SQS: Simple Queue Service

If you are studying for AWS Developer Associate Exam, this guide will help you with quick revision before the exam. it can use as study notes for your preparation.

Dashboard Other Certification Notes

SQS: Simple Queue Service

SQS is a service that manages and operates message oriented middleware. It enables you to decouple and scale micro-services, distributed systems, and serverless applications.

What's a queue? - A form of asynchronous service to service communication used in multiple application architectures. - Messages are stored on teh queue until they are processed and deleted

Standard Queue

- Oldest offering from AWS (over 10 years old)
- Fully managed service
- Scales from 1 message per second to 10,000 per second
- Default retention of messages: 4 days
  - A maximum of 14 days
- There is no limit to how many messages can be in the queue
- Low latency (10 ms on publish and receive)
- Horizontal scaling in terms of number of consumers
- Can have duplicate messages
- Can have messages out of order (best effort ordering)
- Limitation of 256kb per message sent

Delay Queue

- Delay a message so a consumer doesn't see it immediately
  - up to 15 minute delay
- Default is 0 seconds making messages available right away
- You can set a default at queue level
- YOu can override the default using the **DelaySeconds** parameter

Producing Messages

- Define the body
- Add message attributes (metadata - optional)
- Optionally can provide Delay Delivery
- You receive back:
  - Message identifier
  - MD5 hash of the body

Consuming Messages

- Poll SQS for messages
  - receive up to 10 messages at a time
- Next step is to process the message within the visibility timeout
- Finally, delete the message using the message ID & receipt handle

Visibility timeout

- When a consumer polls a message from a queue, the message is *invisible* to the other consumers for a defined period called **Visibility Timeout**
  - Can be set between 0 seconds to 12 hours
  - Default 30 seconds
  - If it takes 15 minutes or greater and consumer fails to process the message, you must wait before processing the message again
  - IF set to 30 seconds or lower and consumer needs time to process the message, another consumer will receive the message and will be processed more than once

- **ChangeMessageVisibility**
  - An API to change the visibility while processing a message
- **DeleteMessage**
  - An API to tell SQS the message was successfully processed

Dead Letter Queue

- If a consumer fails to consume a message within the visibility timeout, the message goes back to the queue. We can put a threshold of how many times a message can be put back to the queue
- After **MaximumReceives** times the threshold is exceeded and the message goes into dead letter queue (DLQ)
  - DLQ is useful for debugging
  - Retention times for the messages should be set to max (14 days) in case of a DLQ
  - DLQ is a queue which is created separately and attached to the normal queue

Long Polling

- When a consumer requests for a message from a queue it can optionally wait for messages to arrive in case there is no available message to be consumed in the queue
- Long polling decreases the number of API calls made to SQS while increasing the efficiency and latency of the application
- The wait time can be set to 1 second up to 20 seconds.
- Long polling should be preferred in case of short polling (wait time of 0 seconds)
- Long polling is enabled by setting a value between 1 and 20 for **WaitTimeSeconds** queue attribute

SQS Extended Client

- Message size limit is 265KB in case of a queue, Extended Client Java library offers an implementation for larger messages using S3
- The data is uploaded into S3, the message will contain a pointer to the data. When the message is consumed, the data is retrieved from the S3 using the pointer

SQS API

- **CreateQueue(MessageRetentionPeriod)**: creates a queue
- **DeleteQueue**: deletes a queue
- **PurgeQueue**: deletes all the messages from a queue
- **SendMessage(DelaySeconds)**: send a message with optional delay
- **ReceiveMessageWaitTimeSeconds**: receives a message using long polling
- **ChangeMessageVisibility**: changes the message wait time in case there is more time needed for processing
- Batch API: **SendMessage**, **DeleteMessage**, **ChangeMessageVisibility**

FIFO Queues

- FIFO = First In First Out
- Provides ordering of the messages in a queue
- Has limited throughput compared to standard queues: 300 msg/s without batching, 3000 msg/s with batching
- Exactly-once send capability
- Messages are processed in order by the consumer
- Queues which are created as FIFO should have their name ending with “fifo”

FIFO Deduplication

- Used for making sure that a message is sent only once
- Deduplication interval is 5 minutes
- Deduplication methods:
  - Content based deduplication: the message body is hashed with SHA-256 algorithm and the hash is used to detect duplicates
  - **MessageDeduplicationID**: attribute can be set explicitly for checking duplicates

FIFO Message Grouping

- **MessageGroupID**: if this attribute is set, a message group can have only one consumer and the messages will arrive in order for the same message group
- Provides ordering at a subset level of messages in case of a queue:
  - Messages which share the same group id arrive in order within the group
  - Each group id can have a different consumer. A queue can have multiple groups
  - Ordering across the groups are not guaranteed