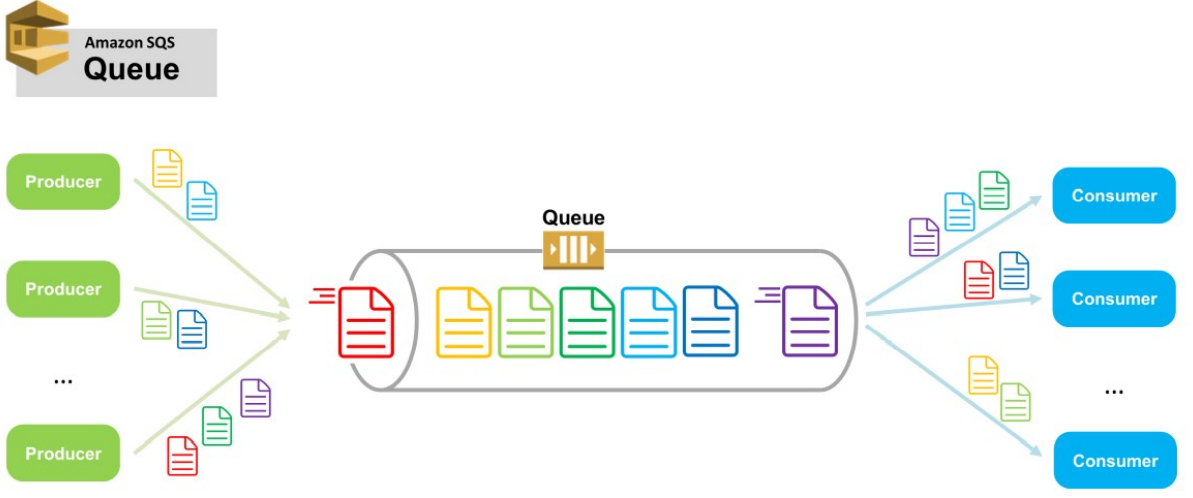


SIMPLE QUEUE SERVICE

* SQS is **pull based system**
* SNS is **push based system**
* SQS is the service which was firstly launched by AWS
* SQS is temporary repository for the messages to get processed by consumers
* Once the consumer consumes the message, the message will be deleted from the queue
* SQS is a message queue used to store messages while waiting for computer to process them
* **We cannot change the queue type after you create it and you can't convert an existing standard queue into a FIFO queue and vice versa.**
* **Services 🡪 Application Integration 🡪 Simple Queue Service**
* **Using Amazon SQS, you can decouple the components of an application so they run independently, easing message management between components.**
* **Any component of a distributed application can store messages in a fail-safe**

**queue.**

* **Messages can contain up to 256 KB(max 2GB) of text in any format. Any**
* **component can later retrieve the messages programmatically using the Amazon SQS API**
* **The queue acts as a buffer between the component producing and saving data, and the component receiving the data for processing.**
* **This means the queue resolves issues that arise if the producer is producing work faster than the consumer can process it, or if the producer or consumer are only intermittently connected to the network.**
* **A queue can be created in any region**
* **Each 64KB ‘chunk’ of payload is billed as 1 request.**
* **Messages can be sent and read simultaneously.**



Benefits and features

1.**Highly scalable standard and FIFO queues**

* Queue scale elastically with your application, nearly unlimited throughputs and limit to the number of messages per queue in standard queues.

2.**Durablity and availability**

* Your queues are distributed on multiple server’s redundant infrastructure provides highly concurrent access to messages

3**.Security**

* Protection in transit and at rest. Transmit sensitive data in encrypted queue and send messages in **virtual private cloud**

4.**Batching**

* send receive or delete message in bathes of up to 10 messages or 256KB to save cost.

Queue Attributes:

Types of Queues

* Standard Queue
* FIFO (First-In-First-Out)

WebServer-1

Route53 🡪 LB WebServer-2 SQS 🡪 Application 🡪 DB

WebServer-3

1.Standard Queue

* Standard Queues are default Queues
* Supports unlimited number of transactions per seconds (TPS) per action
* Duplicate messages are possible, that meaning, single message can be delivered twice.
* **Standard queue provides best-effort ordering which ensures that messages are generally delivered in the same order as they're sent, this is not guaranteed**
* **Standard queues guarantee that a message is delivered at least once.**
* **However, occasionally more than one copy of a message might be delivered out of order.**

2.FIFO (First-In-First-Out)

* FIFO queues have ALL the capabilities of the standard queue
* FIFO (First-in-First-Out) queues are designed to enhance messaging between application when the order of operations and events is critical, or where duplicates can't be tolerated.
* To create a FIFO queue, its queue name must end with ***.fifo(suffix)***
* 300 TPS (Transaction Per Seconds) with high throughput
* Duplicate messages are not allowed
* **Messages are delivered exactly once**
* it will follow strict order-FIFO
* FIFO queues also supports messages groups that allowed multiple ordered messages groups within a single queue.
* FIFO queue also provide exactly-once processing but have a limited number of transactions per seconds (TPS):
  1. FIFO queues support **up to 3,000 messages per** seconds with batching
  2. FIFO queues support **up to 300 messages per** second, per action

Visibility Timeout Out

* It is a time period in which only one consumer can access the message in the queue and during this visibility timeout period no other consumer can access the same message so it can be anything 0 seconds **up to 12 hours.**
* Once the request is pulled from the SQS, the request is SQS will not be deleted, it be invisible for 30 seconds
* Within 30 seconds application server should process the request
* Within 30 seconds, if application server processes the request, it will be deleted from the SQS
* Within 30 seconds, if application server unable to process the request.it will be visible again in SQS
* If Application server takes more than 30 seconds, we can increase the visibility time out up to 12hours
* Default visibility timeout is 30 seconds
* Increase if your task takes > 30 seconds
* Minimum = 0 seconds and Maximum = 12hours
* **Message retention period = default 4 days, Minimum = 1min and Maximum 14days**
* Messages can contain up 256KB of text in any format
* SQS guarantee that your messages will be processed at least once

Producers

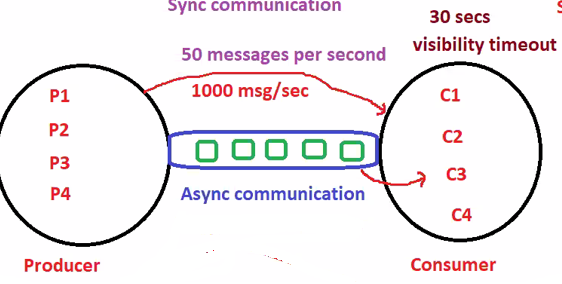
* Messages are sent from applications, microservices and other AWS services

SQS

* Queues store messages and wait for consumers to pull

Consumers

* Messages are processed by applications, Lambda functions, EC2 Instances and other AWS services



Message retention period

* It is a maximum amount of time that a message will be stored in the SQS queue before SQS deletes it now let's say no consumer pulls the messages for 4 days then after that SQS will automatically delete this message in the queue

**it can be anything from 1 minute to 14days**

* Messages can be kept in the queue for 1 minute to 14 days

Delivery delay

* It is the amount of time to delay the first delivery of each message which means now if we give this **15 seconds** so only after 15 seconds the messages will reach the queue and the maximum message size depends on the producer who’s sending the message.
* **Should be between 0 seconds and 15 minutes**

Receive message wait time

* This is the maximum time that the consumer has to wait in order to pull message from the queue can be **anything 0 to 20 seconds**
* **Short polling = 0 seconds and long polling = 20 seconds**
* **The maximum amount of time that a long polling receive call will wait for a message to become available before returning an empty response**
* **Long polling doesn’t return a response until a message arrives in the message queue or the long poll timeout**

Access policy

* Access policy will define who can access your queue there are two types of method one is the basic and the other one is the advance
* advance uses a json object to define an advanced access policy

1.Define who can send messages to queue

2.Define who can receive message from the queue

1.1.1) Only the queue owner

**(Only the owner of the queue can send messages to the queue)**

2.2.2) Only the specified AWS accounts, IAM users and roles

(**Only the specified AWS account IDs, IAM users and roles can send messages to the queue)**

Encryption

* Encryption is optional
* Before encrypt our sensitive data using server side encryption now if we enable this we can use server-side encryption with aws key management services which will allow you to centrally managed all the keys that will protect your SQS messages

Dead-letter queue

* If a message cannot be consumed successfully, we can send it to a **dead letter queue**
* Dead letter queues let you **isolate problematical message** to determine why they are failing so **for example** if one message in queue cannot be pulled by any of the consumer then you can send that message to the dead letter queue this message later can be examined **by any of the user or technician and can find what was the problem with that message**
* In order to enable dead letter queue, we should already have a queue if we have a queue, we can just select the queue name and proceed
* A dead-letter queue must be created first before it is designated as a dead-letter queue
* when a source queue is created, you can assign a dead-letter queue for that source queue.
* The dead-letter queue of a **FIFO queue** must also be a FIFO queue.

similarly, the dead-letter queue of a **standard queue** must also be a **standard queue**

* Both source queue and dead letter queue must in the **same AWS account** and region
* Retry policy will be 5 to 1000 maximum

Message retention & Dead-Letter Queues

When a message is moved to a dead-letter queue, the enqueue timestamp remain unchanged.

**Example**

1. if the retention period of the Dead-Letter Queue is 5 days
2. Message was in the source queue for 1 day before being moved to the Dead-Letter Queue
3. Message will be deleted from the Dead-Letter Queue after 4 days.

If is recommended that the message retention period of a dead-letter queue should be longer than the retention period of the original queue

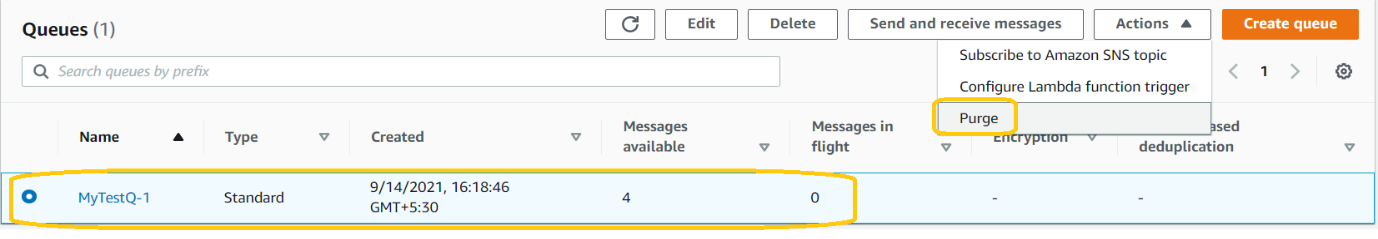
tag

* tag is a label assigned to the aws resources we can use tags to search and filter our resources or track our aws cost

Purge Queue

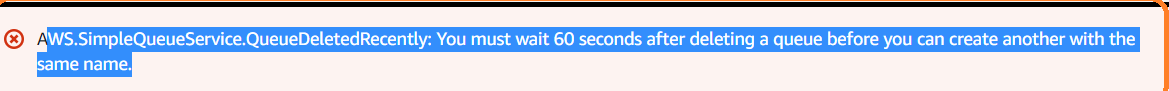
* Deletes **ALL** messages in the queue
* After you purge a queue, **you cannot retrieve any deleted messages**
* It takes up to 60 seconds to purge a queue
* Messages sent to the queue **BEFORE** you call Purge Queue might be received but are deleted within the next minute.
* Messages sent to the queue **AFTER** you call Purge Queue might be deleted while the queue is being purged.

Queue(name of the queue) 🡪 Actions 🡪 Purge (drop down)



Delete Queue

* Delete the selected queue
* After you delete a queue, you **cannot** redrive any messages that were **present in the queue**
* it takes **up 60 seconds** to delete a queue
* Any requests received **during the deletion process will succeed**,
* **Example:** Sending a message. After 60 seconds queue and **ALL** messages will be deleted
* you must **wait 60 seconds to create a queue** with the **same name** **same name** after it is deleted



* A queue can be deleted even when it is no empty. If you want to delete ONLY the messages in the queue but not the queue itself, use PURGE queue

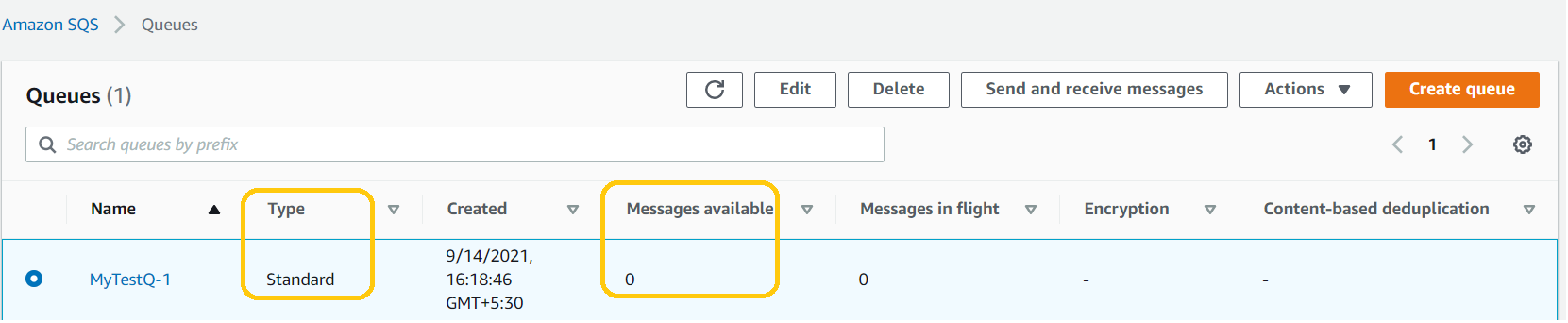
**SQS – LAB**

1.Create SQS queue with standard type

Services 🡪 Application Integration 🡪 Simple Storage Queue

Click on **create queue**

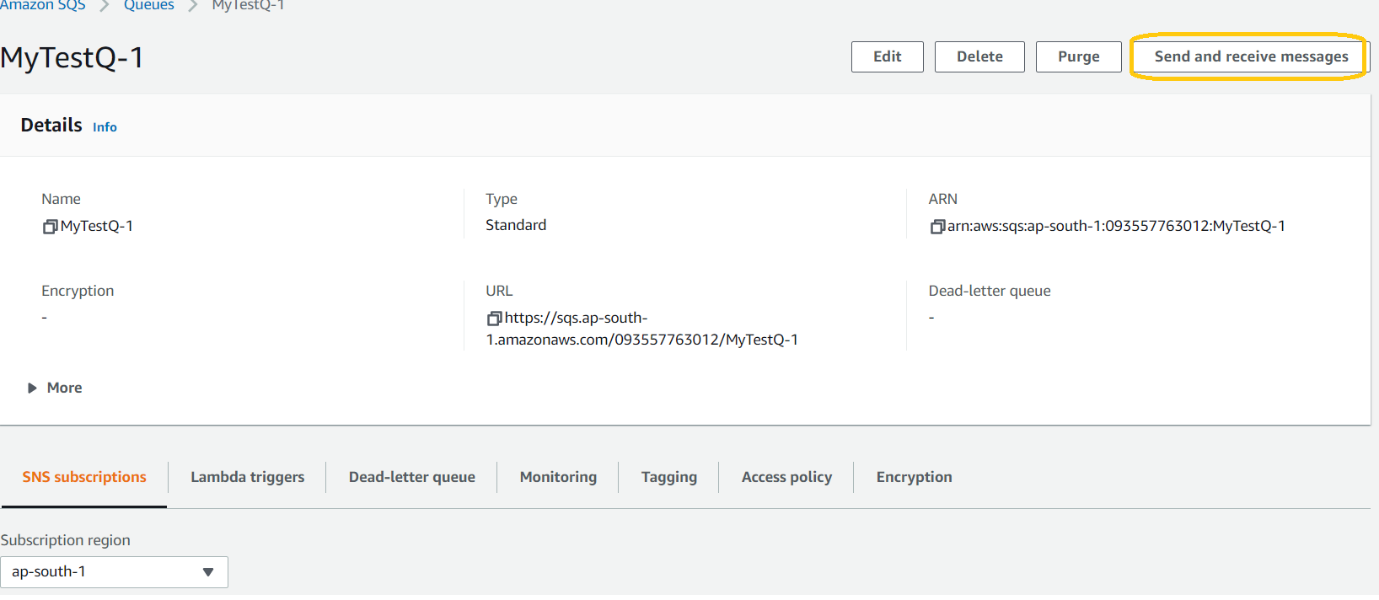
Provide basic details and click on **Create queue** button.



**Add messages to the queue**

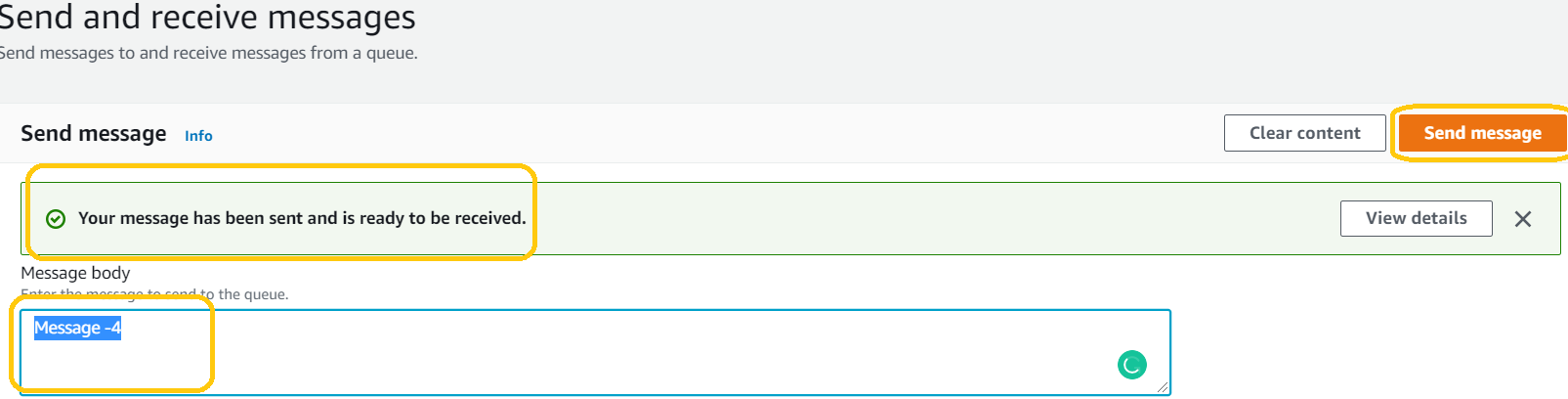
**Amazon SQL --> Queue --> MyTestQ-1 (click on Queue name)**

click on **Send and receive messages**



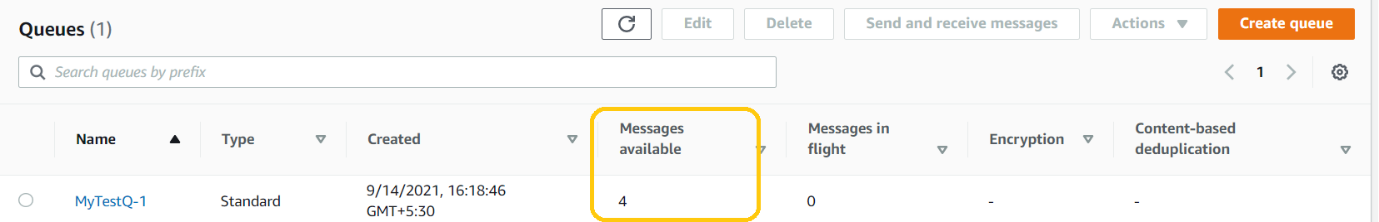
Enter message in body and click on button **send message**

For testing purpose send 4 messages with different message body



Go to queue page and refresh page

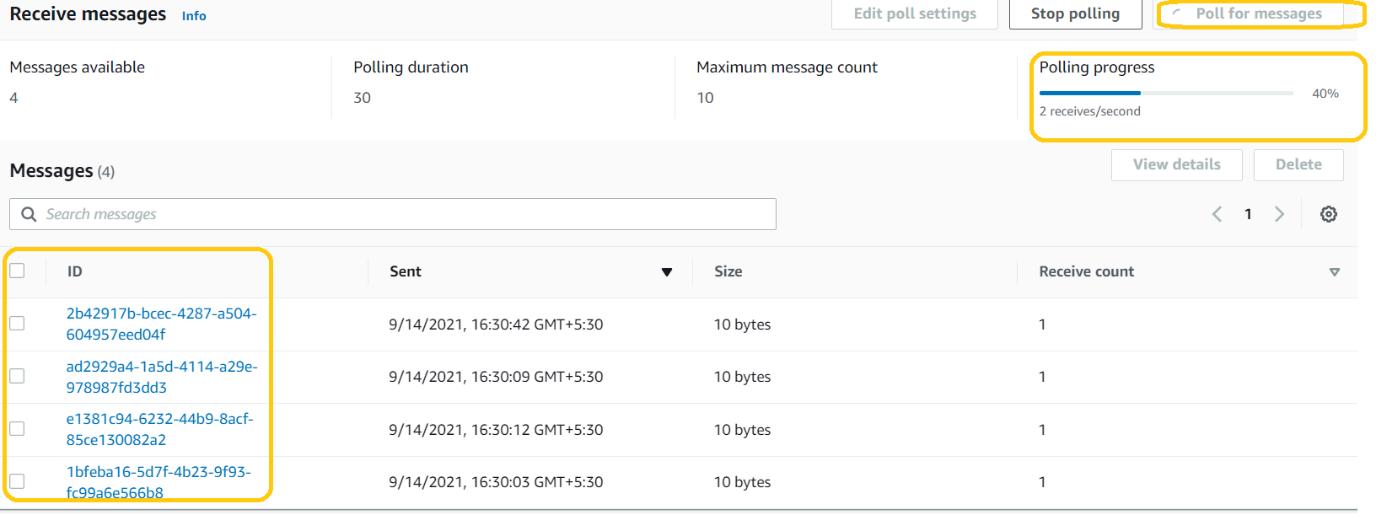
We can see messages list under **message available sectio**n



To receive messages, need to fallow below steps

Queue name 🡪 **click on send and receive messages**

Under Receive message section click on **Poll for message** button we will get list of messages which are in queue.



Note:

* Delete All messages under receive message section
* Delete all messages under queue section
* Finally, Delete queue