SNS and SQS

SNS

producer sends notification to a topic.

This topic can fan out messages to http/email/sqs-queues/lambda

Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. In Amazon SNS, there are two types of clients—publishers and subscribers—also referred to as producers and consumers. Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel. Subscribers (that is, web servers, email addresses, Amazon SQS queues, AWS Lambda functions) consume or receive the message or notification over one of the supported protocols (that is, Amazon SQS, HTTP/S, email, SMS, Lambda) when they are subscribed to the topic.

When using Amazon SNS, you (as the owner) create a topic and control access to it by defining policies that determine which publishers and subscribers can communicate with the topic. A publisher sends messages to topics that they have created or to topics they have permission to publish to. Instead of including a specific destination address in each message, a publisher sends a message to the topic. Amazon SNS matches the topic to a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers. Each topic has a unique name that identifies the Amazon SNS endpoint for publishers to post messages and subscribers to register for notifications. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

SQS

Requests: send receive and delete

100.000 free requests every month and then pay 1$/million

backlog 14 days

A producer (component 1) sends message A to a queue, and the message is distributed across the Amazon SQS servers redundantly.

When a consumer (component 2) is ready to process messages, it consumes messages from the queue, and message A is returned. While message A is being processed, it remains in the queue and isn't returned to subsequent receive requests for the duration of the visibility timeout.

The consumer (component 2) deletes message A from the queue to prevent the message from being received and processed again when the visibility timeout expires.

Note
Amazon SQS automatically deletes messages that have been in a queue for more than maximum message retention period. The default message retention period is 4 days. However, you can set the message retention period to a value from 60 seconds to 1,209,600 seconds (14 days) using the SetQueueAttributes action.

Q: How is Amazon SQS different from Amazon SNS?

Amazon SNS allows applications to send time-critical messages to multiple subscribers through a “push” mechanism, eliminating the need to periodically check or “poll” for updates. Amazon SQS is a message queue service used by distributed applications to exchange messages through a polling model, and can be used to decouple sending and receiving components.

Yes. FIFO (first-in-first-out) queues preserve the exact order in which messages are sent and received. If you use a FIFO queue, you don't have to place sequencing information in your messages. For more information, see FIFO Queue Logicin the *Amazon SQS Developer Guide*.

Standard queues provide a loose-FIFO capability that attempts to preserve the order of messages. However, because standard queues are designed to be massively scalable using a highly distributed architecture, receiving messages in the exact order they are sent is not guaranteed.

Q: Does Amazon SQS guarantee delivery of messages?

Standard queues provide at-least-once delivery, which means that each message is delivered at least once.

FIFO queues provide exactly-once processing, which means that each message is delivered once and remains available until a consumer processes it and deletes it. Duplicates are not introduced into the queue.

FIFO (First-In-First-Out) queues are designed to enhance messaging between applications when the order of operations and events is critical, or where duplicates can't be tolerated, 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.

FIFO queues also provide exactly-once processing but have a limited number of transactions per second (TPS):

- By default, FIFO queues support up to 3,000 messages per second, per API action (SendMessage, ReceiveMessage, or DeleteMessage), with batching. To request a limit increase, file a support request.
- FIFO queues support up to 300 messages per second, per API action (SendMessage, ReceiveMessage, or DeleteMessage) without batching.

Note

The name of a FIFO queue must end with the .fifo suffix. The suffix counts towards the 80-character queue name limit. To determine whether a queue is FIFO, you can check whether the queue name ends with the suffix.