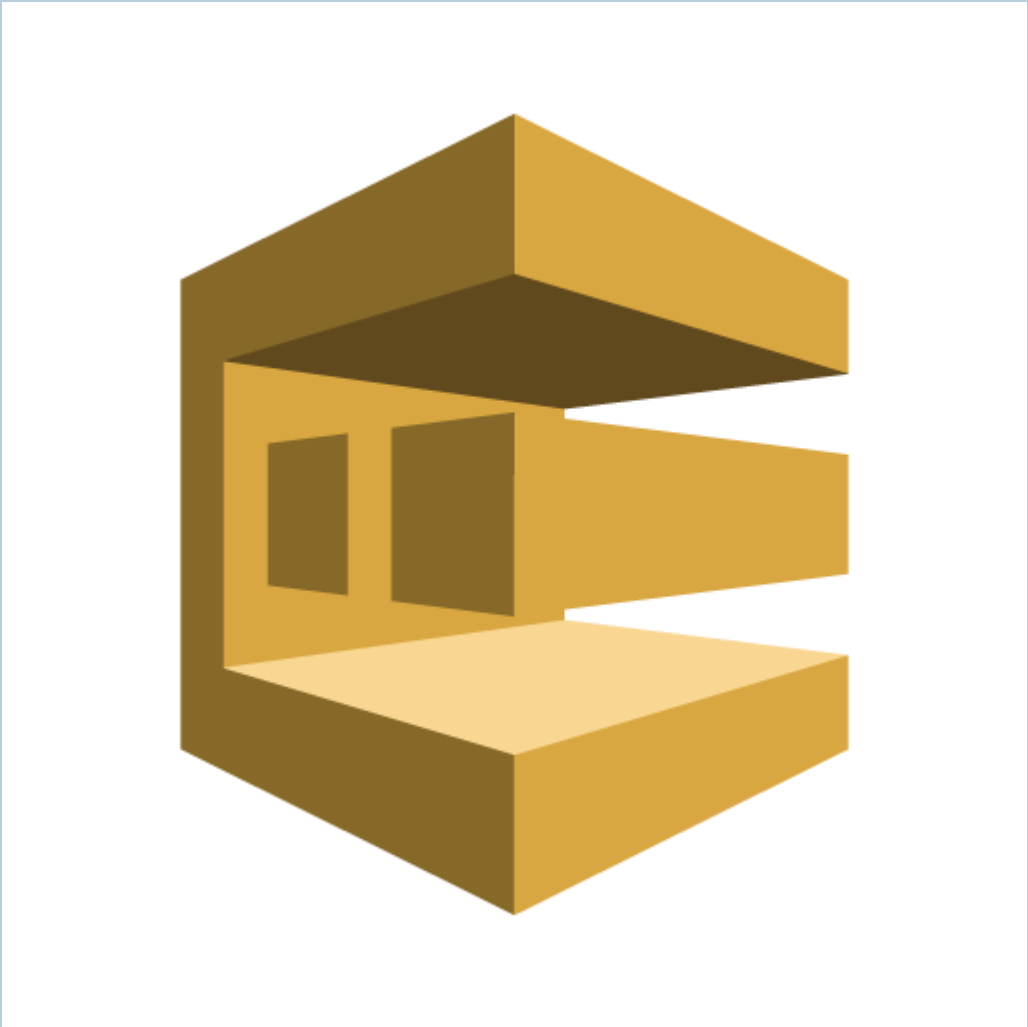
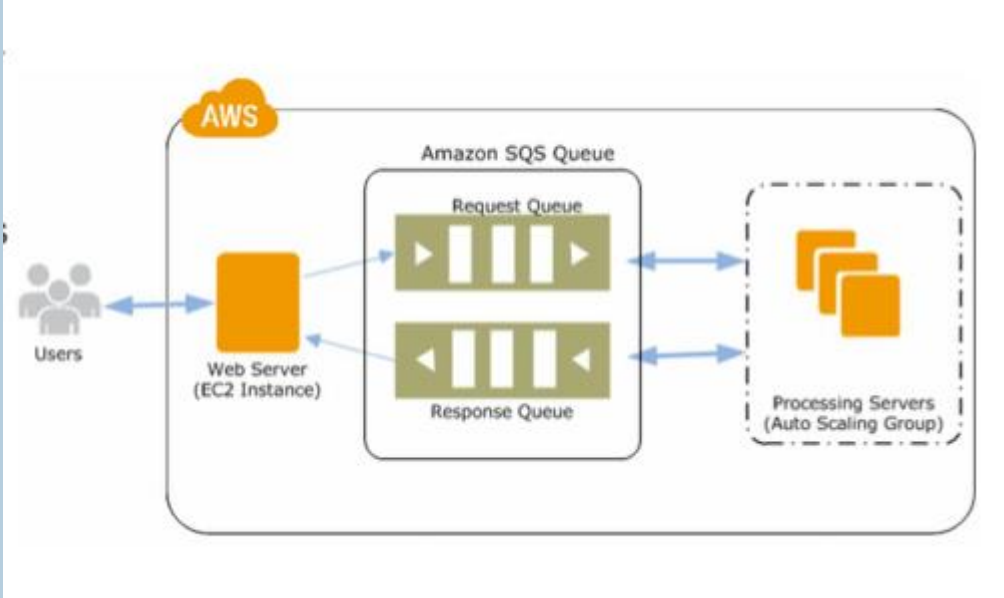
Uses of Amazon SQS



What is Amazon SQS?

Amazon Simple Queue Service (SQS) is a fully managed message queuing service in AWS cloud that enables software applications to implement queueing services in microservices, distributed systems, and server-less applications. SQL is scalable. Using SQS, you can send, store, and receive messages at any volume, without losing messages or requiring other services to be available.

How does Amazon SQS work?



Distributed queues

There are three main parts in a distributed messaging system: the components of your distributed system, your queue (distributed on Amazon SQS servers), and the messages in the queue.

In the following scenario, your system has several *producers* (components that send messages to the queue) and *consumers* (components that receive messages from the queue). The queue (which holds messages A through E) redundantly stores the messages across multiple Amazon SQS servers

Amazon SQS dead-letter queues

Amazon SQS supports *dead-letter queues*, which other queues (*source queues*) can target for messages that can't be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system because they let you isolate problematic messages to determine why their processing doesn't succeed. For information about creating a queue and configuring a dead-letter queue for it using the Amazon SQS console, see Configuring a dead-letter queue (console).

Key features of SQS

SQS is a cloud service and can be used by any type of software, application, or other service. SQS works at its own independent service in the cloud. A software connects with SQS using a connection by passing the credentials and queue names. SQL also allows applications to create and delete custom queues.

- At-Least-Once Delivery — A message in the queue is delivered at least once. Message delivery is guaranteed, no message is lost.[Text Wrapping Break]
- Multiple components can work on a single queue. SQS uses a lock mechanism, if one component is using a message, it is made hidden to other components. Upon successful processing, message is deleted from the queue. If the message processing fails, it stays in the queue and is made visible to all the components. This feature is called Visibility Timeout.

- There are two types of queues — Standard and FIFO. In standard queue the messages are picked up randomly. It might not be in the order it entered the queue while FIFO queue uses first-in-first-out, it ensures the order.
- For the messages that cannot be processed are kept in dead-letter queue.
- Billing is done based on the number of requests to the queue. SQS is a good service to be used for applications to increase efficiency, reliability and performance.

Benefits of SQS

Here is a list of key benefits of SQS.

SQS is easy to setup without any overhead

Implementing SQS, there is no upfront cost or administrative cost. It is all configurable in AWS cloud and APIs are provide to manage queues programmatically. AWS's pay-as-you-go model allows you to pay for what you use only.

SQL is reliable

Use Amazon SQS to transmit any volume of data, at any level of throughput, without losing messages or requiring other services to be available. SQS lets you decouple application components so that they run and fail independently, increasing the overall fault tolerance of the system. Multiple copies of every message

are stored redundantly across multiple availability zones so that they are available whenever needed.

SQS is secure

You can use Amazon SQS to exchange sensitive data between applications using server-side encryption (SSE) to encrypt each message body. Amazon SQS SSE integration with AWS Key Management Service (KMS) allows you to centrally manage the keys that protect SQS messages along with keys that protect your other AWS resources. AWS KMS logs every use of your encryption keys to AWS CloudTrail to help meet your regulatory and compliance needs.

SQS is scalable and and cost-effective

Amazon SQS leverages the AWS cloud to dynamically scale based on demand. SQS scales elastically with your application so you don't have to worry about capacity planning and pre-provisioning. There is no limit to the number of messages per queue, and standard queues provide nearly unlimited throughput. Costs are based on usage which provides significant cost saving versus the "always-on" model of self-managed messaging middleware.

The benefits of using SQS

For Serverless developers, using SQS generally provides a wealth of benefits, which you can read about below.

Pay for what you use When using SQS, you only get charged for the messages you read and write (see the details in the Pricing section). There aren't any recurring or base fees.

Ease of setup Since SQS is a managed service, so you don't need to set up any infrastructure to start using SQS. You can simply use the API to read and write messages, or use the SQS <-> Lambda integration.

Options for Standard and FIFO queues When creating an SQS queue, you can choose between a standard queue and a FIFO queue out of the box. Both of these queue types can be useful for different purposes.

Automatic deduplication for FIFO queues Deduplication is important when using queues, and for FIFO queues SQS will do the work to remove any duplicate messages for you. This makes FIFO queues on SQS suitable for tasks where it's critical to have each task done exactly once.

A separate queue for unprocessed messages This feature of SQS is useful for debugging. All messages that couldn't be processed are sent into a "dead-letter" queue where you can inspect them. This queue has all the usual integrations enabled, so you can subscribe to it using an AWS Lambda event, for example, to send a notification when an item can't be processed.

- **Work Queues:** Decouple components of a distributed application that may not all process the same amount of work simultaneously.
- **Buffer and Batch Operations:** Add scalability and reliability to the architecture and smooth out temporary volume spikes without losing messages or increasing latency
- **Request Offloading:** Move slow operations off of interactive request paths by enqueueing the request.
- **Fan-out:** Combine SQS with SNS to send identical copies of a message to multiple queues in parallel for simultaneous processing.

Auto Scaling: SQS queues can be used to determine the load on an application, and combined with Auto Scaling, the EC2 instances can be scaled in or out, depending on the volume of traffic.