

# AWS Integration & Messaging

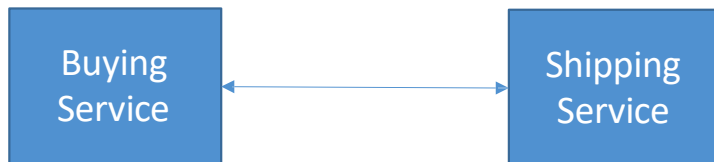
SQS, SNS & Kinesis

A decorative graphic element at the bottom of the slide, consisting of a thick orange bar that tapers to the right, with a thin grey bar layered on top of it.

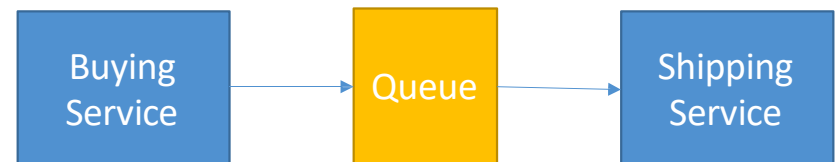
# Section Introduction

- When we start deploying multiple applications, they will inevitably need to communicate with one another
- There are two patterns of application communication


**1) Synchronous communications  
(application to application)**



**2) Asynchronous / Event based  
(application to queue to application)**

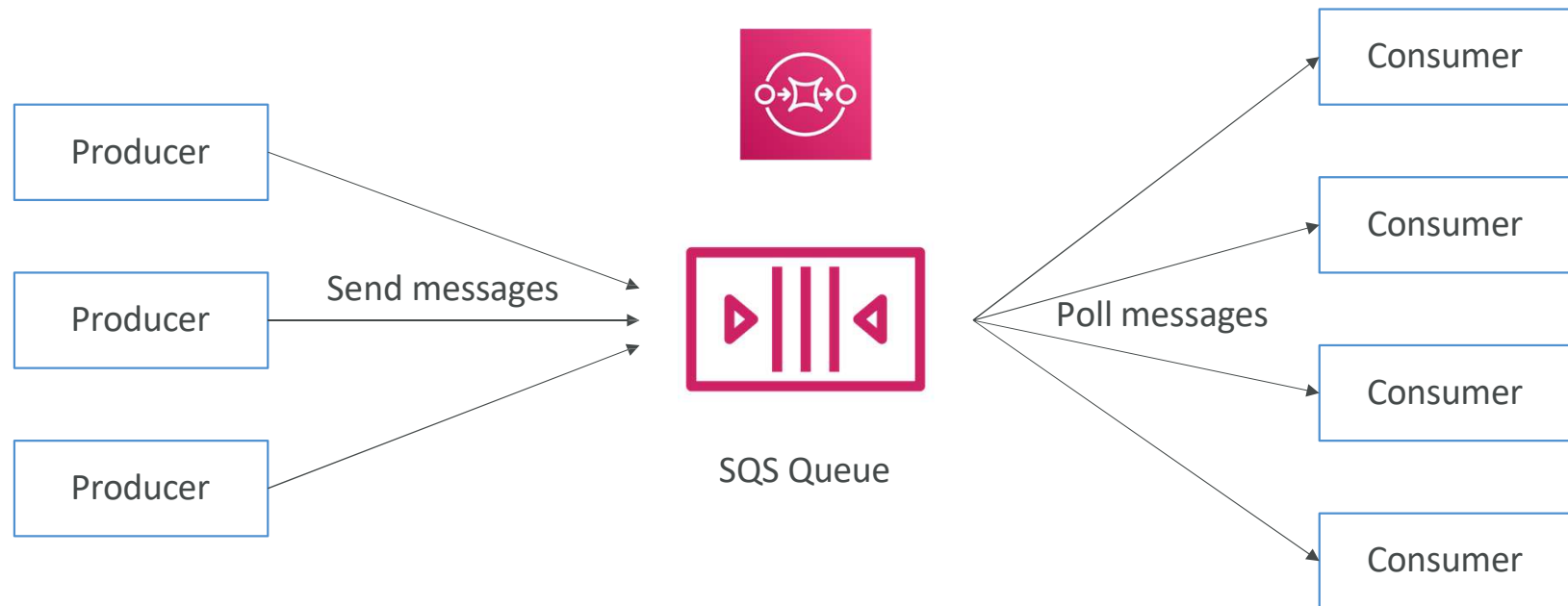


# Section Introduction

- Synchronous between applications can be problematic if there are sudden spikes of traffic
  - What if you need to suddenly encode 1000 videos but usually it's 10?
  - In that case, it's better to decouple your applications,
    - using SQS: queue model
    - using SNS: pub/sub model
    - using Kinesis: real-time streaming model
  - These services can scale independently from our application!
- 

# Amazon SQS

## What's a queue?



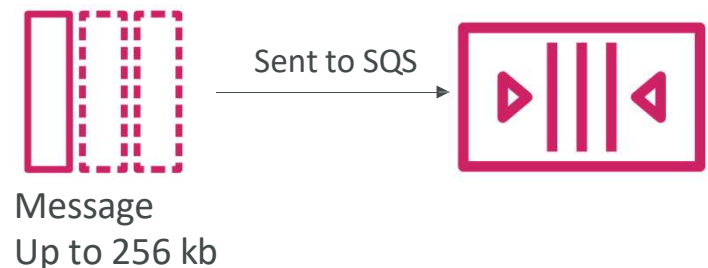
# Amazon SQS - Standard Queue



- Oldest offering (over 10 years old)
- Fully managed service, used to decouple applications
- Attributes:
  - Unlimited throughput, unlimited number of messages in queue
  - Default retention of messages: 4 days, maximum of 14 days
  - Low latency (<10 ms on publish and receive)
  - Limitation of 256KB per message sent
- Can have duplicate messages (at least once delivery, occasionally)
- Can have out of order messages (best effort ordering)

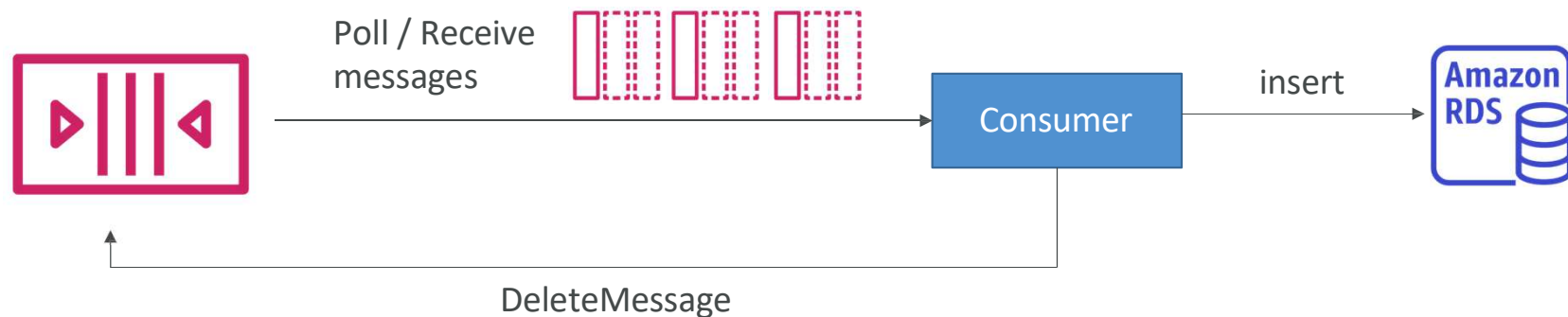
# SQS - Producing Messages

- Produced to SQS using the SDK (SendMessage API)
- The message is persisted in SQS until a consumer deletes it
- Message retention: default 4 days, up to 14 days
- Example: send an order to be processed
  - Order id
  - Customer id
  - Any attributes you want
- SQS standard: unlimited throughput

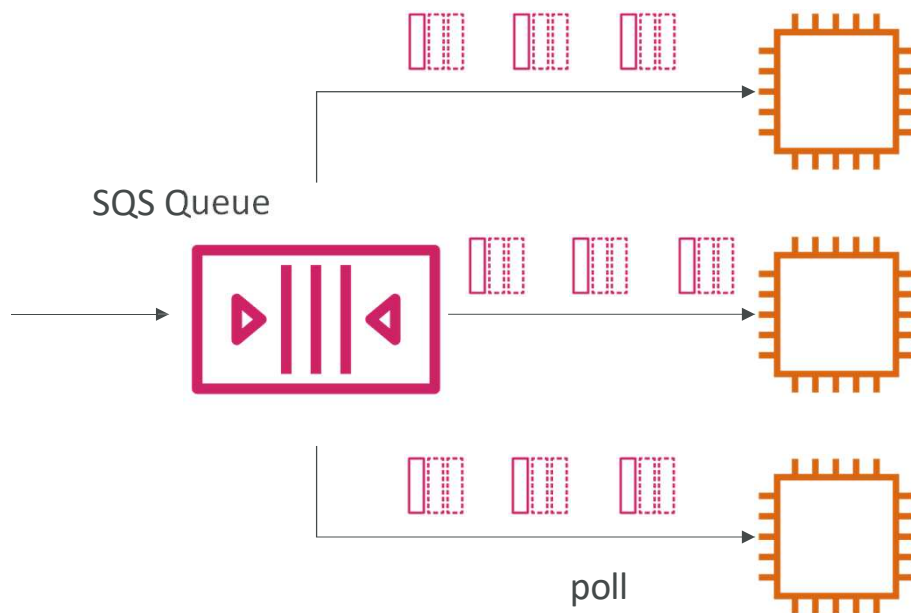


# SQS - Consuming Messages

- Consumers (running on EC2 instances, servers, or AWS Lambda)...
- Poll SQS for messages (receive up to 10 messages at a time)
- Process the messages (example: insert the message into an RDS database)
- Delete the messages using the DeleteMessage API



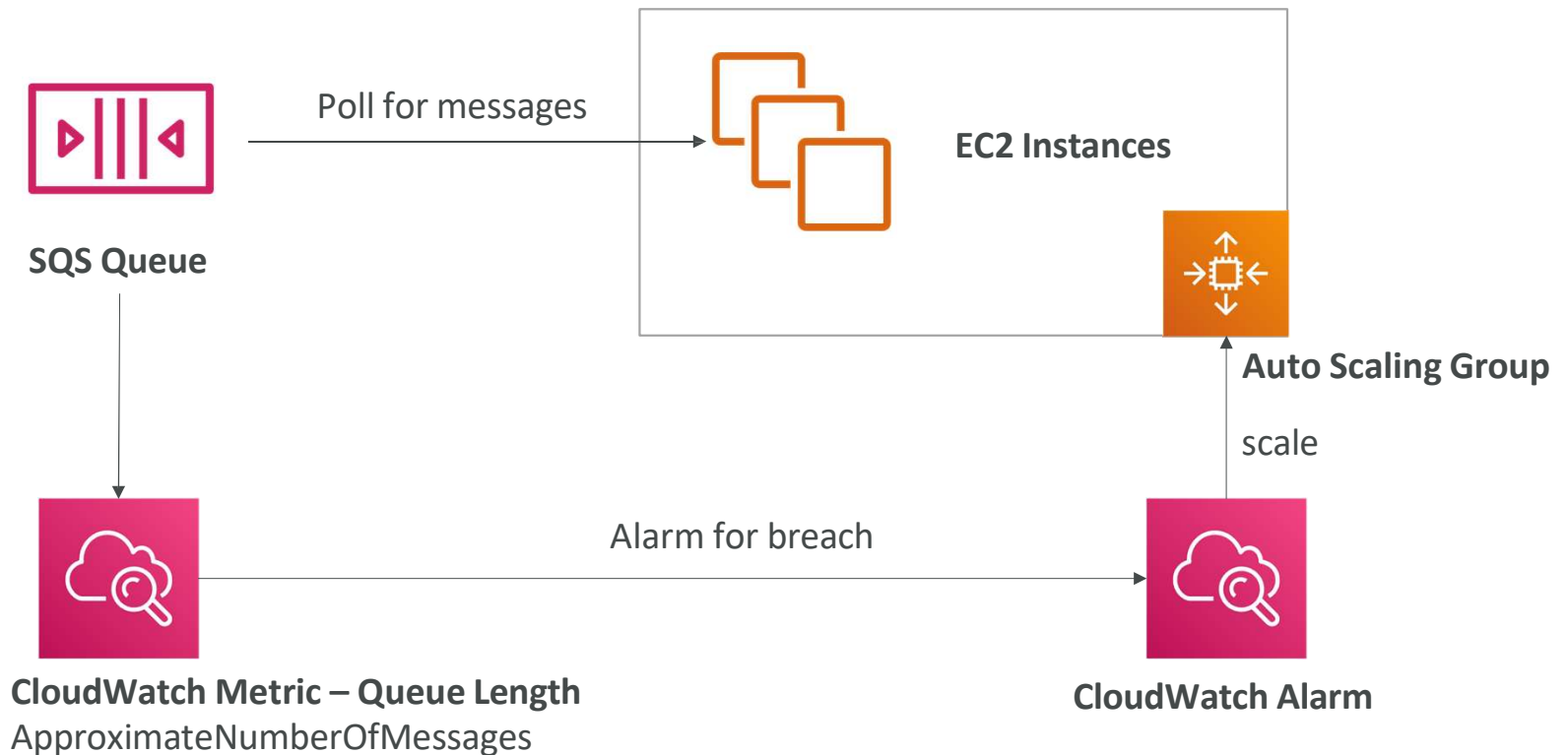
# SQS - Multiple EC2 Instances Consumers



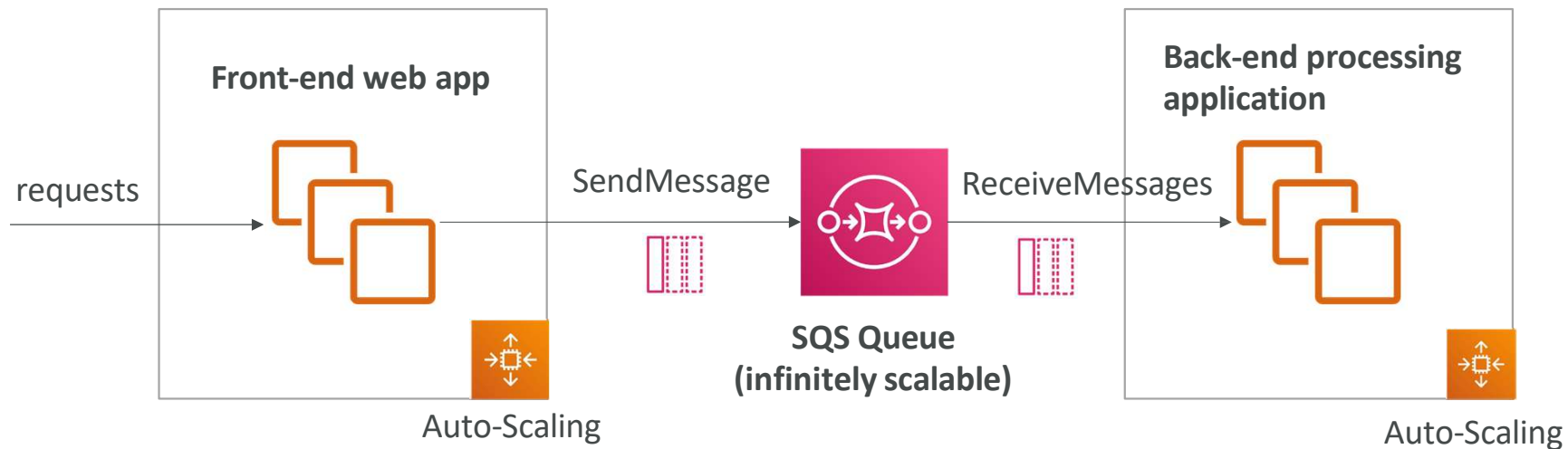
- Consumers receive and process messages in parallel
- At least once delivery
- Best-effort message ordering
- Consumers delete messages after processing them
- We can scale consumers horizontally to improve throughput of processing




# SQS with Auto Scaling Group (ASG)



# SQS to decouple between application tiers

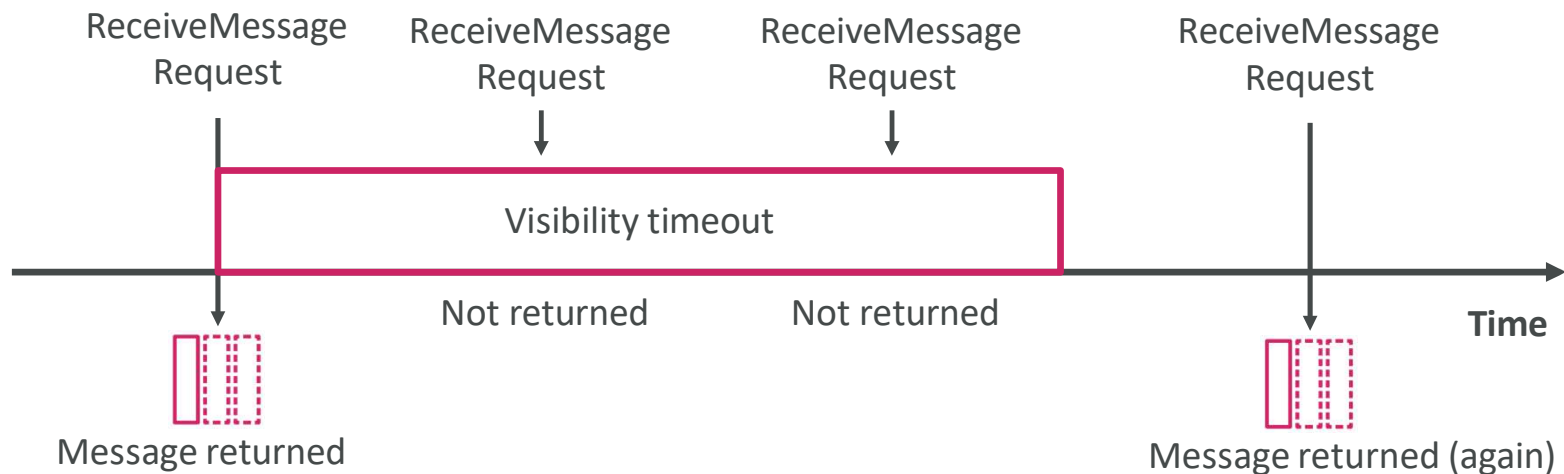


# Amazon SQS - Security

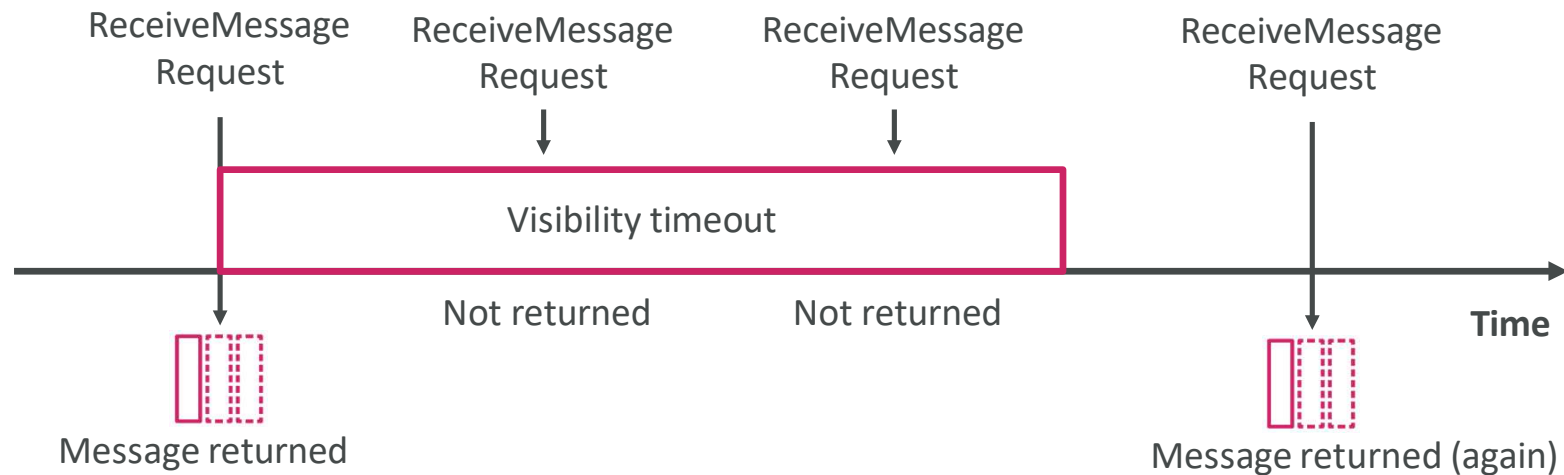
- Encryption:
    - In-flight encryption using HTTPS API
    - At-rest encryption using KMS keys
    - Client-side encryption if the client wants to perform encryption/decryption itself
  - Access Controls: IAM policies to regulate access to the SQS API
  - SQS Access Policies (similar to S3 bucket policies)
    - Useful for cross-account access to SQS queues
    - Useful for allowing other services (SNS, S3...) to write to an SQS queue
- 

# SQS - Message Visibility Timeout

- After a message is polled by a consumer, it becomes invisible to other consumers
- By default, the “message visibility timeout” is 30 seconds
- That means the message has 30 seconds to be processed
- After the message visibility timeout is over, the message is “visible” in SQS



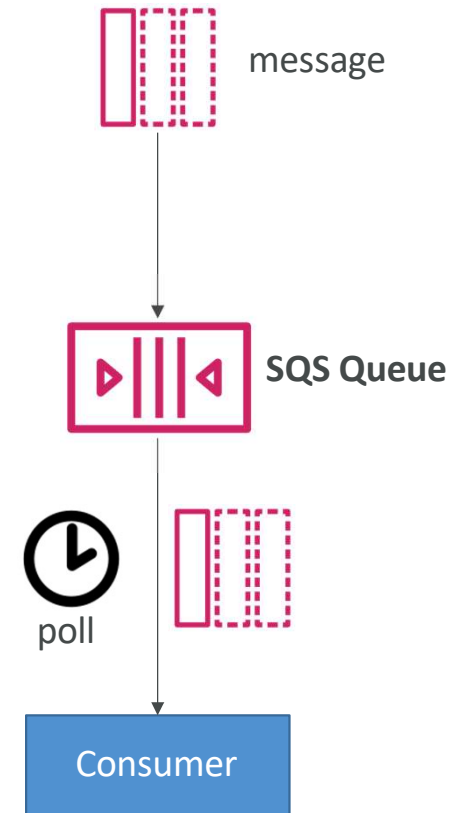
# SQS - Message Visibility Timeout



- If a message is not processed within the visibility timeout, it will be processed twice
- A consumer could call the `ChangeMessageVisibility` API to get more time
- If visibility timeout is high (hours), and consumer crashes, re-processing will take time
- If visibility timeout is too low (seconds), we may get duplicates

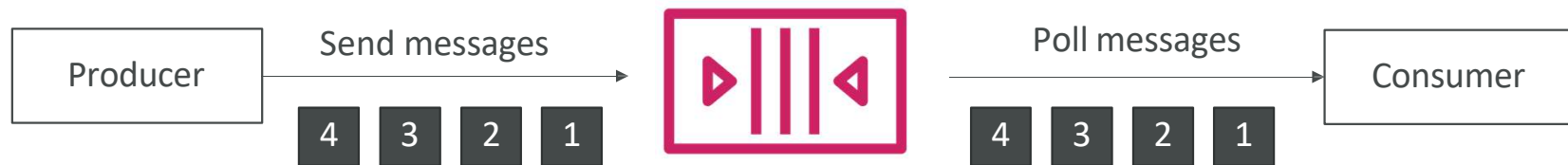
# Amazon SQS - Long Polling

- When a consumer requests messages from the queue, it can optionally “wait” for messages to arrive if there are none in the queue
- This is called Long Polling
- LongPolling decreases the number of API calls made to SQS while increasing the efficiency and reducing latency of your application
- The wait time can be between 1 sec to 20 sec (20 sec preferable)
- Long Polling is preferable to Short Polling
- Long polling can be enabled at the queue level or at the API level using WaitTimeSeconds



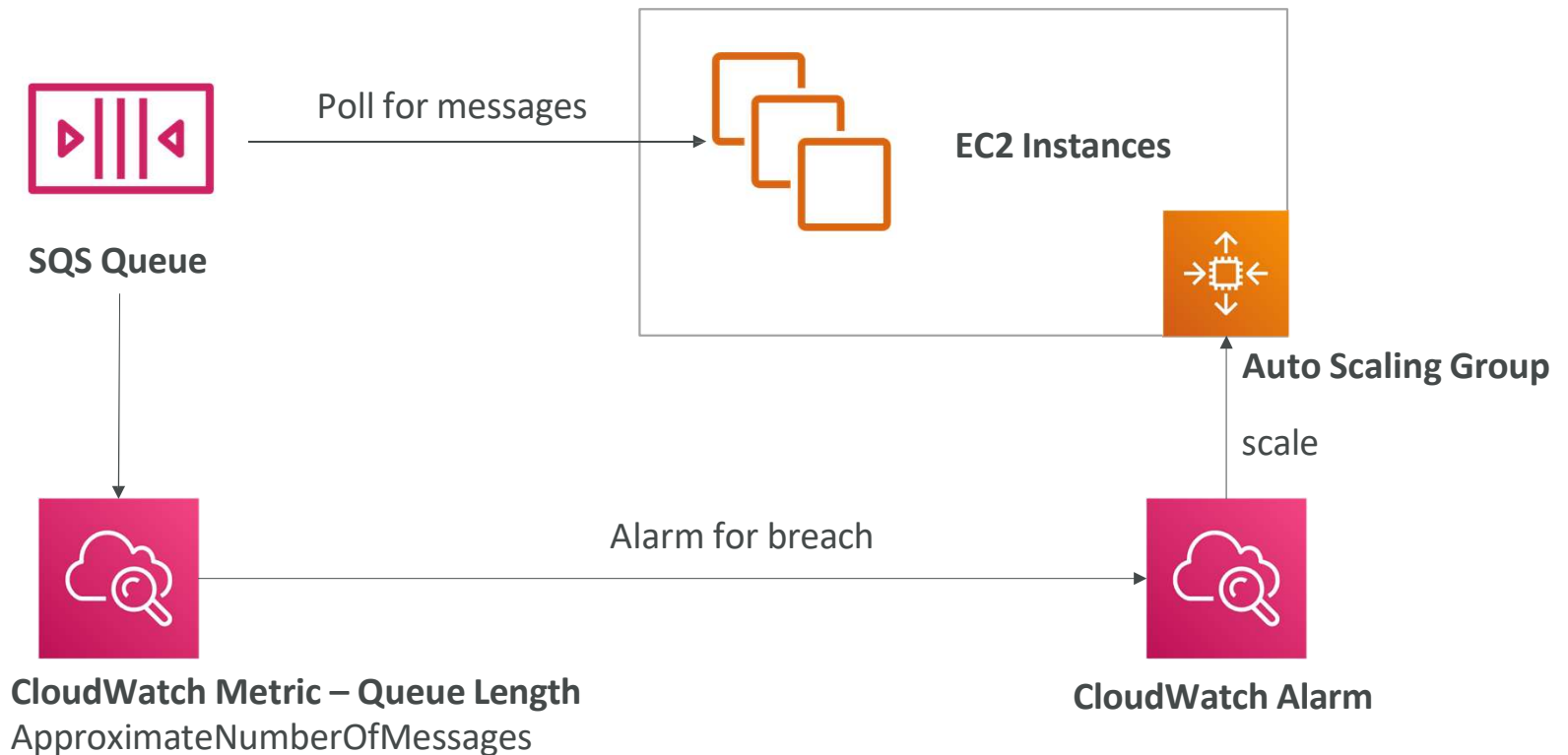
# Amazon SQS – FIFO Queue

- FIFO = First In First Out (ordering of messages in the queue)



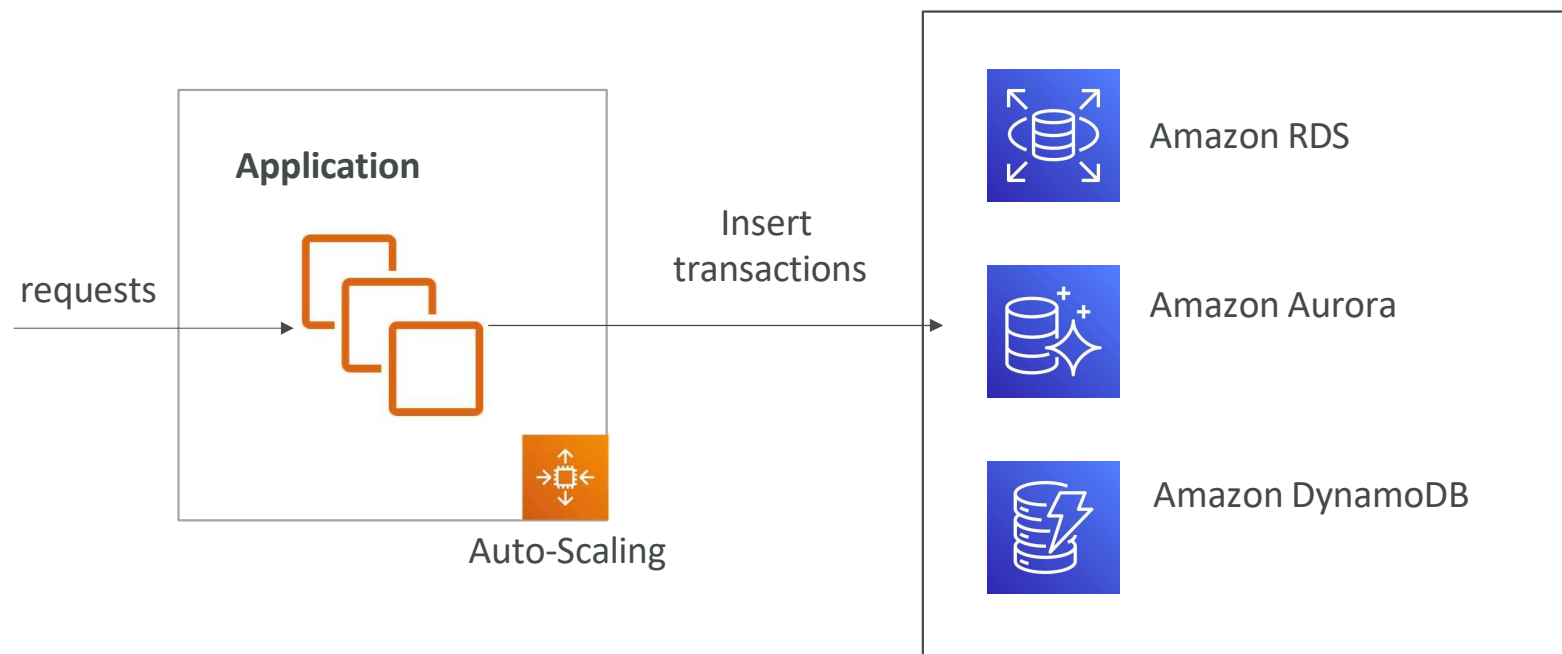
- Limited throughput: 300 msg/s without batching, 3000 msg/s with
- Exactly-once send capability (by removing duplicates)
- Messages are processed in order by the consumer

# SQS with Auto Scaling Group (ASG)

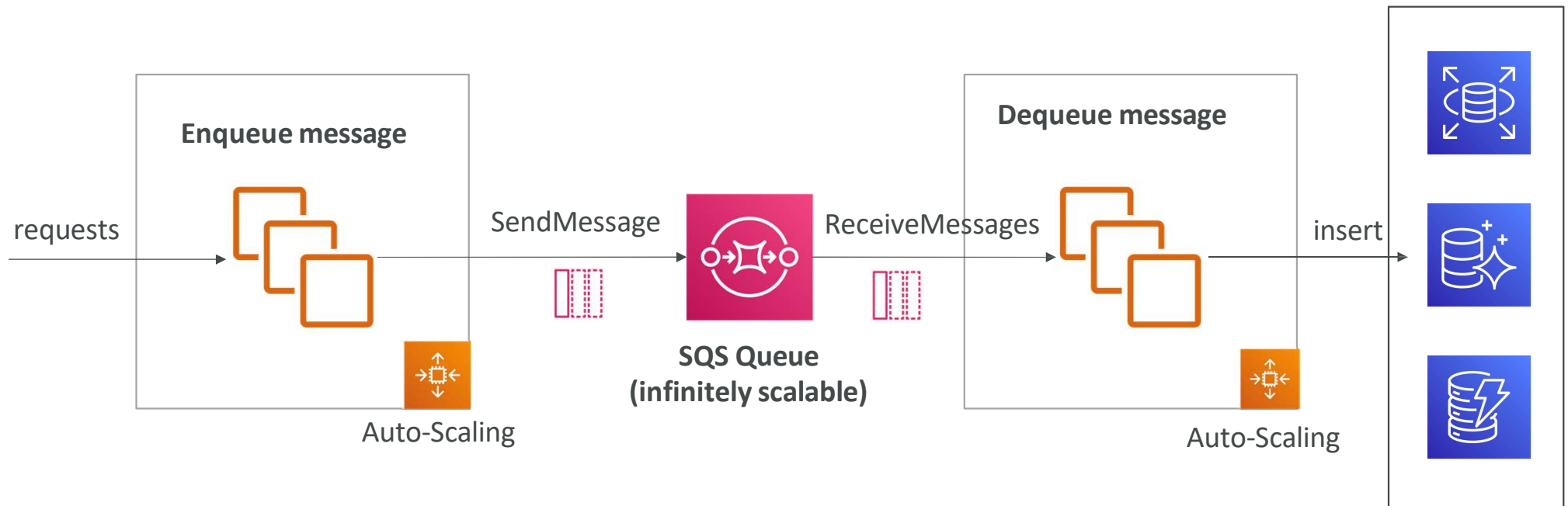




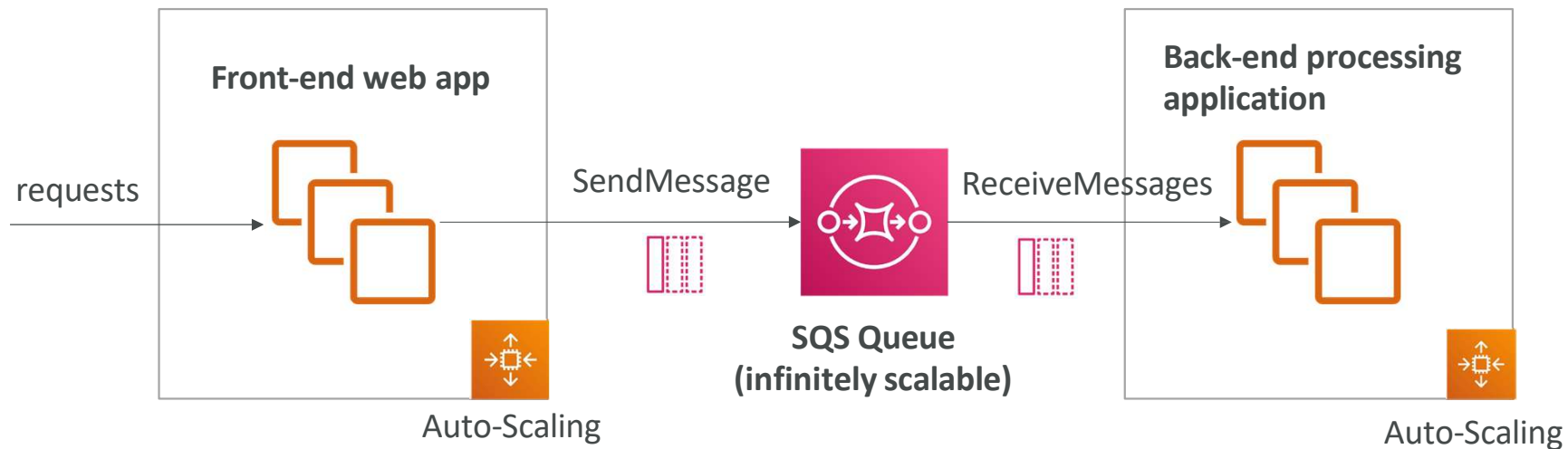
# If the load is too big, some transactions may be lost



# SQS as a buffer to database writes



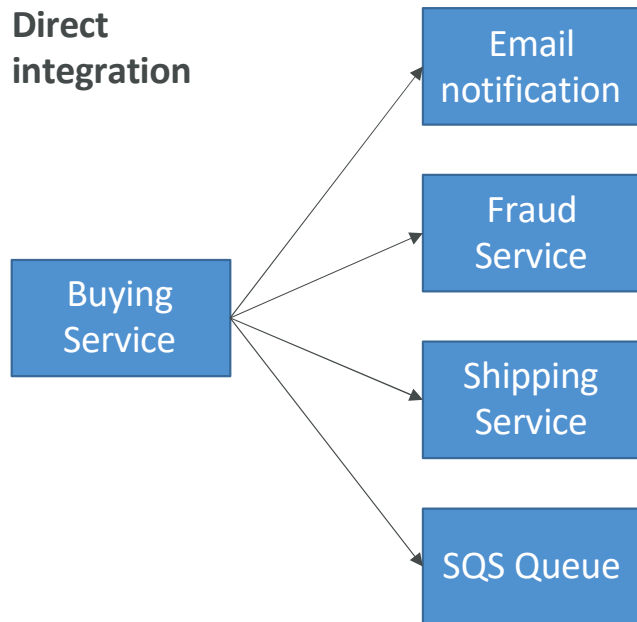
# SQS to decouple between application tiers



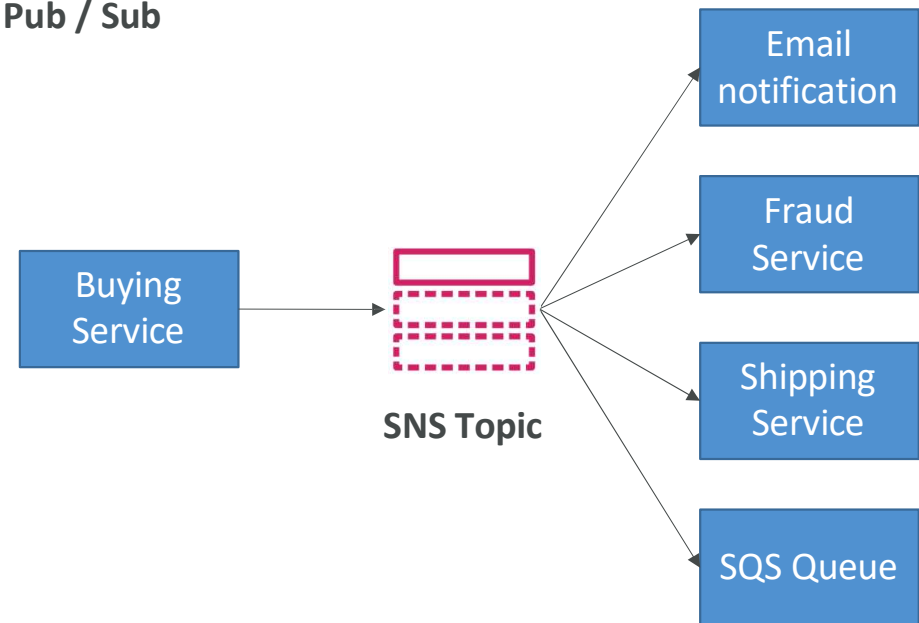
# Amazon SNS

- What if you want to send one message to many receivers?

Direct  
integration



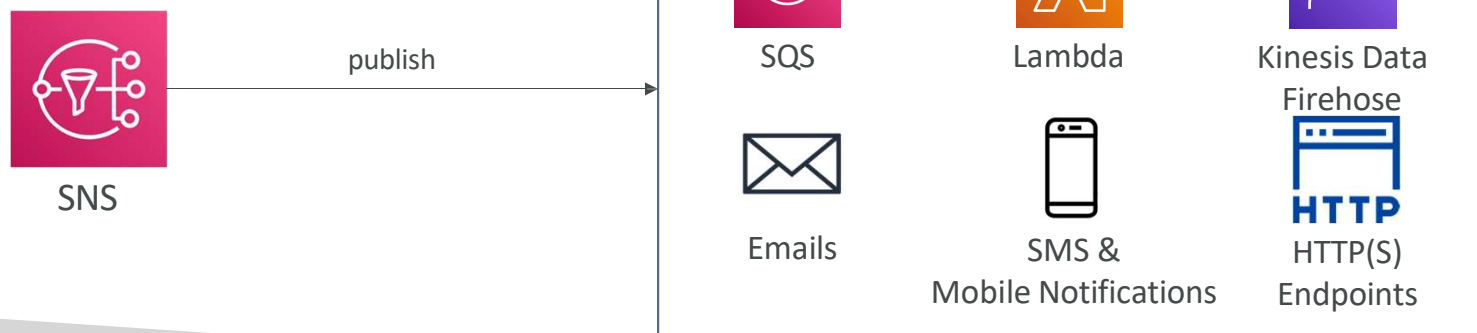
Pub / Sub



# Amazon SNS

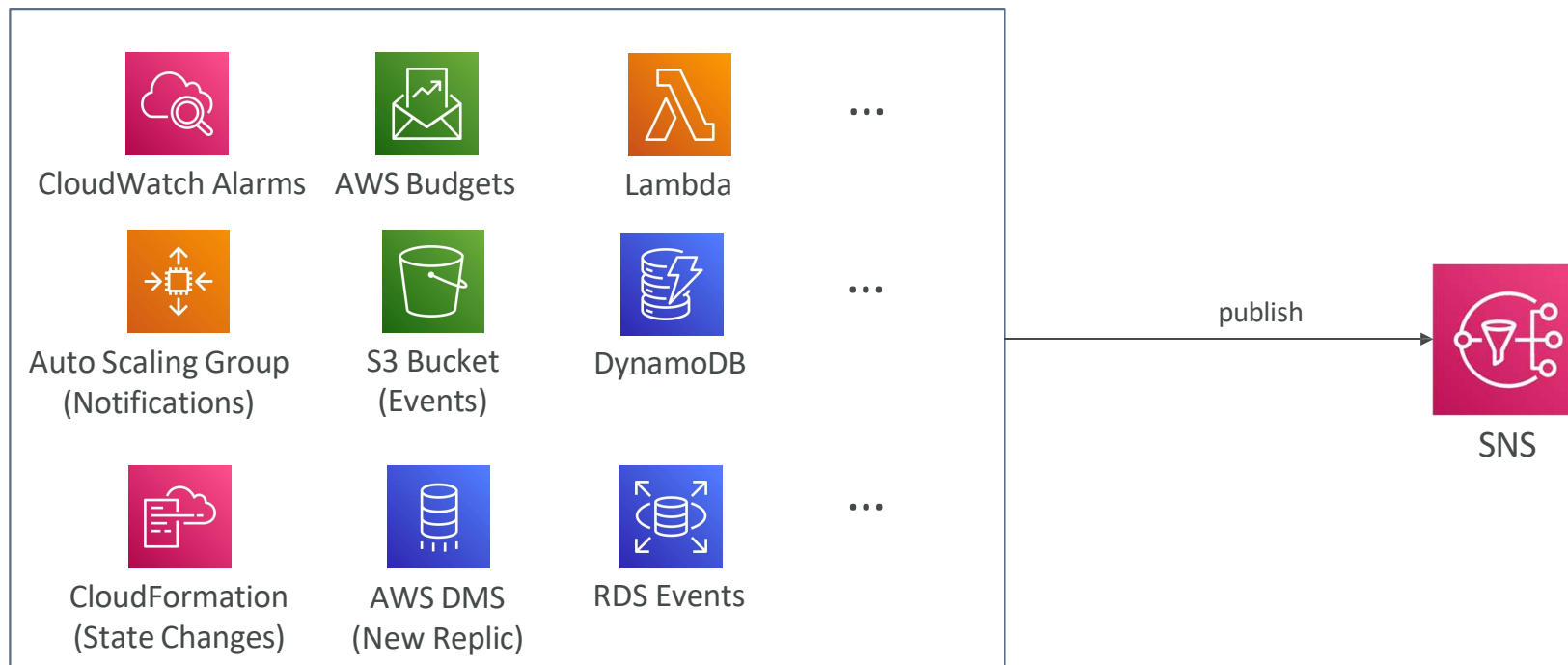


- The “event producer” only sends message to one SNS topic
- As many “event receivers” (subscriptions) as we want to listen to the SNS topic notifications
- Each subscriber to the topic will get all the messages (note: new feature to filter messages)
- Up to 12,500,000 subscriptions per topic
- 100,000 topics limit




# SNS integrates with a lot of AWS services


- Many AWS services can send data directly to SNS for notifications



# Amazon SNS - How to publish

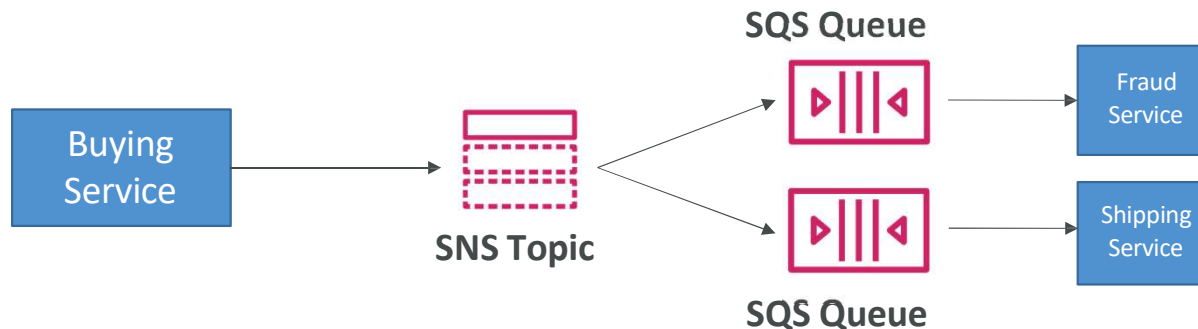
- Topic Publish (using the SDK)
    - Create a topic
    - Create a subscription (or many)
    - Publish to the topic
  - Direct Publish (for mobile apps SDK)
    - Create a platform application
    - Create a platform endpoint
    - Publish to the platform endpoint
    - Works with Google GCM, Apple APNS, Amazon ADM...
- 

# Amazon SNS - Security

- Encryption:
    - In-flight encryption using HTTPS API
    - At-rest encryption using KMS keys
    - Client-side encryption if the client wants to perform encryption/decryption itself
  - Access Controls: IAM policies to regulate access to the SNS API
  - SNS Access Policies (similar to S3 bucket policies)
    - Useful for cross-account access to SNS topics
    - Useful for allowing other services ( S3...) to write to an SNS topic
- 



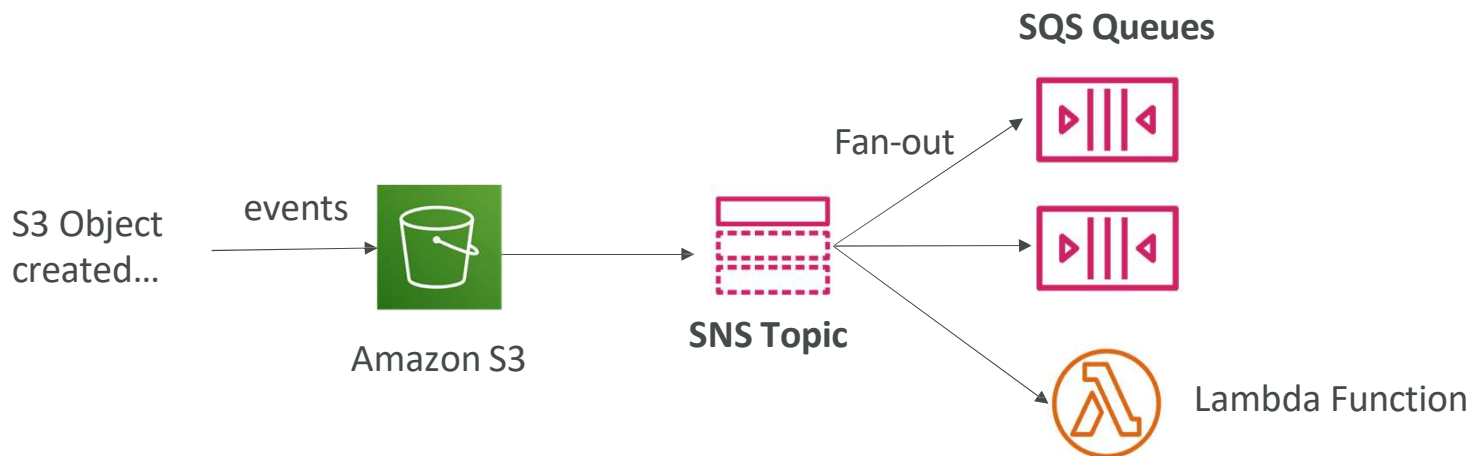
# SNS + SQS: Fan Out



- Push once in SNS, receive in all SQS queues that are subscribers
- Fully decoupled, no data loss
- SQS allows for: data persistence, delayed processing and retries of work
- Ability to add more SQS subscribers over time
- Make sure your SQS queue access policy allows for SNS to write
- Cross-Region Delivery: works with SQS Queues in other regions

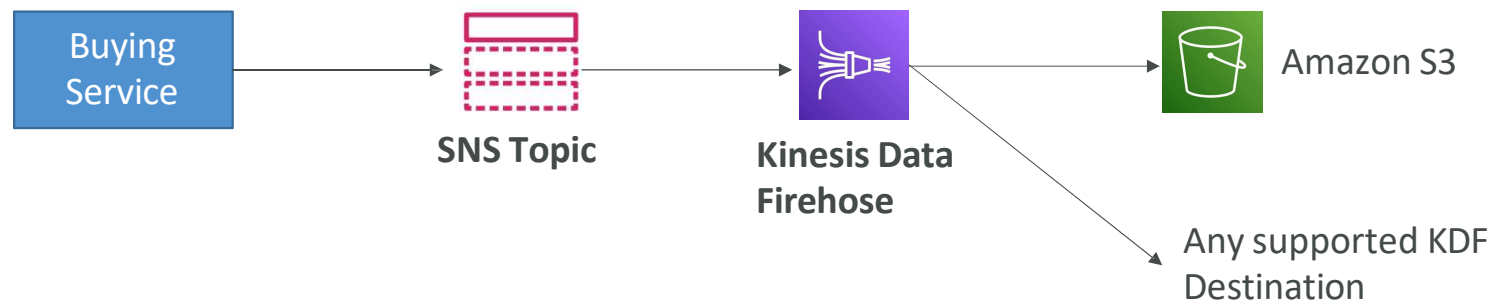
# Application: S3 Events to multiple queues

- For the same combination of: event type (e.g. object create) and prefix (e.g. images/) you can only have one S3 Event rule
- If you want to send the same S3 event to many SQS queues, use fan-out



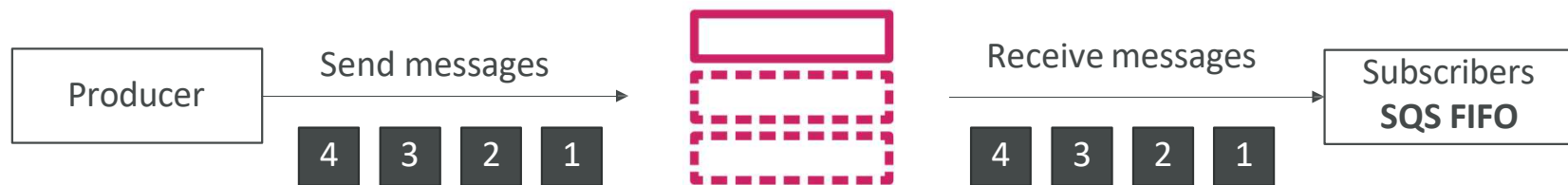
# Application: SNS to Amazon S3 through Kinesis Data Firehose

- SNS can send to Kinesis and therefore we can have the following solutions architecture:



# Amazon SNS - FIFO Topic

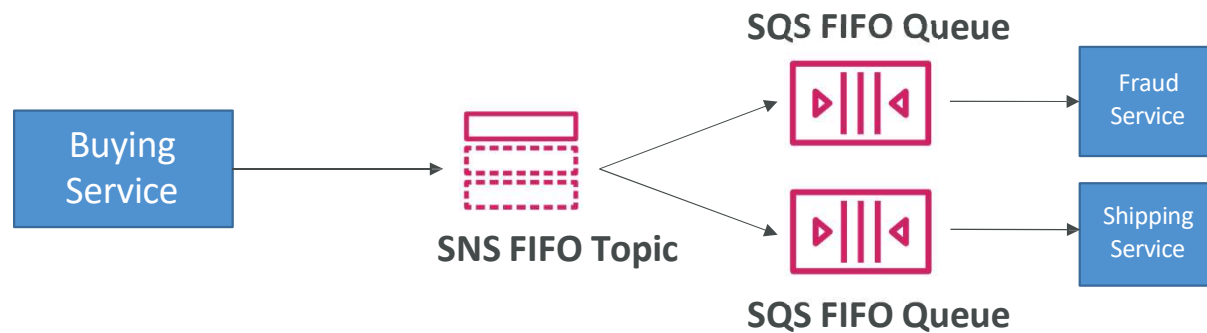
- FIFO = First In First Out (ordering of messages in the topic)



- Similar features as SQS FIFO:
  - Ordering by Message Group ID (all messages in the same group are ordered)
  - Deduplication using a Deduplication ID or Content Based Deduplication
- Can have SQS Standard and FIFO queues as subscribers
- Limited throughput (same throughput as SQS FIFO)

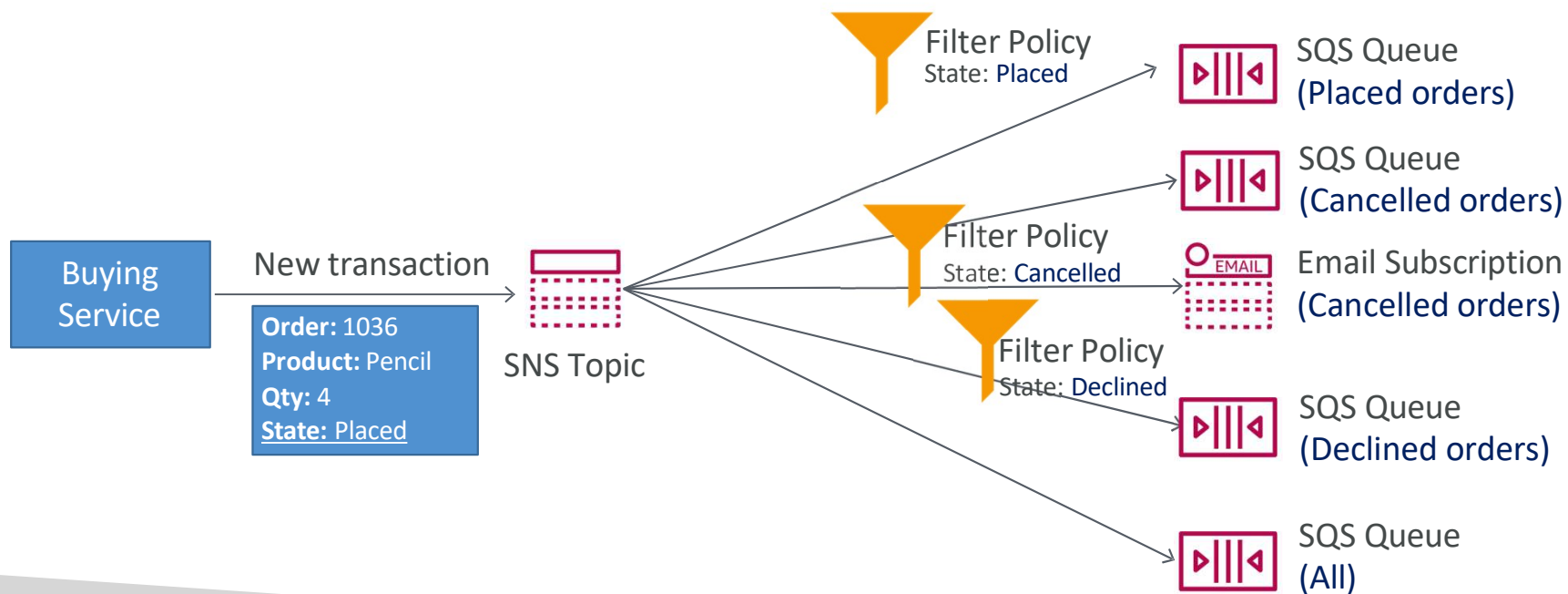
# SNS FIFO + SQS FIFO: Fan Out

- In case you need fan out + ordering + deduplication



# SNS - Message Filtering

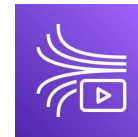
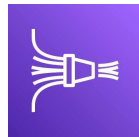
- JSON policy used to filter messages sent to SNS topic's subscriptions
- If a subscription doesn't have a filter policy, it receives every message



# Kinesis Overview

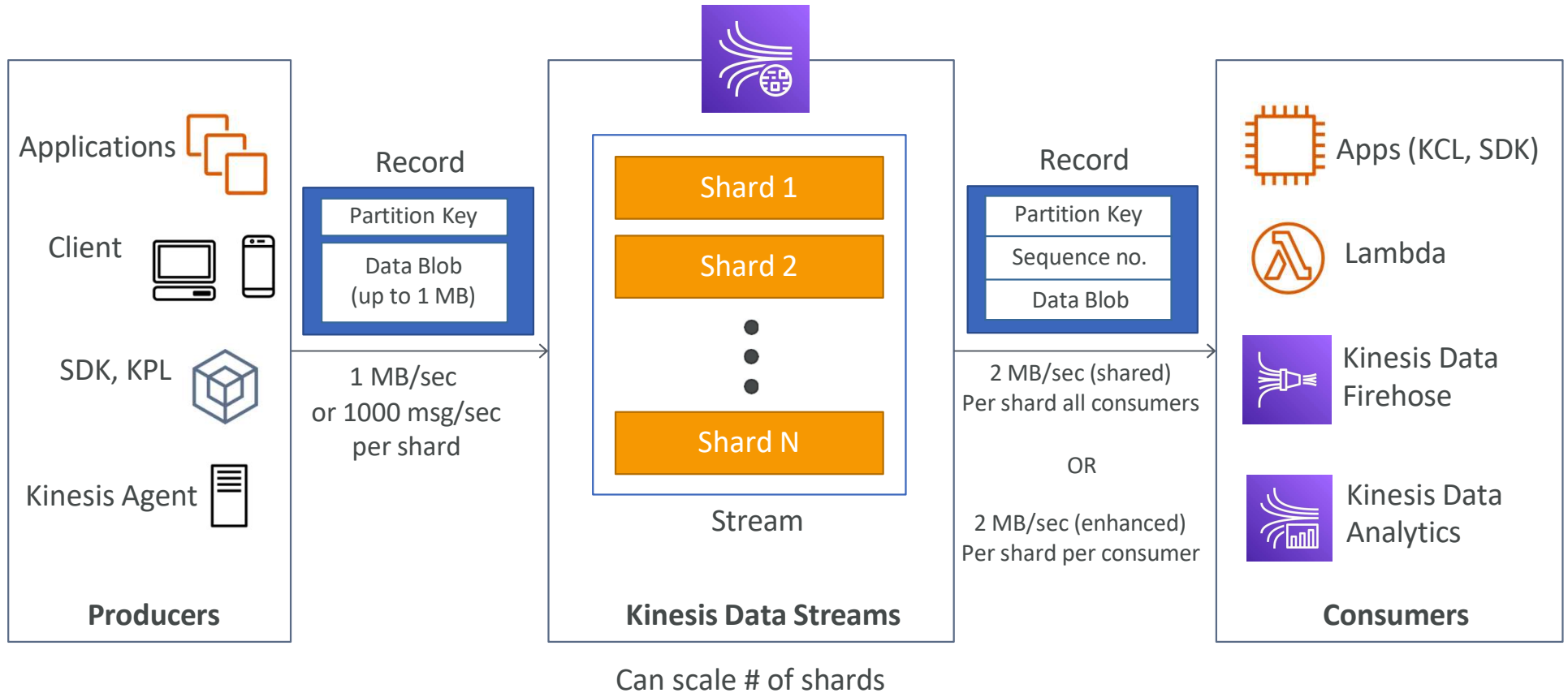


- Makes it easy to collect, process, and analyze streaming data in real-time
- Ingest real-time data such as: Application logs, Metrics, Website clickstreams, IoT telemetry data...



- Kinesis Data Streams: capture, process, and store data streams
- Kinesis Data Firehose: load data streams into AWS data stores
- Kinesis Data Analytics: analyze data streams with SQL or Apache Flink
- Kinesis Video Streams: capture, process, and store video streams

# Kinesis Data Streams






# Kinesis Data Streams



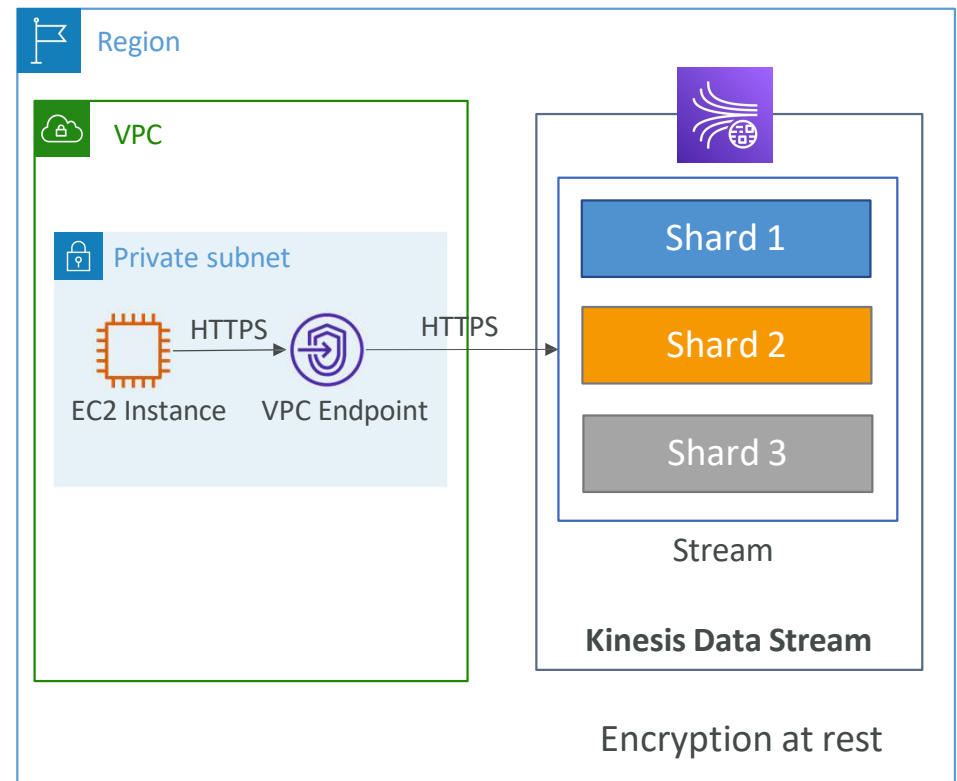
- Retention between 1 day to 365 days
- Ability to reprocess (replay) data
- Once data is inserted in Kinesis, it can't be deleted (immutability)
- Data that shares the same partition goes to the same shard (ordering)
- Producers: AWS SDK, Kinesis Producer Library (KPL), Kinesis Agent
- Consumers:
  - Write your own: Kinesis Client Library (KCL), AWS SDK
  - Managed: AWS Lambda, Kinesis Data Firehose, Kinesis Data Analytics,

# Kinesis Data Streams - Capacity Modes

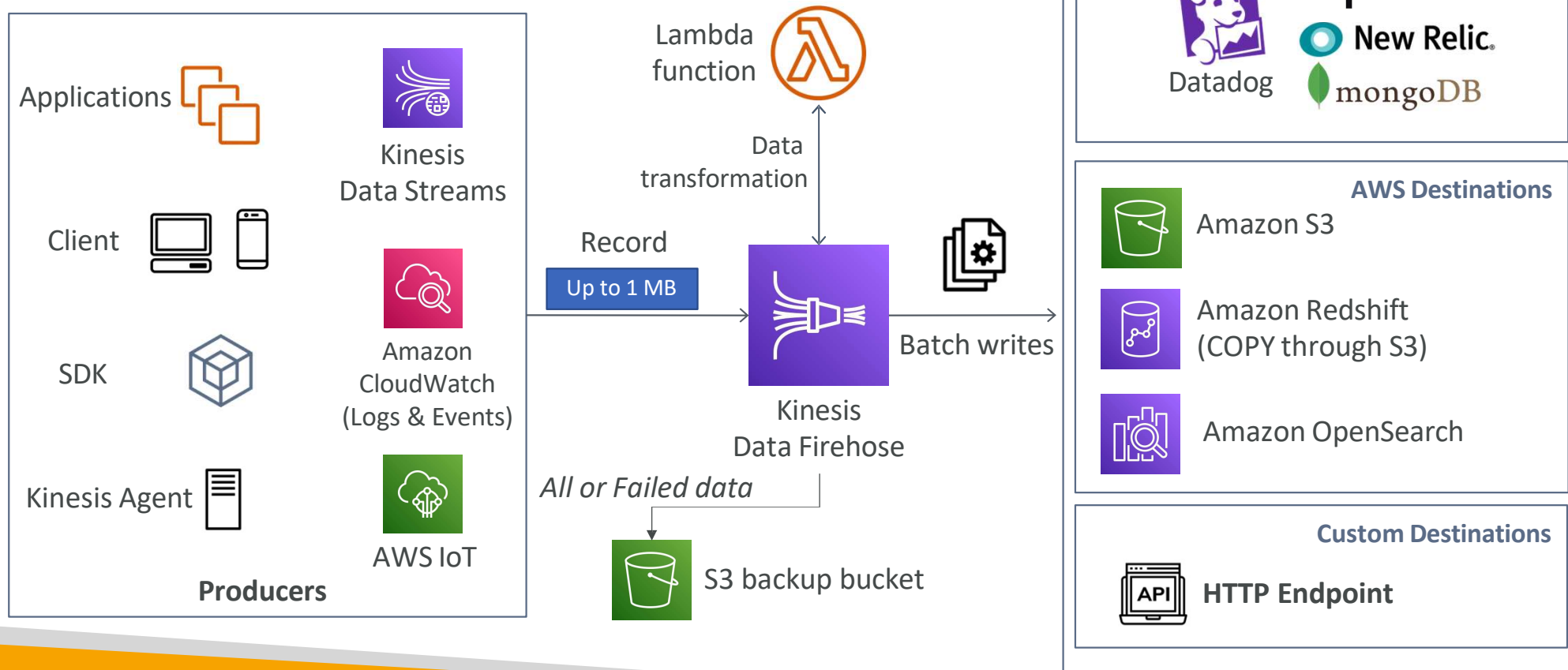
- Provisioned mode:
    - You choose the number of shards provisioned, scale manually or using API
    - Each shard gets 1MB/s in (or 1000 records per second)
    - Each shard gets 2MB/s out (classic or enhanced fan-out consumer)
    - You pay per shard provisioned per hour
  - On-demand mode:
    - No need to provision or manage the capacity
    - Default capacity provisioned (4 MB/s in or 4000 records per second)
    - Scales automatically based on observed throughput peak during the last 30 days
    - Pay per stream per hour & data in/out per GB
- 

# Kinesis Data Streams Security

- Control access / authorization using IAM policies
- Encryption in flight using HTTPS endpoints
- Encryption at rest using KMS
- You can implement encryption/decryption of data on client side (harder)
- VPC Endpoints available for Kinesis to access within VPC
- Monitor API calls using CloudTrail



# Kinesis Data Firehose



# Kinesis Data Firehose



- Fully Managed Service, no administration, automatic scaling, serverless
  - AWS: Redshift / Amazon S3 / OpenSearch
  - 3rd party partner: Splunk / MongoDB / DataDog / NewRelic / ...
  - Custom: send to any HTTP endpoint
- Pay for data going through Firehose
- Near Real Time
  - Buffer interval: 0 seconds (no buffering) to 900 seconds
  - Buffer size: minimum 1MB
- Supports many data formats, conversions, transformations, compression
- Supports custom data transformations using AWS Lambda
- Can send failed or all data to a backup S3 bucket

# Kinesis Data Streams vs Firehose



## Kinesis Data Streams

- Streaming service for ingest at scale
- Write custom code (producer / consumer)
- Real-time (~200 ms)
- Manage scaling (shard splitting / merging)
- Data storage for 1 to 365 days
- Supports replay capability

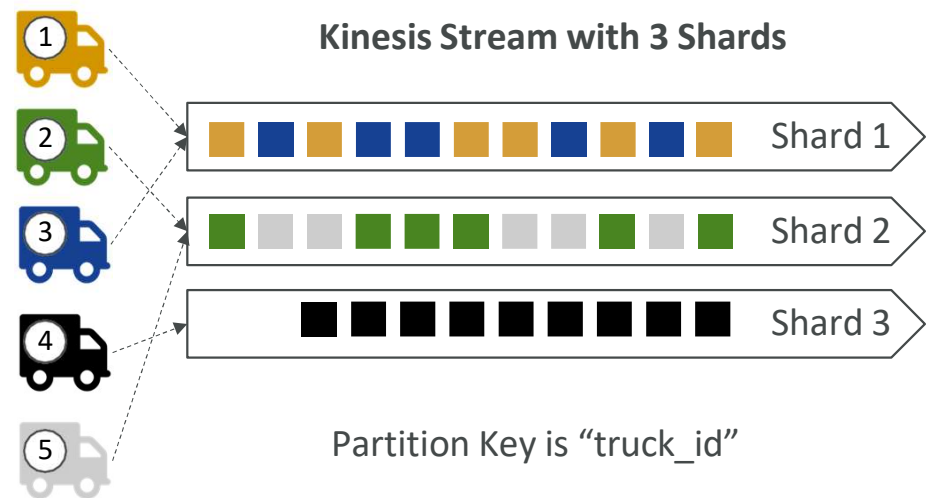


## Kinesis Data Firehose

- Load streaming data into S3 / Redshift / OpenSearch / 3<sup>rd</sup> party / custom HTTP
- Fully managed
- Near real-time
- Automatic scaling
- No data storage
- Doesn't support replay capability

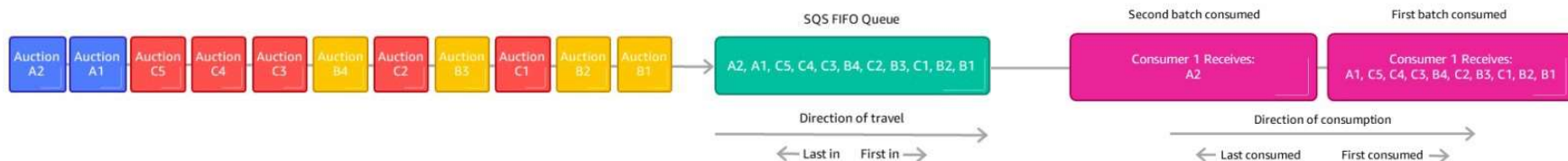
# Ordering data into Kinesis

- Imagine you have 100 trucks (truck\_1, truck\_2, ... truck\_100) on the road sending their GPS positions regularly into AWS.
- You want to consume the data in order for each truck, so that you can track their movement accurately.
- How should you send that data into Kinesis?
- Answer: send using a “Partition Key” value of the “truck\_id”
- The same key will always go to the same shard

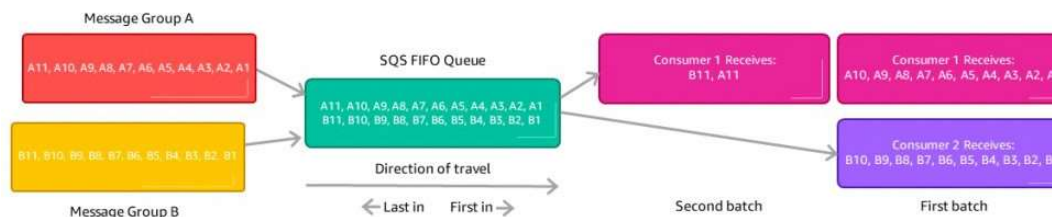


# Ordering data into SQS

- For SQS standard, there is no ordering.
- For SQS FIFO, if you don't use a Group ID, messages are consumed in the order they are sent, with only one consumer



- You want to scale the number of consumers, but you want messages to be “grouped” when they are related to each other
- Then you use a Group ID (similar to Partition Key in Kinesis)





# Kinesis vs SQS ordering

- Let's assume 100 trucks, 5 kinesis shards, 1 SQS FIFO
- Kinesis Data Streams:
  - On average you'll have 20 trucks per shard
  - Trucks will have their data ordered within each shard
  - The maximum amount of consumers in parallel we can have is 5
  - Can receive up to 5 MB/s of data
- SQS FIFO
  - You only have one SQS FIFO queue
  - You will have 100 Group ID
  - You can have up to 100 Consumers (due to the 100 Group ID)
  - You have up to 300 messages per second (or 3000 if using batching)

# SQS vs SNS vs Kinesis

## SQS:



- Consumer “pull data”
- Data is deleted after being consumed
- Can have as many workers (consumers) as we want
- No need to provision throughput
- Ordering guarantees only on FIFO queues
- Individual message delay capability

## SNS:



- Push data to many subscribers
- Up to 12,500,000 subscribers
- Data is not persisted (lost if not delivered)
- Pub/Sub
- Up to 100,000 topics
- No need to provision throughput
- Integrates with SQS for fan-out architecture pattern
- FIFO capability for SQS FIFO

## Kinesis:



- Standard: pull data
  - 2 MB per shard
- Enhanced-fan out: push data
  - 2 MB per shard per consumer
- Possibility to replay data
- Meant for real-time big data, analytics and ETL
- Ordering at the shard level
- Data expires after X days
- Provisioned mode or on-demand capacity mode

# Amazon MQ



- SQS, SNS are “cloud-native” services: proprietary protocols from AWS
- Traditional applications running from on-premises may use open protocols such as: MQTT, AMQP, STOMP, Openwire, WSS
- When migrating to the cloud, instead of re-engineering the application to use SQS and SNS, we can use Amazon MQ
- Amazon MQ is a managed message broker service for



- Amazon MQ doesn’t “scale” as much as SQS / SNS
- Amazon MQ runs on servers, can run in Multi-AZ with failover
- Amazon MQ has both queue feature (~SQS) and topic features (~SNS)

# Amazon MQ - High Availability

