Developing with Messaging Services

ASYNC PROCESSING · AMAZON SQS · STANDARD & FIFO QUEUES · AMAZON SNS · FANOUT & FILTERING · KINESIS DATA STREAMS STUDY & REVIEW

01 MUST KNOW

if you remember five things, remember these

01 **Decouple with messaging:** a **producer** hands a message to a managed service and moves on; a **consumer** processes it later, independently. This **loose coupling** improves fault tolerance and responsiveness versus a tightly coupled synchronous call chain, where one failed component fails the whole system.

02 **Three patterns, three services: message queues → Amazon SQS**, **pub/sub → Amazon SNS**, **data streams → Kinesis Data Streams**. Queues: one consumer processes then *deletes* each message. Pub/sub: SNS *pushes* a copy to every subscriber. Streams: many consumers *read the same* high-volume records and never delete them.

03 **Standard vs FIFO SQS: standard** = best-effort ordering, **at-least-once** delivery, nearly unlimited throughput. **FIFO** = strict ordering, **exactly-once** processing, more limited throughput. Choose FIFO only when order or deduplication is critical — otherwise handle order/duplicates in code.

04 **SQS message lifecycle:** `SendMessage` → `ReceiveMessage` (starts the **visibility timeout**, default 30s, range 0s–12h) → consumer must `DeleteMessage` before it expires, or the message reappears for another consumer. After **maxReceiveCount** failed attempts a redrive policy moves it to a **dead-letter queue**.

05 **SNS fanout and filtering:** publishers post to a **topic**; SNS pushes to all subscribers — **A2A** (SQS, Lambda, Firehose, HTTP/S) and **A2P** (email, SMS, push). Subscribe multiple SQS queues to a topic to **fanout** one event to parallel consumers; a **filter policy** (JSON on the subscription) delivers only matching messages.

02 KEY TERMS

TERM	DEFINITION
Producer	Component that creates and sends messages to a messaging service, then moves on without waiting for a result
Consumer	Component that retrieves and processes messages from a queue or stream
Loosely coupled	Design where components run and fail independently; a consumer failure does not stall the producer (fault tolerant)
Message queue	Temporary repository that holds a message until a consumer retrieves, processes, and deletes it
Pub/sub messaging	Publishers send to topics; messages are pushed to all subscribers, who never poll
Visibility timeout	Period after ReceiveMessage during which a message is hidden from other consumers (default 30s, range 0s–12h)
Dead-letter queue (DLQ)	Ordinary SQS queue that receives messages the source queue failed to process after maxReceiveCount attempts (set in its redrive policy)
Long polling	ReceiveMessage queries all SQS servers and waits (WaitTimeSeconds 1–20) to eliminate empty responses; short polling (=0) samples a subset
SNS topic	Logical access point and communication channel; publishers post to it and SNS delivers to every subscribed endpoint
Filter policy	JSON object of attributes on a subscription; SNS delivers a message only when its attributes match
Shard	Capacity unit of a Kinesis stream — writes up to 1,000 records/s or 1 MB/s; reads up to 5 tx/s or 2 MB/s
Partition key	Value that groups Kinesis data records by shard within a stream

03 THREE AWS MESSAGING SERVICES AT A GLANCE

match the pattern to the use case

SERVICE	PATTERN	HOW CONSUMERS GET MESSAGES	BEST FOR
Amazon SQS	Message queue	Consumers **poll**; each message processed once, then **deleted**	Decoupling components; buffering background work
Amazon SNS	Pub/sub	Messages **pushed** to every subscriber; no polling	Broadcasting one event to many consumers (fanout)
Kinesis Data Streams	Data stream	Many consumers **read the same** records; none deleted	High-volume, near-real-time aggregation and analytics

04 STANDARD VS FIFO SQS QUEUES

FIFO only when order or dedup is critical

ATTRIBUTE	STANDARD (DEFAULT)	FIFO
Ordering	Best-effort — some messages out of order	Order strictly preserved
Delivery	At-least-once (duplicates possible)	Exactly-once processing (no duplicates)
Throughput	Nearly unlimited TPS	More limited TPS
Use when	Media encoding, credit-card validation volume, non-order-dependent writes	Deposit before withdrawal, no double charge, register before enroll

05 SQS MESSAGE LIFECYCLE

receiving is not deleting

01 Producer `SendMessage` → stored redundantly across SQS servers → **02** Consumer `ReceiveMessage` → **visibility timeout** starts (default 30s) → **03** Consumer processes the message → **04** Consumer `DeleteMessage` before the timeout expires → **05** Not deleted in time → message reappears; after **maxReceiveCount** → dead-letter queue

06 MESSAGE QUEUE (SQS) VS PUB/SUB (SNS)

polling model vs push model

MESSAGE QUEUE — AMAZON SQS

one message, one successful consumer

- Producer adds messages; consumers **poll** the queue
- Each message is processed by one consumer, then **deleted**
- Interim buffer that absorbs load and decouples components
- Standard or FIFO delivery semantics

PUB/SUB — AMAZON SNS

one message, many subscribers

- Publisher sends to a **topic**; SNS **pushes** to all subscribers
- Every subscriber receives its own copy of the message
- Subscribers never poll; a filter policy narrows what they receive
- A2A (SQS, Lambda, Firehose, HTTP/S) and A2P (email, SMS, push)

07 EXAM TRAPS

where the knowledge check gets you

- × “Standard SQS queues guarantee order.” — No. Standard queues give **best-effort ordering** and **at-least-once** delivery, so messages may arrive out of order or more than once. Only **FIFO** guarantees order and exactly-once processing.
- × “Receiving a message deletes it.” — Receiving only starts the **visibility timeout**. The consumer must explicitly call `DeleteMessage`; if it doesn't before the timeout expires, the message becomes visible again and can be processed twice.
- × “SNS subscribers poll the topic.” — Pub/sub is **push**: SNS delivers a copy to every subscriber. Polling is the queue (SQS) and stream (Kinesis) model, not SNS.
- × “Stream consumers delete records after processing.” — Kinesis consumers **do not delete**; each keeps its own pointer, and multiple consumers can read the same records for different purposes.
- × “You must use SQS/SNS resource policies instead of IAM.” — For same-account access either works. You **must** use resource-based (SQS or SNS) policies to grant access to **other AWS accounts**.
- × “Long polling means a longer visibility timeout.” — Different settings. Long polling is `WaitTimeSeconds` (1–20) waiting for a message; the **visibility timeout** (0s–12h) hides a received message from other consumers.
- × “A dead-letter queue is a special queue type.” — A DLQ is an **ordinary SQS queue** named in the source queue's redrive policy; messages move there after **maxReceiveCount** failed processing attempts.
- × “Enabling SSE encrypts the messages already on the queue.” — Encryption applies **from that point forward** — messages already on the queue stay unencrypted; only messages arriving afterward are encrypted.

08 SELF-QUIZ

answers are upside-down below

Q1 A tightly coupled synchronous call chain sacrifices which single system property?

Q2 Which AWS service implements pub/sub messaging?

Q3 What is the default SQS visibility timeout, and its allowed range?

Q4 Which SQS queue type provides strict ordering and exactly-once processing?

Q5 After processing a received message, what must a consumer do so no one reprocesses it?

Q6 Which polling mode queries all SQS servers and waits for a message to cut empty responses?

Q7 In an SNS fanout pattern, what typically subscribes to the topic for parallel processing?

Q8 What JSON construct on a subscription makes a subscriber receive only matching messages?

Q9 In Kinesis Data Streams, what determines which shard a data record is written to?

Q10 Name the two Kinesis services that simplify use cases — one delivers to data stores, one runs SQL analytics.

ANSWER KEY (rotate the page)

Q1 fault tolerance — one component's failure fails the whole system **Q2** Amazon SNS **Q3** 30 seconds; 0 seconds to 12 hours **Q4** FIFO **Q5** call DeleteMessage using the message's receipt handle **Q6** long polling (WaitTimeSeconds 1–20) **Q7** multiple SQS queues **Q8** a filter policy **Q9** the partition key **Q10** Kinesis Data Firehose (delivery) and Kinesis Data Analytics (SQL analytics)