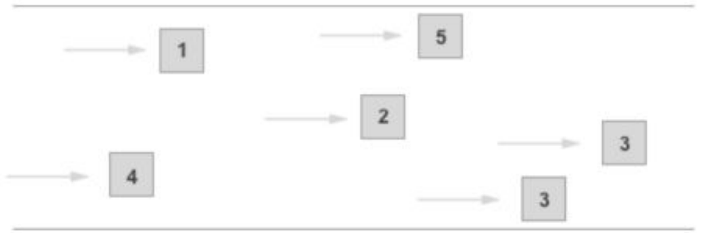
SQS

Standard Queue

High Throughput: Standard queues have nearly-unlimited transactions per second (TPS).

At-Least-Once Delivery: A message is delivered at least once, but occasionally more than one copy or a message is delivered.

Best-Effort Ordering: Occasionally, messages are delivered in an order different from which they were sent.



Send data between applications when the throughput is important, for example:

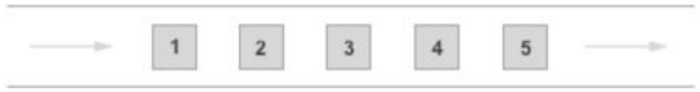
- Decouple live user requests from intensive background work: let users upload media while resizing or encoding it.
- Allocate tasks to multiple worker nodes: process a high number of credit card validation requests.
- Batch messages for future processing: schedule multiple entries to be added to a database.

FIFO Queue

First-In-First-out Delivery: The order in which messages are sent and received is strictly preserved.

Exactly-Once Processing: A message is guaranteed to be delivered at least once, but all duplicates of the message are removed.

Limited Throughput: 300 transactions per second (TPS).



Send data between applications when the order of events is important, for example:

- Ensure that user-entered commands are executed in the right order.
- Display the correct product price by sending price modifications in the right order.
- Prevent a student from enrolling in a course before registering for an account.

SQS

Pricing

- [Pay only for what you use]

- AWS **Free Tier** includes **1 million requests** with Amazon Simple Queue Service (SQS).

2

SNS

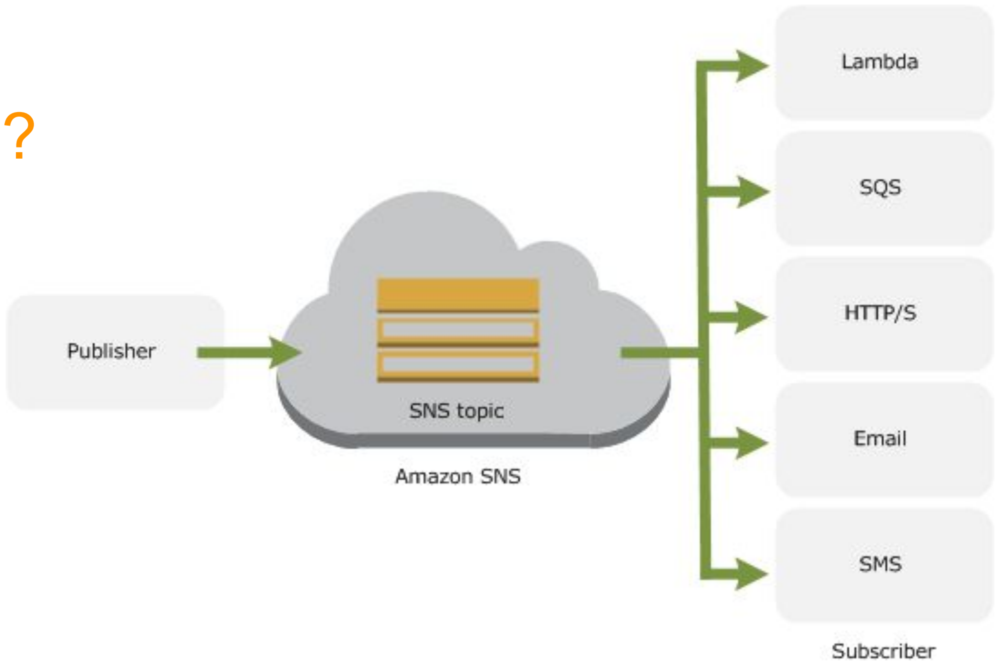
SNS

What is SNS?

- Amazon **Simple Notification Service** (Amazon SNS) is a managed service that provides message delivery from **publishers** to **subscribers** (also known as **producers** and **consumers**).

SNS

What is SNS?

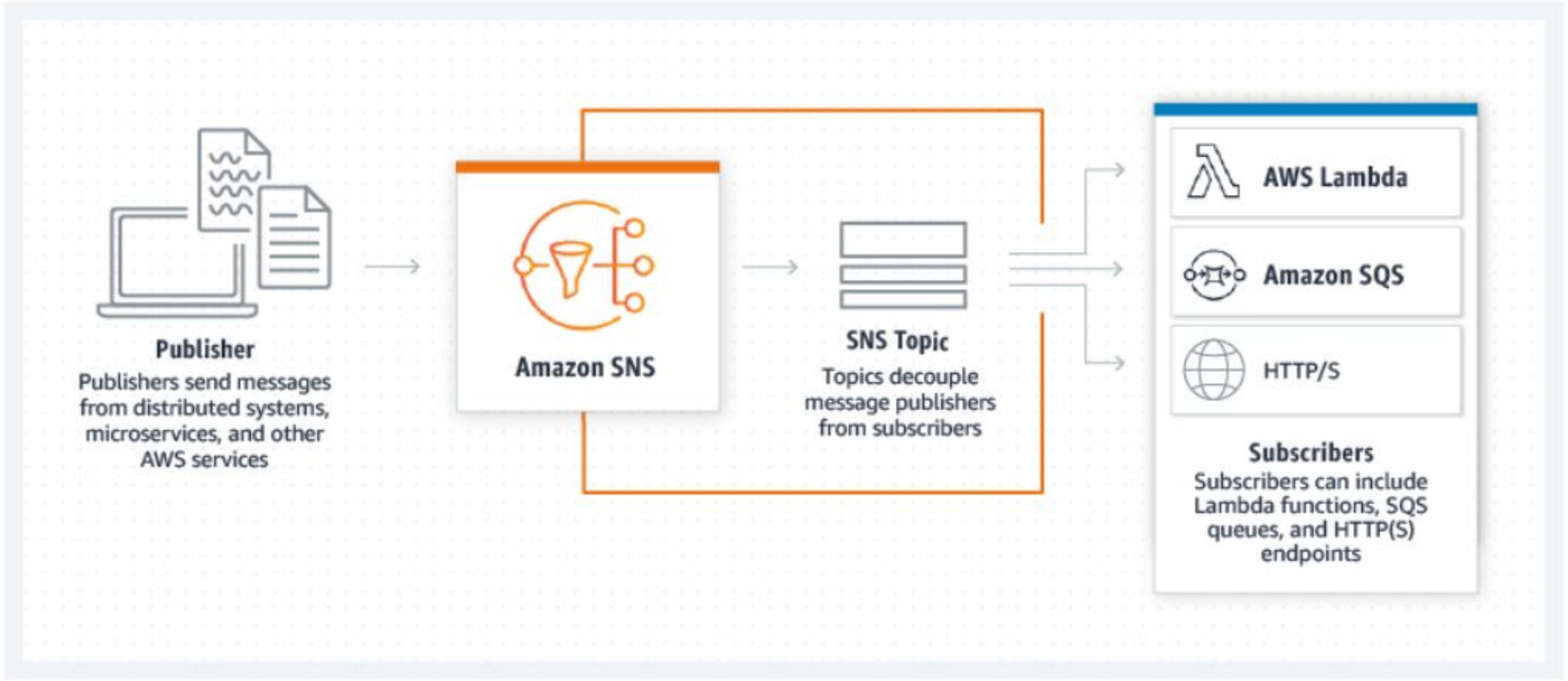


- Clients can subscribe to the **SNS topic** and receive published messages using a supported protocol, such as Amazon SQS, AWS Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

SNS

What is SNS?

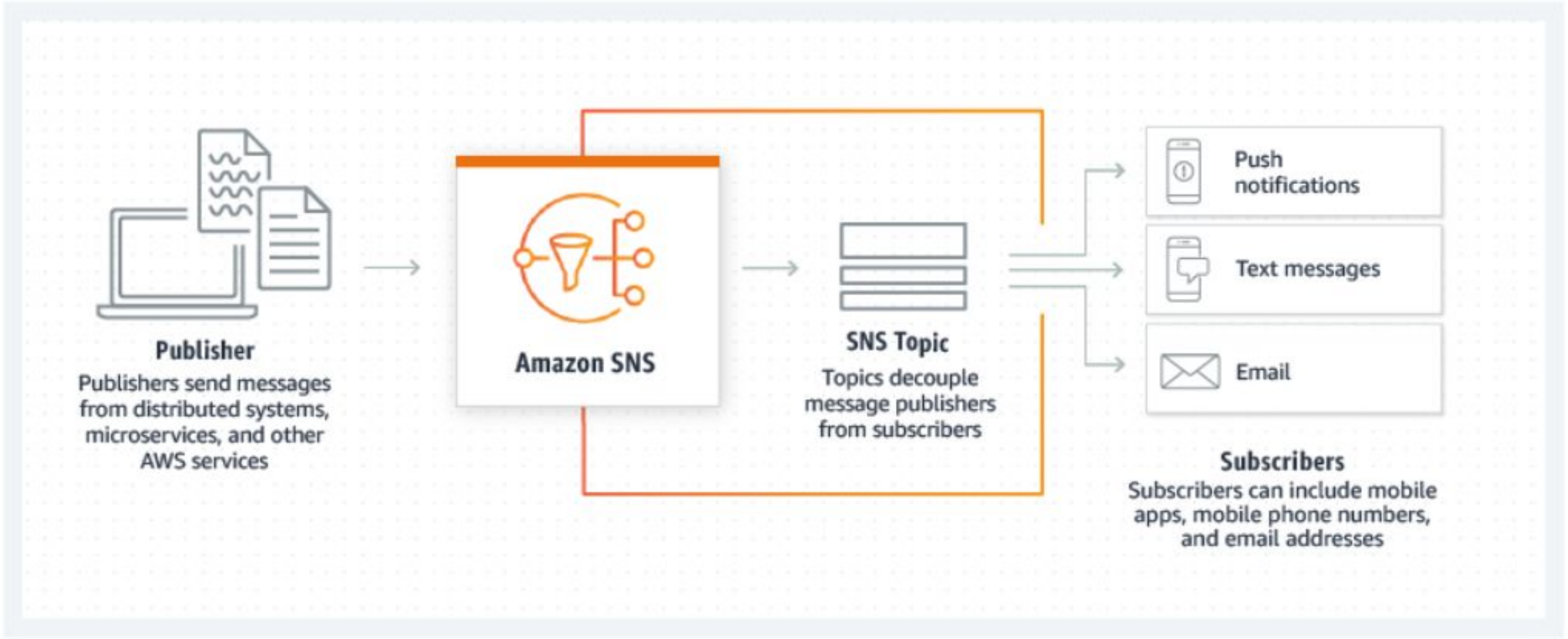
Application-to-Application (A2A)


Publisher
Publishers send messages from distributed systems, microservices, and other AWS services
Amazon SNS
SNS Topic
Topics decouple message publishers from subscribers
AWS Lambda
Amazon SQS
HTTP/S
Subscribers
Subscribers can include Lambda functions, SQS queues, and HTTP(S) endpoints

SNS

What is SNS?

Application-to-Person (A2P)



Publisher
Publishers send messages from distributed systems, microservices, and other AWS services
Amazon SNS
SNS Topic
Topics decouple message publishers from subscribers
Push notifications
Text messages
Email
Subscribers
Subscribers can include mobile apps, mobile phone numbers, and email addresses

CLARUSWAY
WAY TO REINVENT YOURSELF

SNS

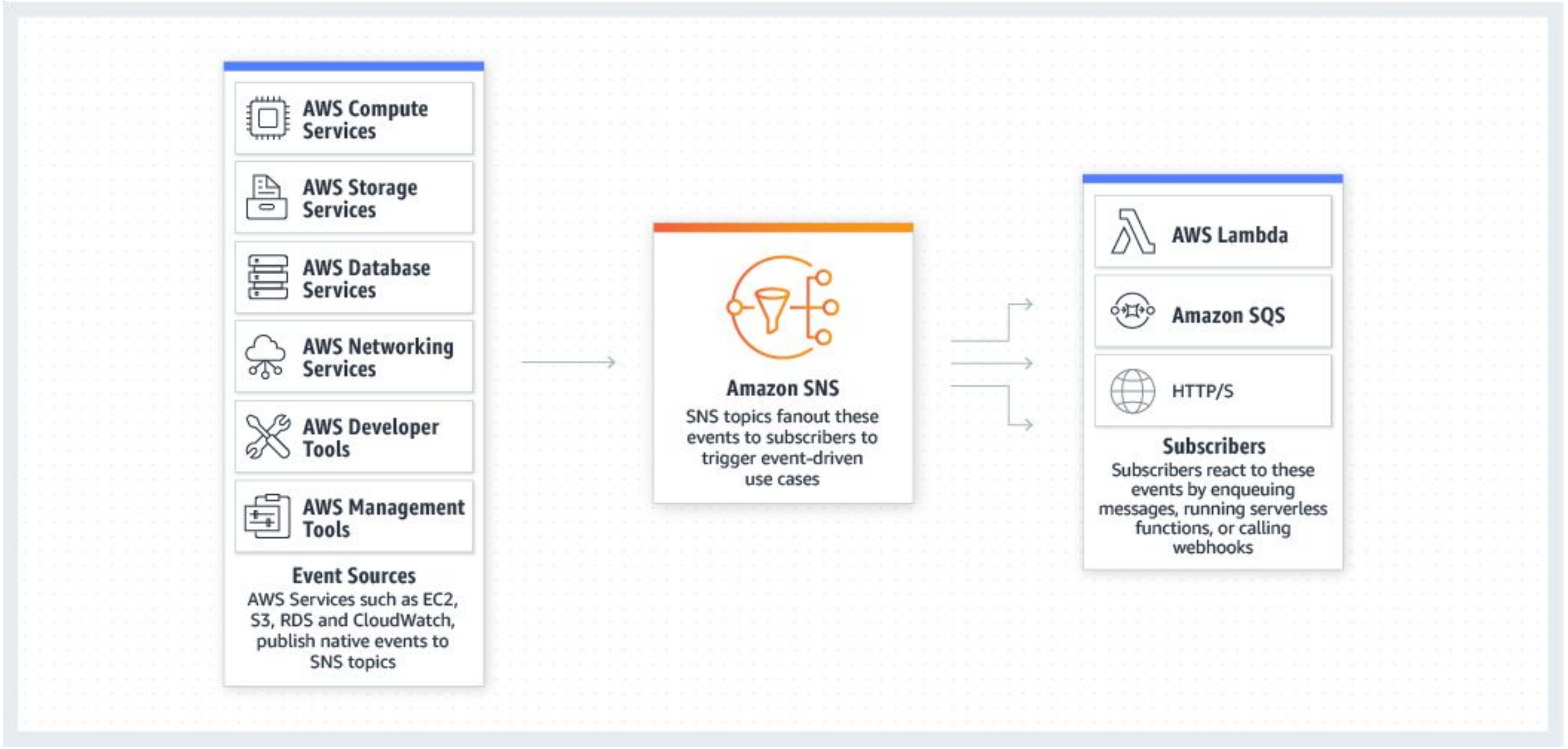
What is SNS?

SNS is integrated with many AWS Services

- CloudWatch (Alarms/Events)
- S3 (Bucket Events)
- CloudFormation (State changes etc.)
- Auto Scaling Groups
- And many others **can invoke SNS.**

SNS

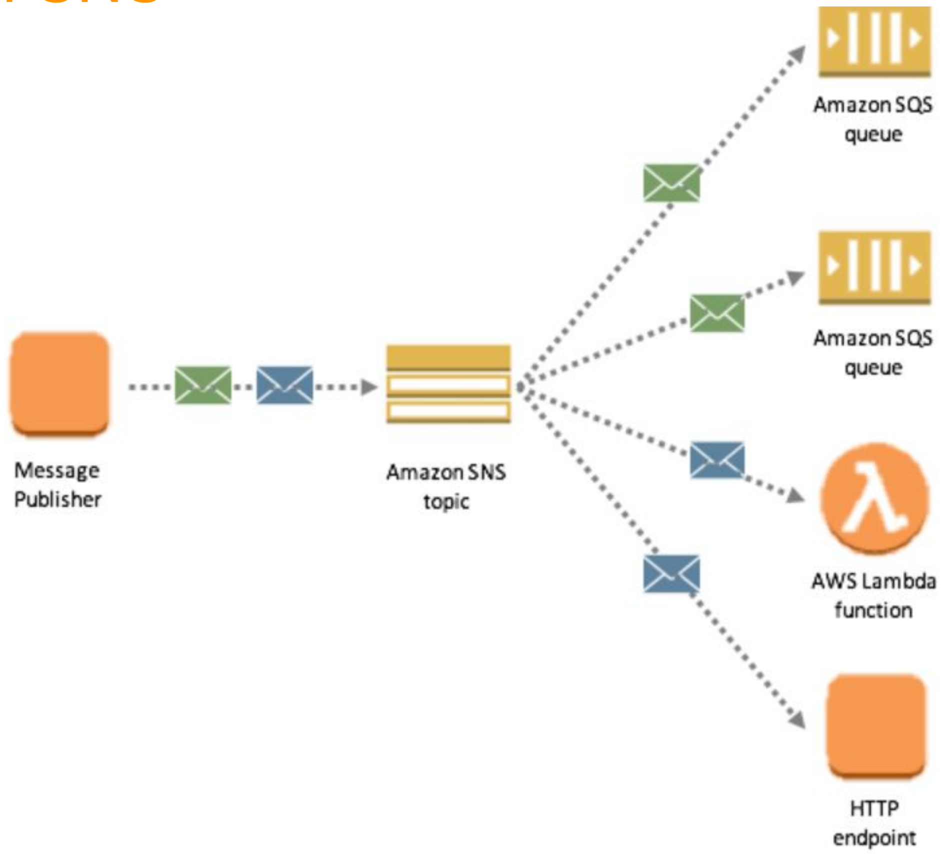
What is SNS?



AWS Compute Services
AWS Storage Services
AWS Database Services
AWS Networking Services
AWS Developer Tools
AWS Management Tools
Event Sources
AWS Services such as EC2, S3, RDS and CloudWatch, publish native events to SNS topics
Amazon SNS
SNS topics fanout these events to subscribers to trigger event-driven use cases
AWS Lambda
Amazon SQS
HTTP/S
Subscribers
Subscribers react to these events by enqueuing messages, running serverless functions, or calling webhooks

SNS

Fan Out with SNS



SNS

Pricing

- Amazon SNS has **no upfront costs** and you can **pay as you go**. You pay based on the number of notifications you publish, the number of notifications you deliver, and any additional API calls for managing topics and subscriptions. Delivery **pricing varies by endpoint type**. You can get started for free with the SNS free tier.

Key Differences between SNS & SQS

Entity Type
SQS : Queue (similar to JMS, MSMQ).
SNS : Topic-Subscriber (Pub/Sub system).

Message consumption
SQS : Pull Mechanism — Consumers poll messages from SQS.
SNS : Push Mechanism — SNS pushes messages to consumers.

Persistence
SQS : Messages are persisted for some duration if no consumer available. The retention period value is from 1 minute to 14 days. The default is 4 days.
SNS : No persistence. Whichever consumer is present at the time of message arrival, get the message and the message is deleted. If no consumers available then the message is lost.
In SQS the message delivery is guaranteed but in SNS it is not.

Consumer Type
SQS : All the consumers are supposed to be identical and hence process the messages in exact same way.
SNS : All the consumers are (supposed to be) processing the messages in different ways.

Use Cases

Choose SQS if:
You need a simple queue with no particular additional requirements.
Decoupling two applications and allowing parallel asynchronous processing. Only one subscriber is needed.

Choose SNS if:
You would like to be able to publish and consume batches of messages.
You would like to allow same message to be processed in multiple ways.
Multiple subscribers are needed.

SQS & SNS

Hands on

- Hands-on!!!
- Let’s go to the AWS Management Console

THANKS!

Any questions?

@sumod

sumod@clarusway.com