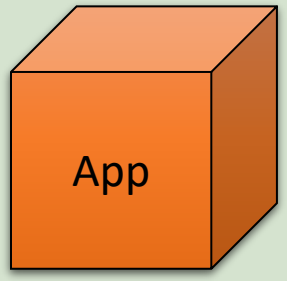
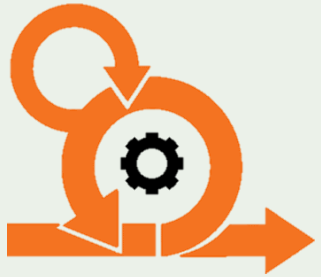
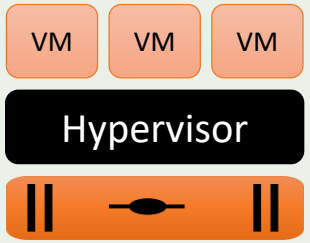
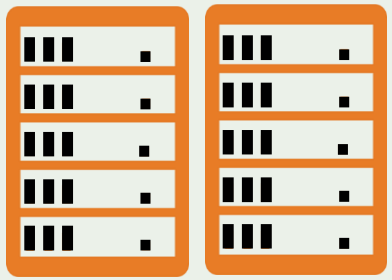
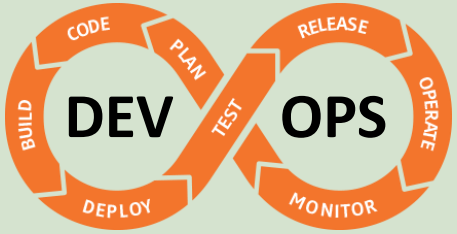
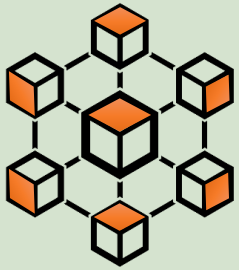
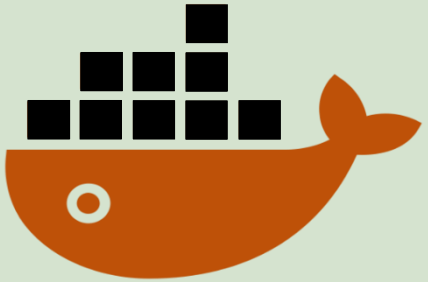
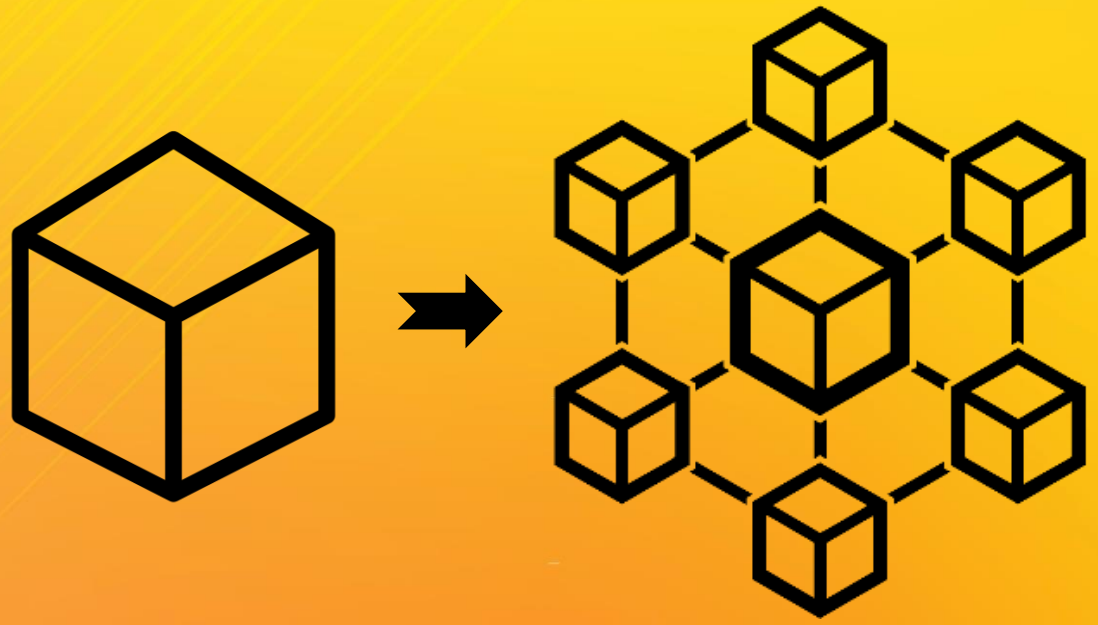
Decoupled Architecture

Agenda

- How applications evolved?
- Monolith vs. Microservices
- Decoupled Architecture
- Amazon SQS
- Amazon SNS

How applications evolved?

TimeLine	Development Process	Application Architecture	Deployment & Packaging	Application Hosting Infrastructure
1980 to 2000	 Waterfall	 Monolithic	 Physical Server	 Datacenter
2000 to 2010	 Agile	 N-Tier	 Virtual Servers	 Hosted
2010 to Current	 DevOps	 Microservices	 Containers	 Cloud



Monolith vs. Microservices

A food stall vs. a pizza place

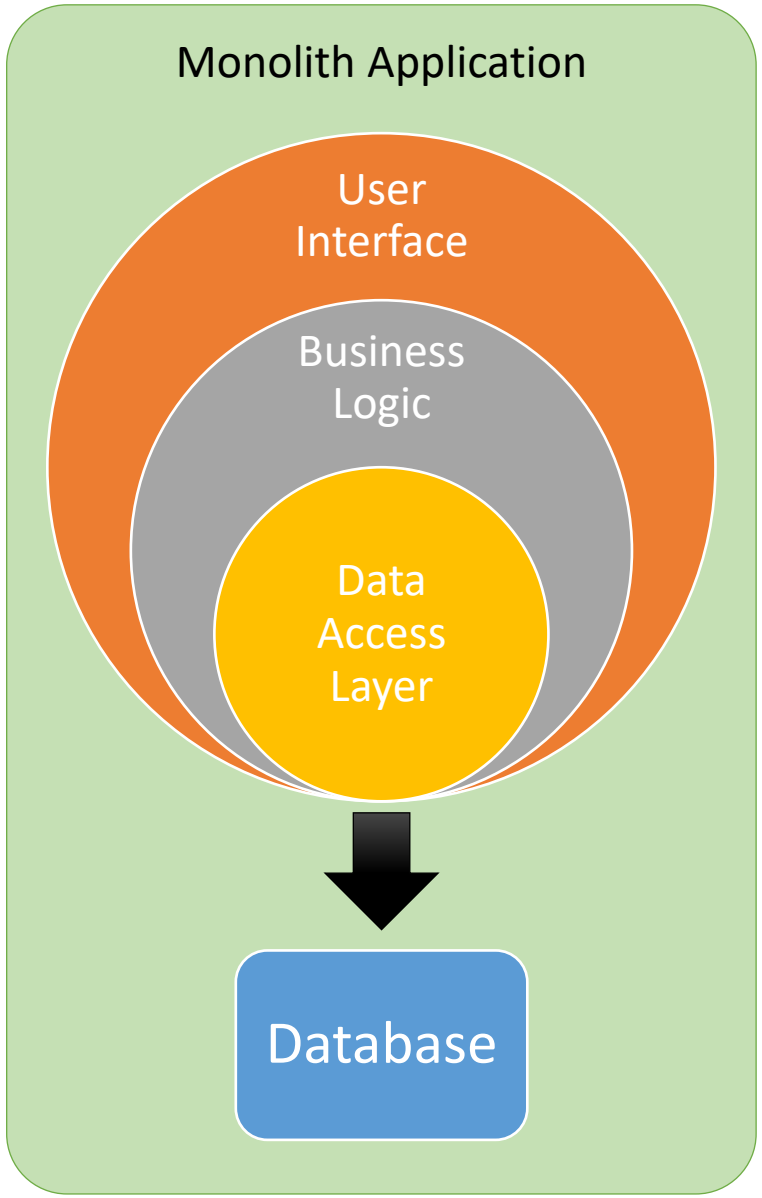
VS.

A food stall vs. a fine dining restaurant

VS.

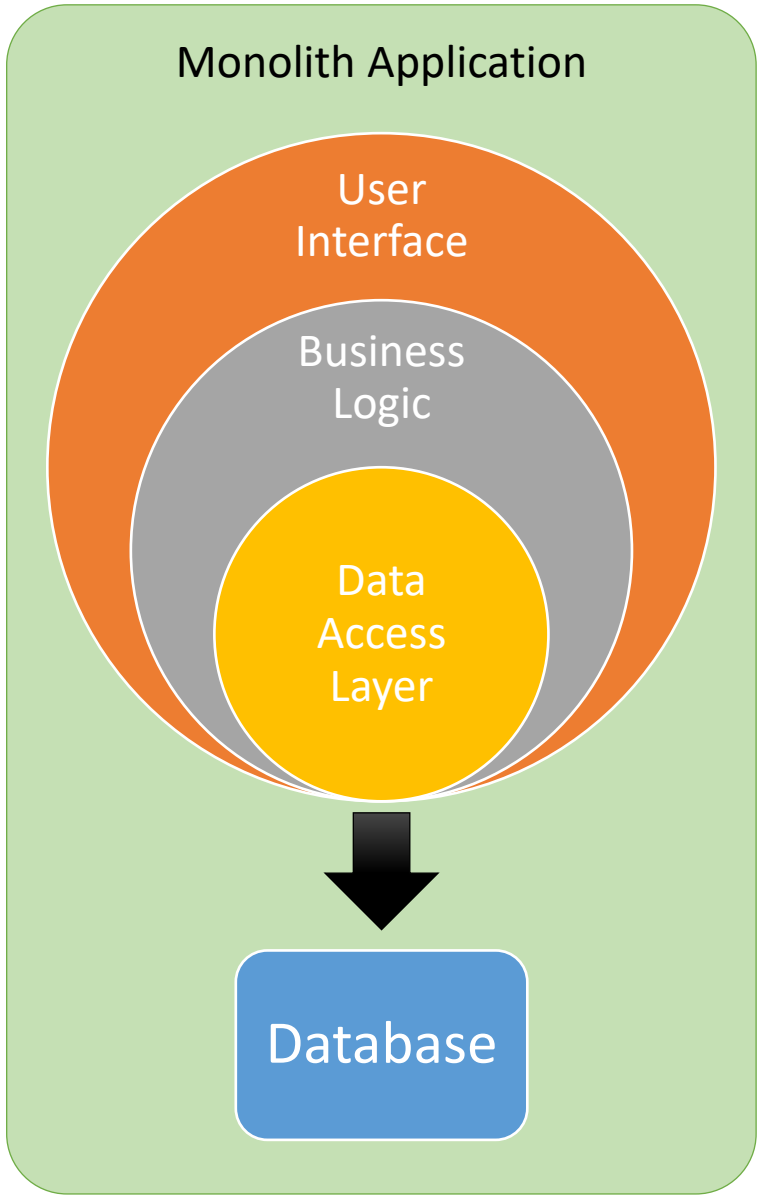
What is a Monolith Application?

- One large system and is usually one code-base
- Built as a single and indivisible unit
- Designed without modularity
- Deployed all at once
- Components depend on each other to work



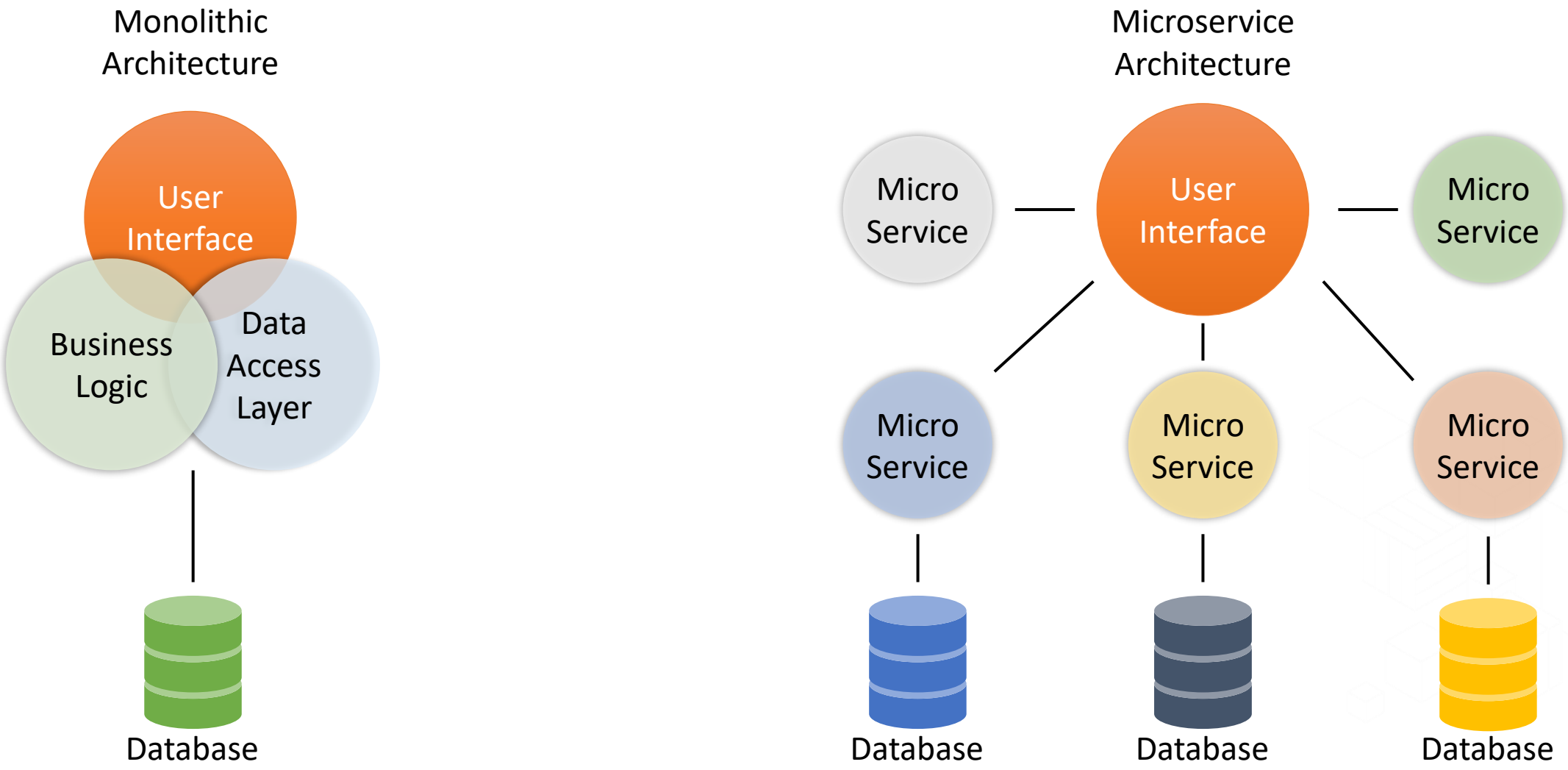
Monolith Challenges

- Barrier to adopting new technologies
- Continuous deployment is difficult
- Complex / large code base
- Difficult to scale
- Difficult to maintain

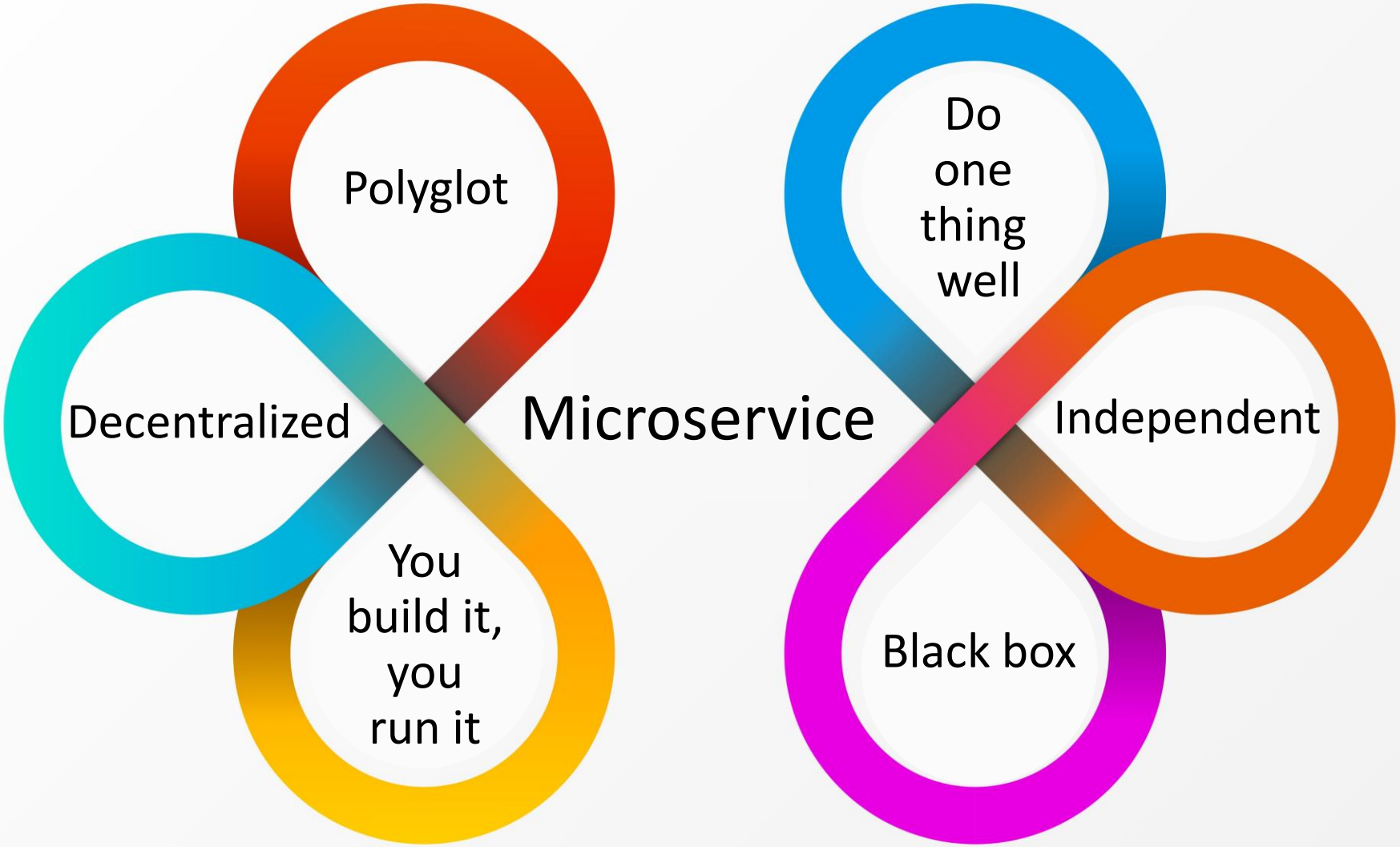


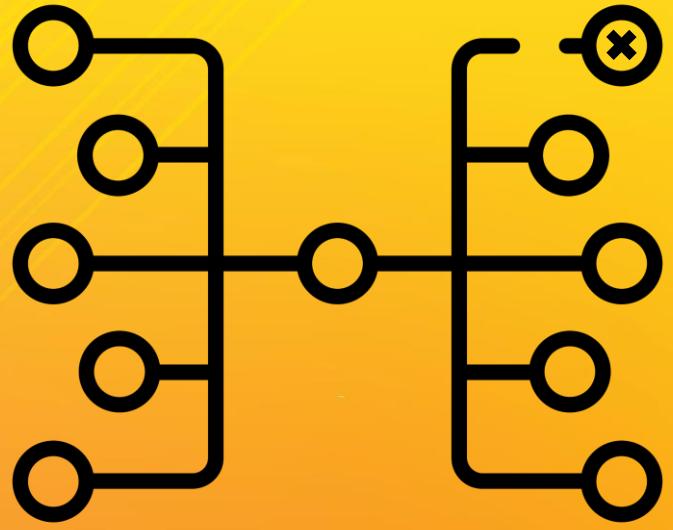
What Are Microservices?

- Applications composed of independent services that communicate over well-defined APIs



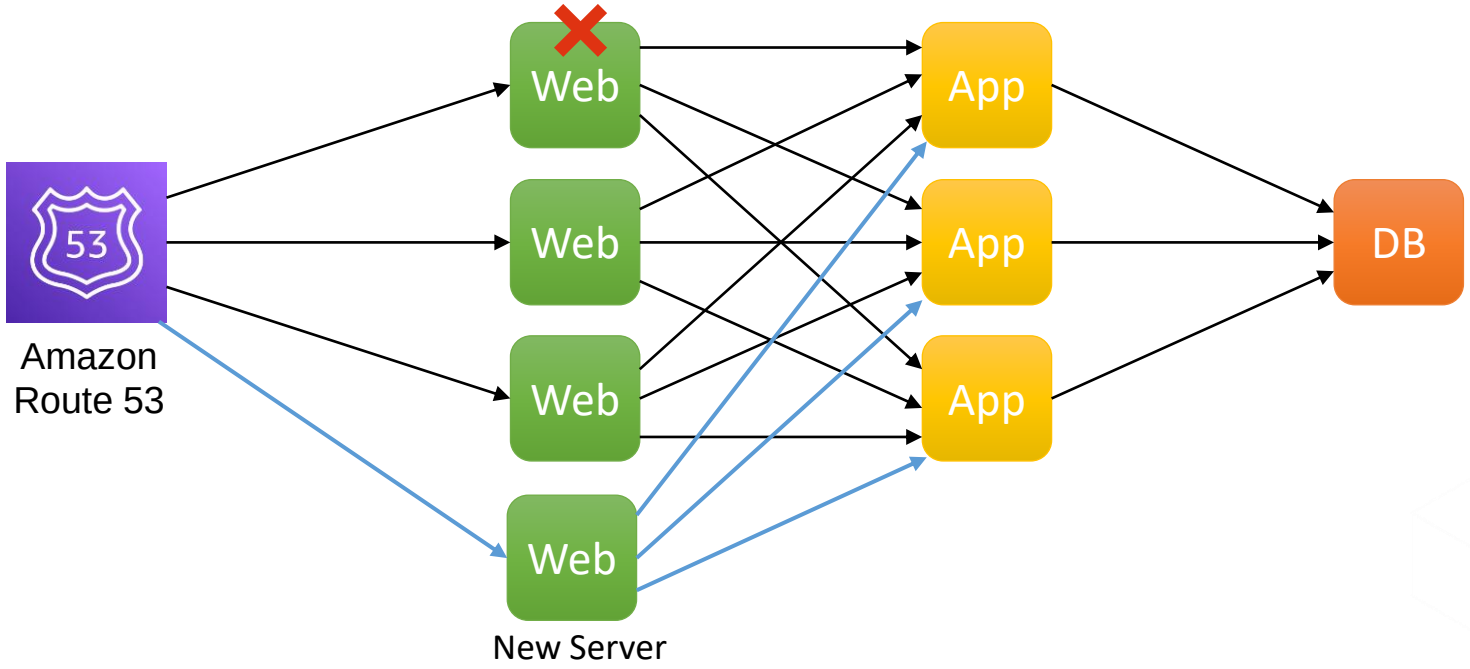
Characteristics of a Microservice



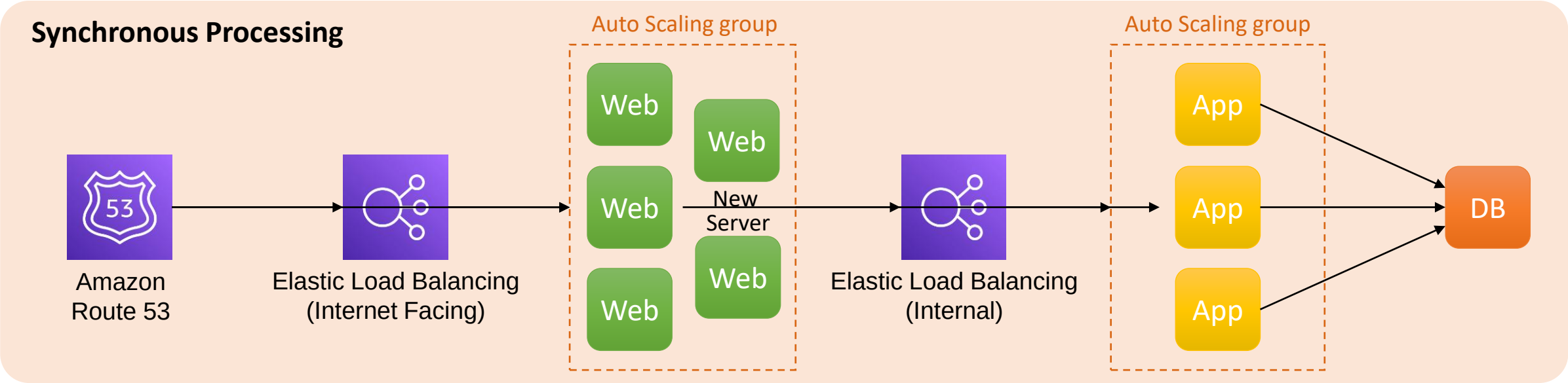


Decoupled Architecture on AWS

Tightly Coupled Architectures



Decoupled Architectures



Synchronous vs. Asynchronous Processing

Synchronous
Sender Receiver both should be present

Asynchronous
Receiver need not to be present

Letter Box

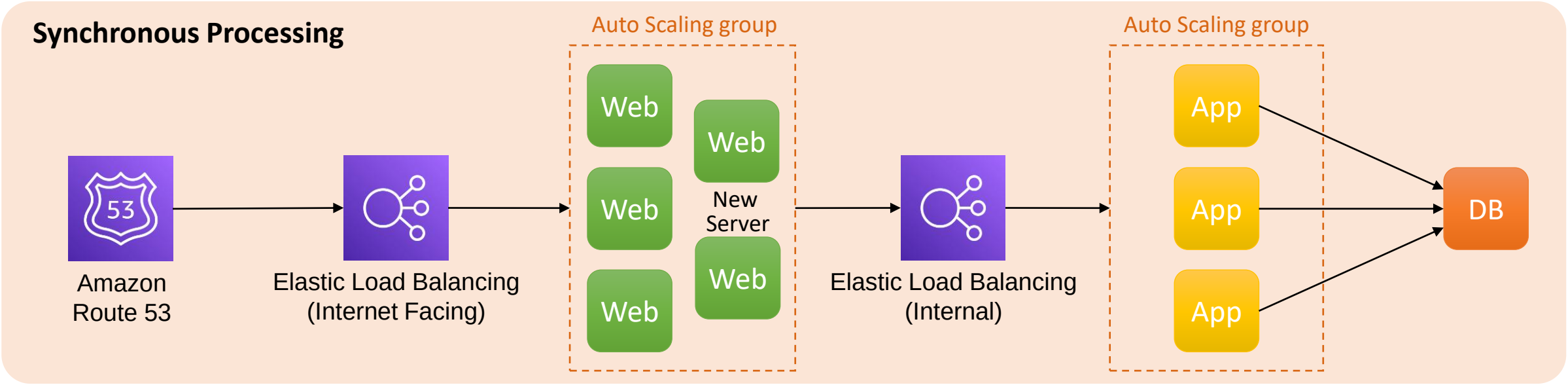
- Decoupled Architecture

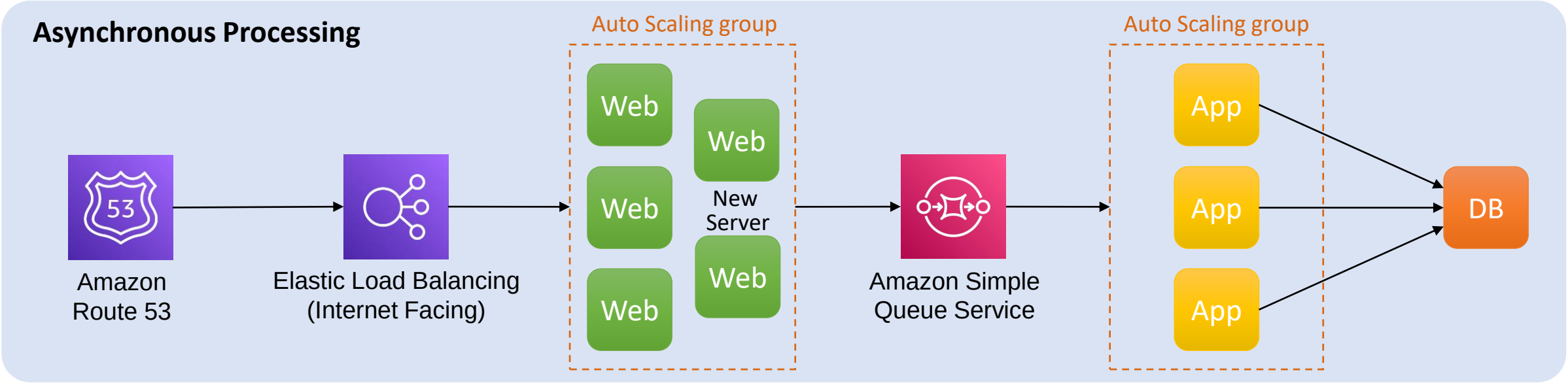
1. You install letter Box

2. Postman drops letters

3. You pick up the letters

Decoupled Architectures



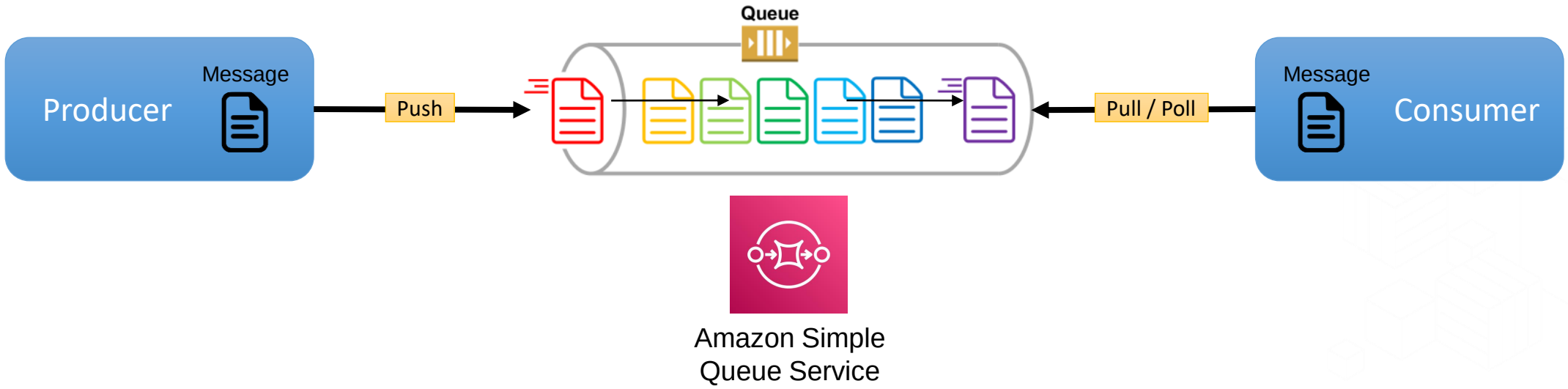


Amazon Simple Queue Service (SQS)

Queue at a grocery store

Amazon SQS

- An Amazon SQS queue is a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components.
- A SQS queue can play the role of the buffer to decouple producers and consumers.
- It provides a generic web services API that you can access using any programming language that the AWS SDK supports.



Terminology

- Message Producer
  - An application, process, or service that sends messages to the queue
- Message Receiver
  - An application, process, or service that receives (polls) messages from the queue, processes them, and deletes them from the queue
- Access policy
  - Defines the accounts, users and roles that can access this queue, and the actions that are allowed.
- Dead Letter Queue
  - If a message can't be consumed successfully, you can send it to a dead-letter queue (DLQ). Dead-letter queues let you isolate problematic messages to determine why they are failing.

Terminology

- Polling
  - Long Polling
    - The *ReceiveMessage* request queries all of the servers for messages.
  - Short Polling
    - The *ReceiveMessage* request queries only a subset of the servers
- Visibility Timeout
  - Visibility timeout sets the length of time that a message received from a queue (by one consumer) will not be visible to the other message consumers.

Terminology

- Message Producer
  - An application, process, or service that sends messages to the queue
- Message Receiver
  - An application, process, or service that receives (polls) messages from the queue, processes them, and deletes them from the queue
- Access policy
  - Defines the accounts, users and roles that can access this queue, and the actions that are allowed.
- Dead Letter Queue
  - If a message can't be consumed successfully, you can send it to a dead-letter queue (DLQ). Dead-letter queues let you isolate problematic messages to determine why they are failing.

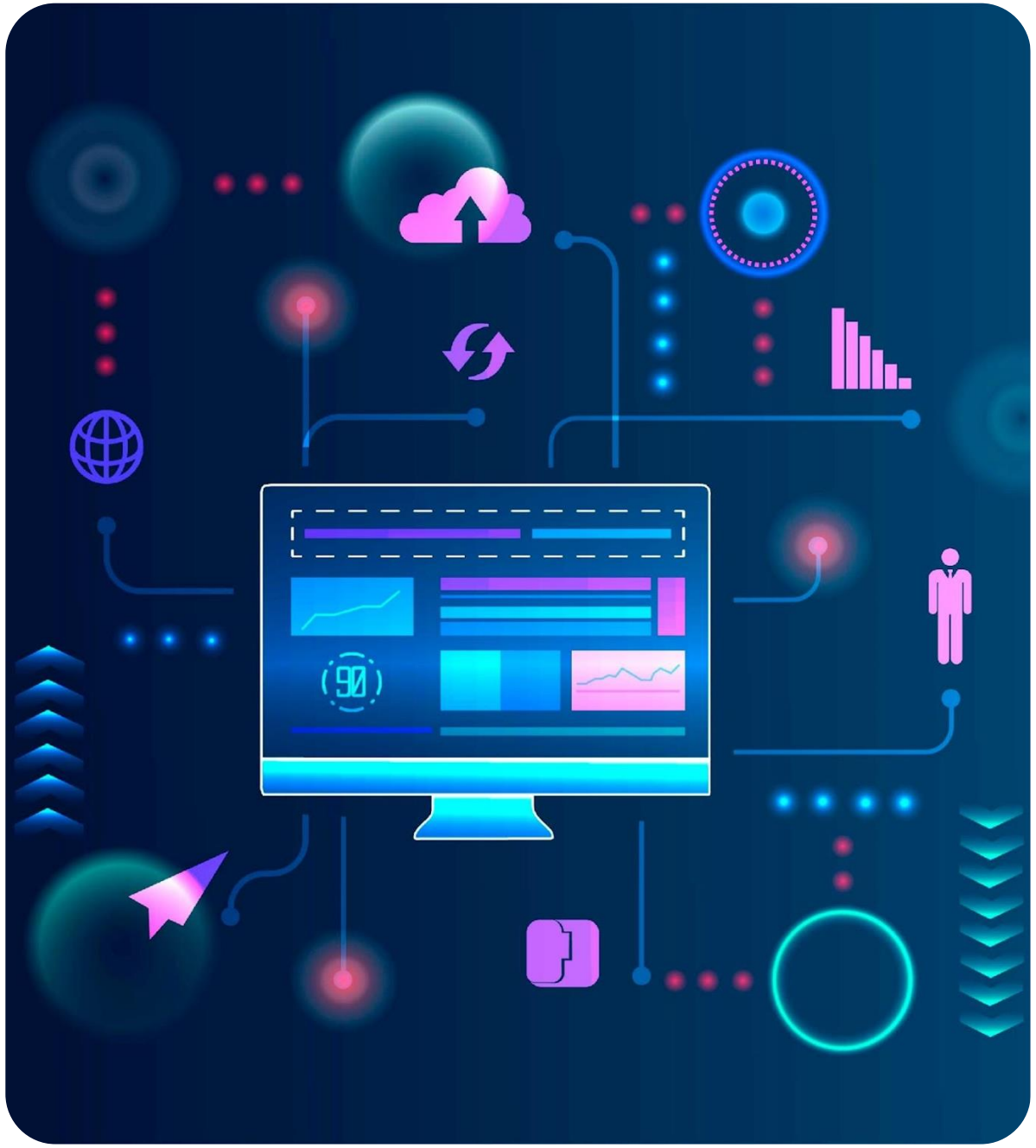
Terminology

- Polling
  - Long Polling
    - The *ReceiveMessage* request queries all of the servers for messages.
  - Short Polling
    - The *ReceiveMessage* request queries only a subset of the servers
- Visibility Timeout
  - Visibility timeout sets the length of time that a message received from a queue (by one consumer) will not be visible to the other message consumers.

Royal Mail - We missed you card

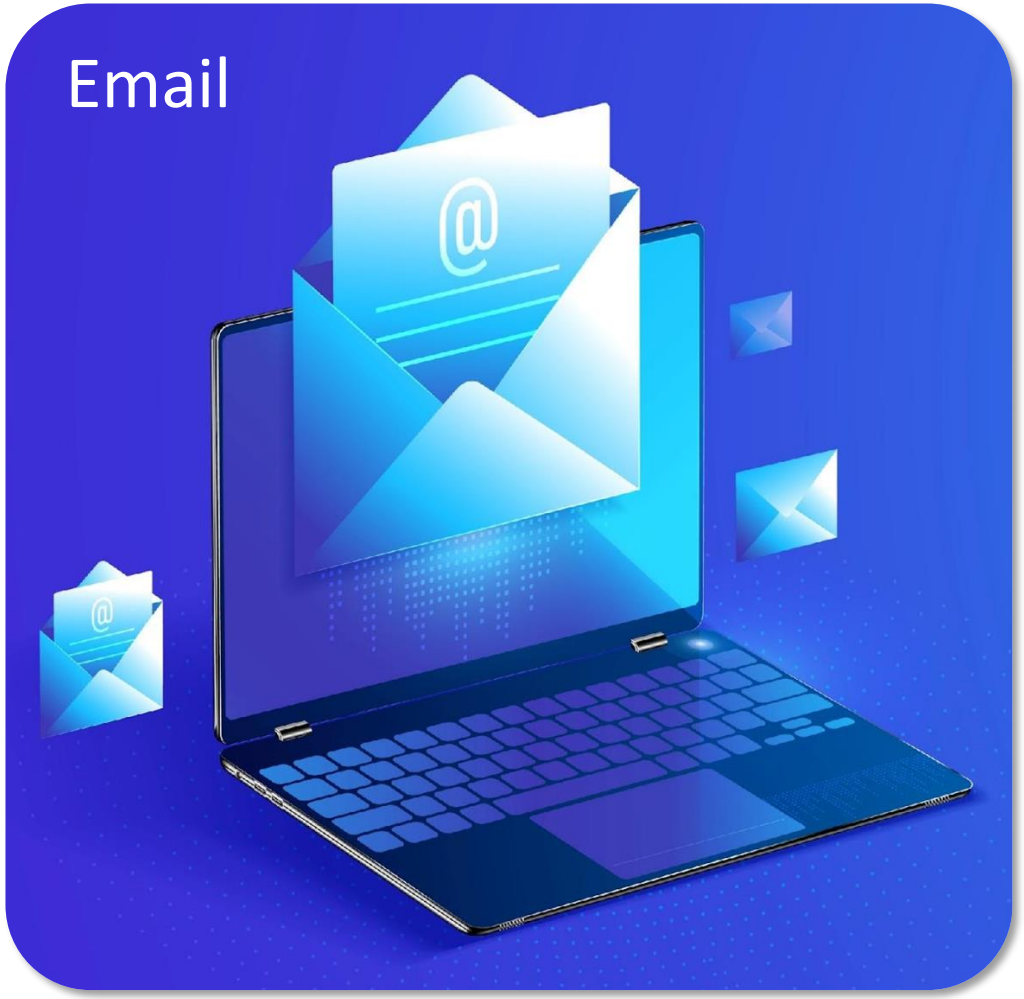
Standard Queue vs. FIFO Queue

Standard Queue	FIFO Queue
High Throughput: Standard queues have nearly-unlimited transactions per second (TPS).	**Limited Throughput:** 300 transactions per second (TPS).
At-Least-Once Delivery: A message is delivered at least once, but occasionally more that one copy of a message is delivered.	**Exactly-Once Processing:** A message is delivered once and remains available until a consumer processes and deletes it. Duplicates are not introduced into the queue.
Best-Effort Ordering: Occasionally, message might be delivered in an order different from which they were sent.	**First-In-First-Out Delivery:** The order in which messages are sent and received is strictly preserved.
	



Amazon Simple Notification Service (Amazon SNS)

Notifications



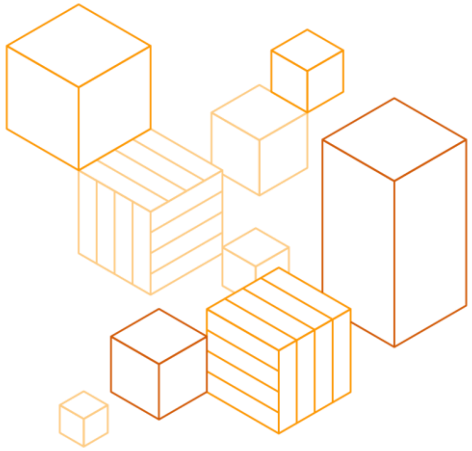
Notifications Types

- Application-to-Application (A2A)
  - Messaging between distributed systems, microservices, and event-driven serverless applications.
- Application-to-Person (A2P)
  - Messaging your customers with SMS texts, push notifications, and email.
- Use cases
  - Application and system alerts
  - Push email and text messaging
  - Mobile push notifications

Amazon Simple Notification Service (SNS)

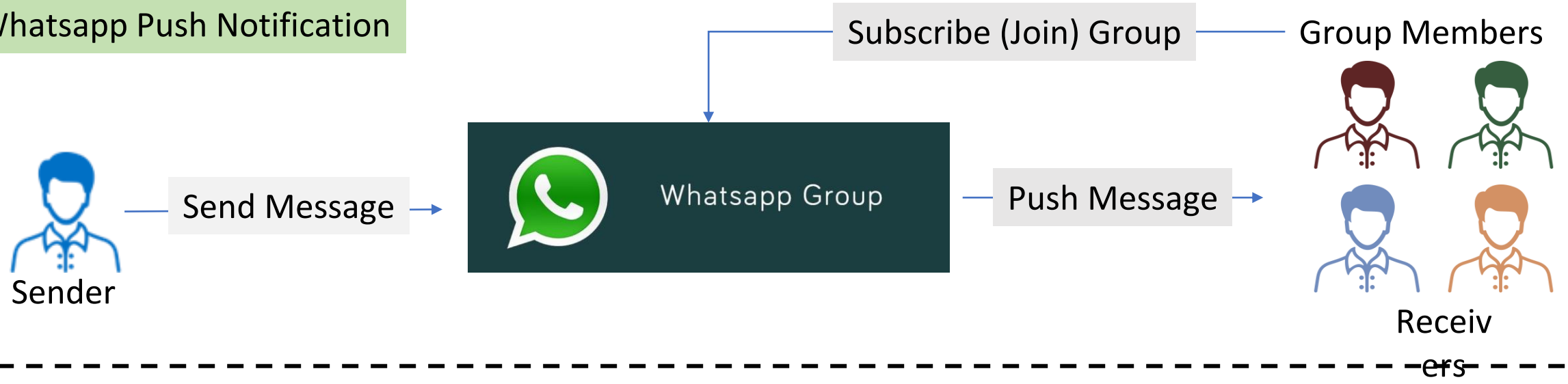
- Amazon SNS makes it easy to set up, operate, and send notifications from the cloud.

- It follows the “publish-subscribe” (pub-sub) messaging paradigm, with notifications being delivered to clients using a “push” mechanism

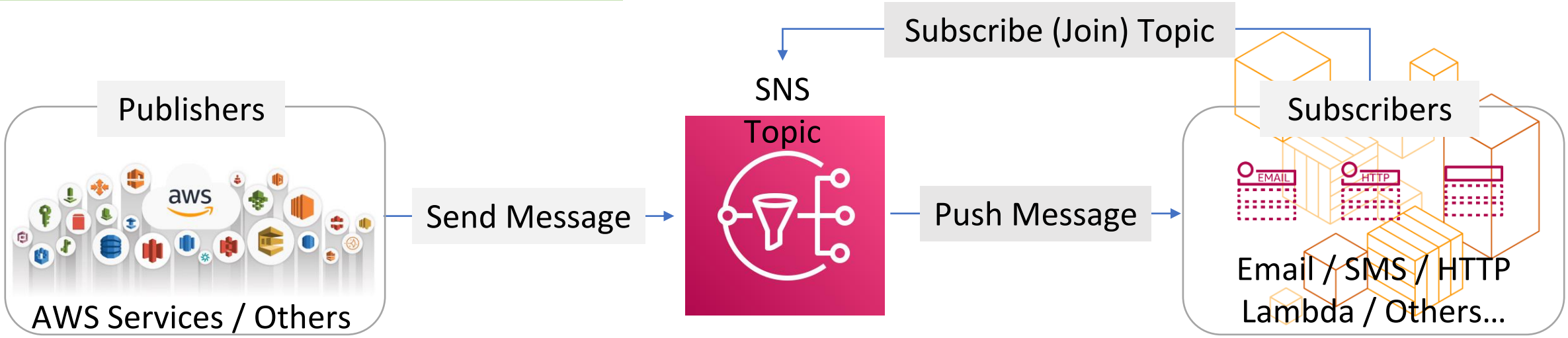


Amazon SNS Concepts

Whatsapp Push Notification



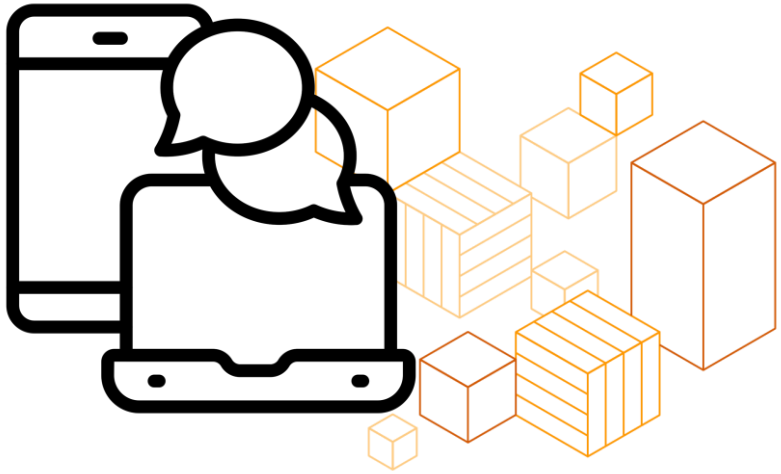
Amazon Simple Notification Service (SNS)



Amazon Simple Notification Service (SNS)

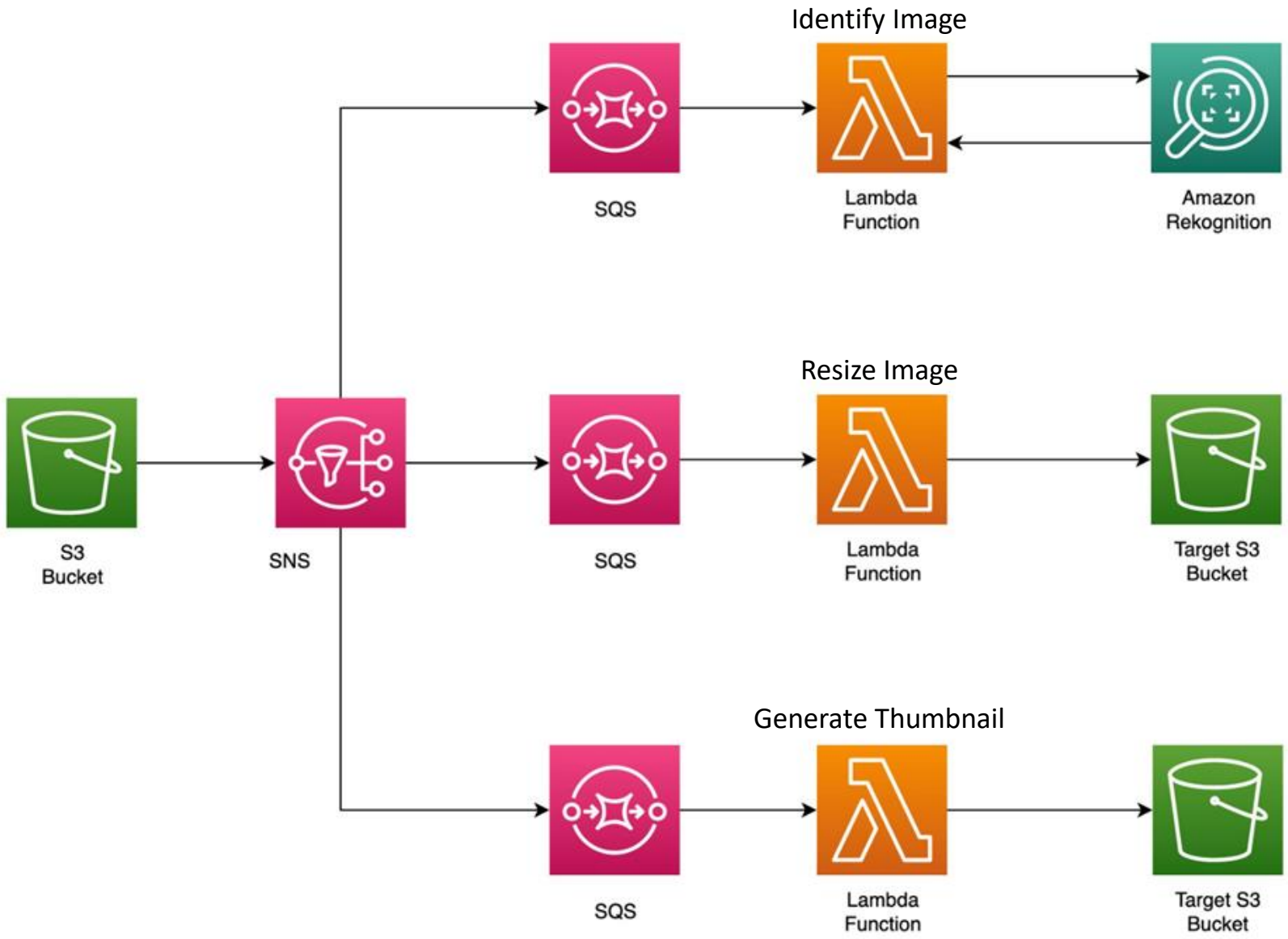
- Publish/subscribe messaging, or pub/sub messaging, is a form of asynchronous service-to-service communication used in serverless and microservices architectures.

- Pub/sub messaging can be used to enable event-driven architectures, or to decouple applications in order to increase performance, reliability and scalability.

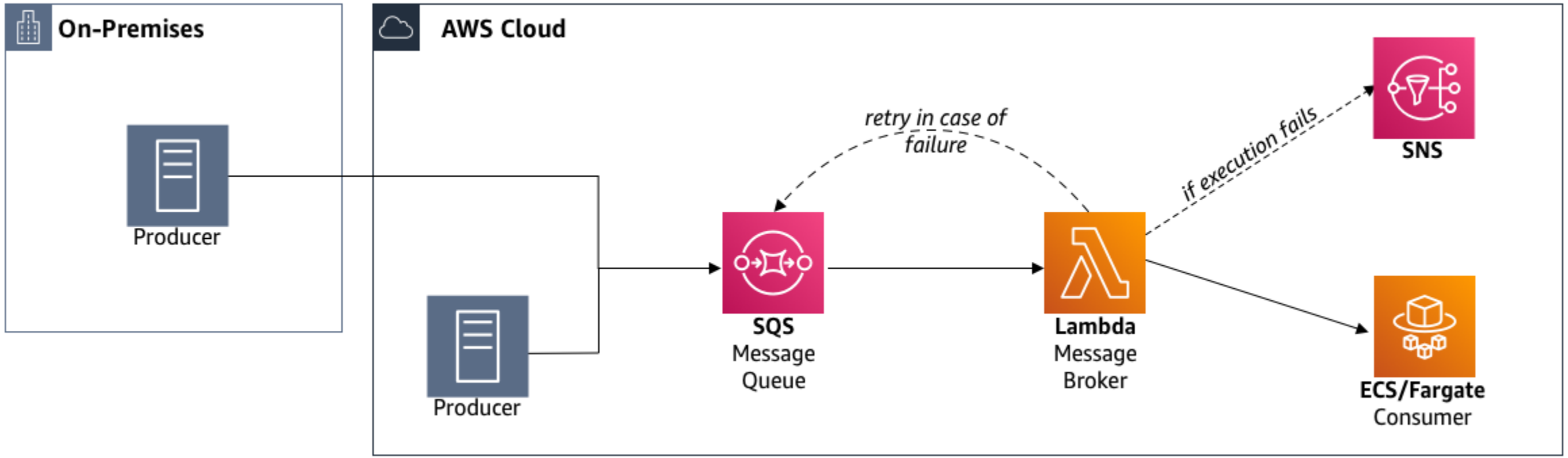


Architecture Patterns

Parallel processing with “Fan Out” architecture

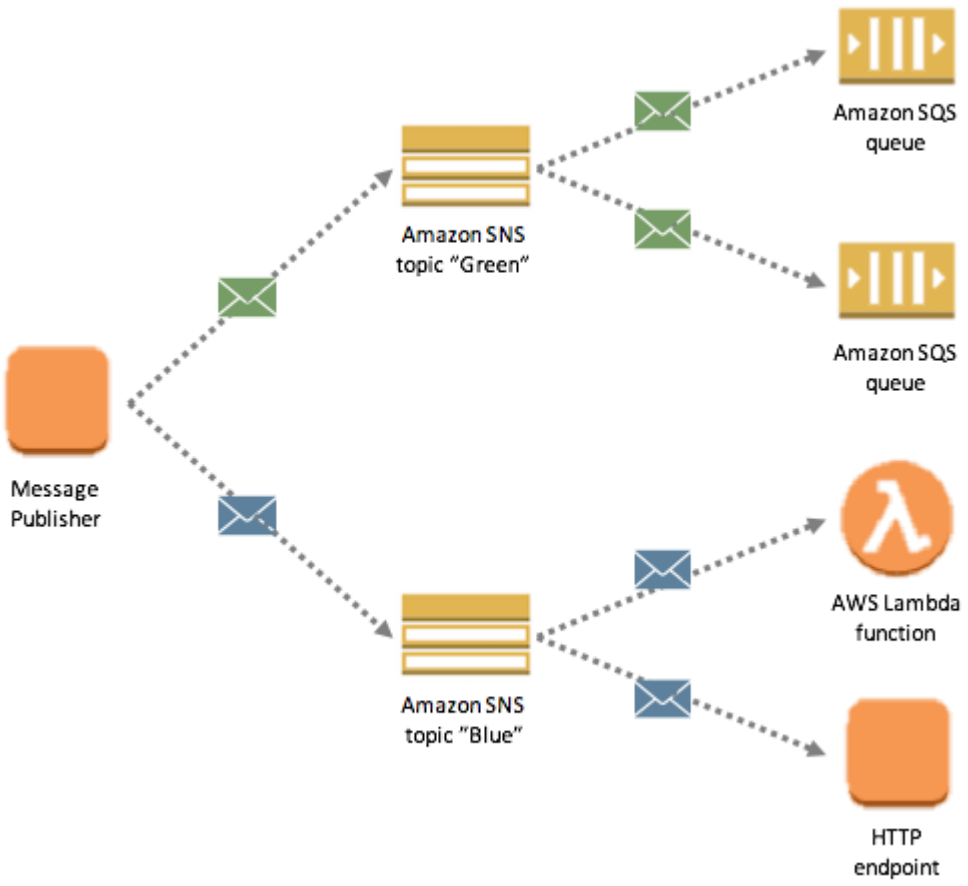


Queue Integration with Third-party Services on AWS

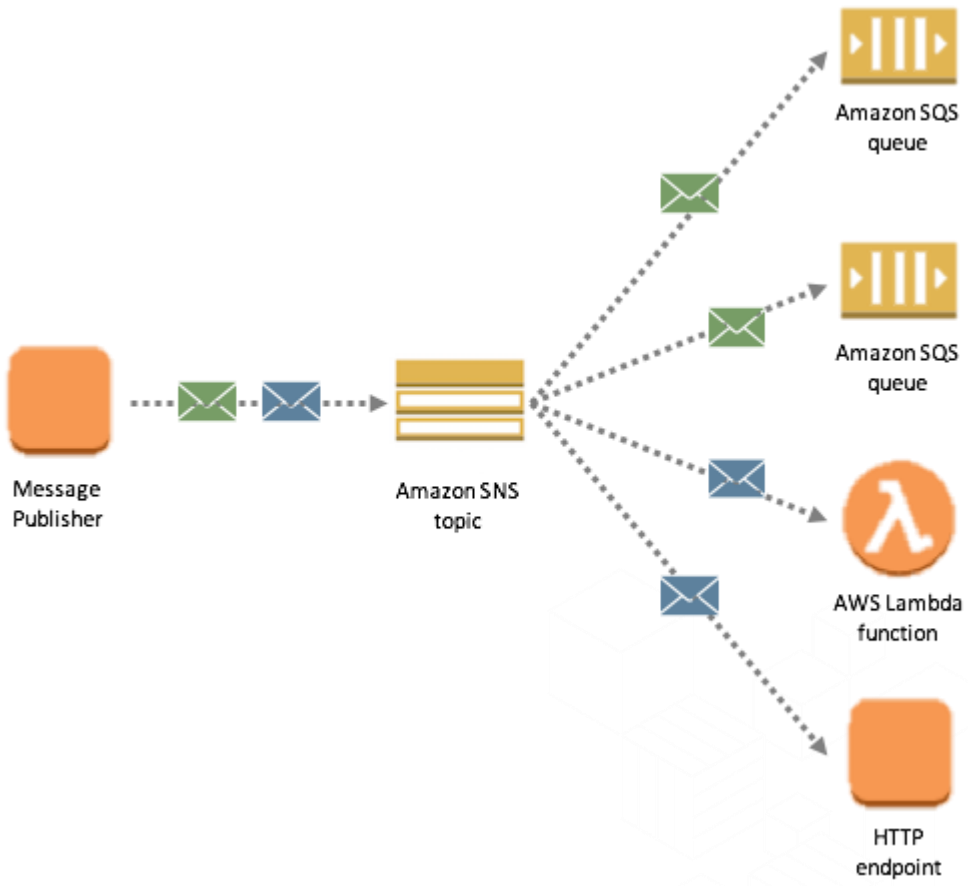


Simplify Your Pub/Sub Messaging with Amazon SNS Message Filtering

- Topic based filtering



- Attribute-based filtering



Amazon SNS vs. Amazon SQS

	Amazon SNS	Amazon SQS
Message persistence	No	Yes
Delivery mechanism	Push (passive)	Poll (active)
Producer and consumer	Publisher and Subscriber	Send or receive
Distribution model	One to many (1:N)	One to one (1:1)
Most common use cases	Application to Application (A2A) Application to Person (A2P)	Application to Application (A2A)

Reference:

FAQs

Category:

Application Integration

Amazon Simple Notification Service (Amazon SNS)

Complete book:

Click Here

Created by:

Ashish Prajapati

What?

- Amazon Simple Notification Service (Amazon SNS) is a fully managed push-based messaging service for both application-to-application (A2A) and application-to-person (A2P) communication.
- It provides message delivery from publishers to subscribers (also known as *producers* and *consumers*).

Why?

- Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications.
- Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

When?

- You can use Amazon SNS to support a wide variety of needs including event notification, monitoring applications, workflow systems, time-sensitive information updates, mobile applications, and any other application that generates or consumes notifications.

Where?

- Amazon SNS is a regional service.

Who?

- With simple APIs requiring minimal up-front development effort, no maintenance or management, Amazon SNS gives developers an easy mechanism to incorporate a powerful notification system with their applications.

How?

- Developers must first create a “topic” which is an “access point” – identifying a specific subject or event type – for publishing messages and allowing clients to subscribe for notifications. Topic owner can set policies for it such as limiting who can publish messages or subscribe, or specifying which protocols will be supported (i.e. HTTP/HTTPS, email, SMS).

How much?

- Standard topic - number of monthly API requests made, and the number of deliveries to various endpoints.
- FIFO topic - pricing is based on the number of published messages, the number of subscribed messages, and their respective amount of payload data.

Reference:

FAQs

Category:

Application Integration

Amazon Simple Queue Service (Amazon SQS)

Complete book:

Click Here

Created by:

Ashish Prajapati

What?

- 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.

Why?

- SQS eliminates the complexity and overhead associated with managing and operating message-oriented middleware, and empowers developers to focus on differentiating work.
- SQS scales elastically with your application so you don't have to worry about capacity planning and pre-provisioning.

When?

- You want to send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available. Use SQS standard queues for maximum throughput, best-effort ordering, and at-least-once delivery. Use SQS FIFO queues to guarantee that messages are processed exactly once, in the exact order.

Where?

- Amazon SQS is a regional service. Amazon SQS stores all message queues and messages within a single, highly-available AWS region with multiple redundant Availability Zones (AZs), so that no single computer, network, or AZ failure can make messages inaccessible.

Who?

- AWS manages the backend for Amazon SQS service including scaling and durability.
- Customers can control who can send messages to a message queue and who can receive messages from a message queue. Amazon SQS has its own resource-based permissions system.

How?

- Messages are sent from producers (applications, microservices, and other AWS services) to SQS Queue.
- It stores messages and wait for consumer to poll. Consumer applications (Lambda Functions, EC2 Instances and other AWS services) pull/poll the messages and process it.

How much?

- The cost of Amazon SQS is calculated per request, plus data transfer charges for data transferred out of Amazon SQS (unless data is transferred to Amazon EC2 instances or to AWS Lambda functions within the same region).
- Each 64 KB chunk of a payload is billed as 1 request (for example, an API action with a 256 KB payload is billed as 4 requests).