Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components.

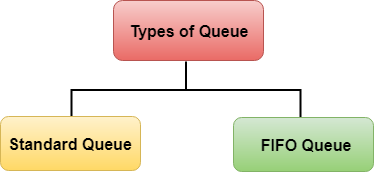
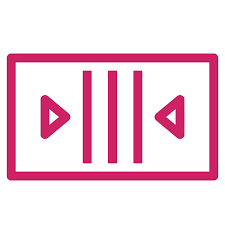
**SQS**

It is fully managed service and complicated service.

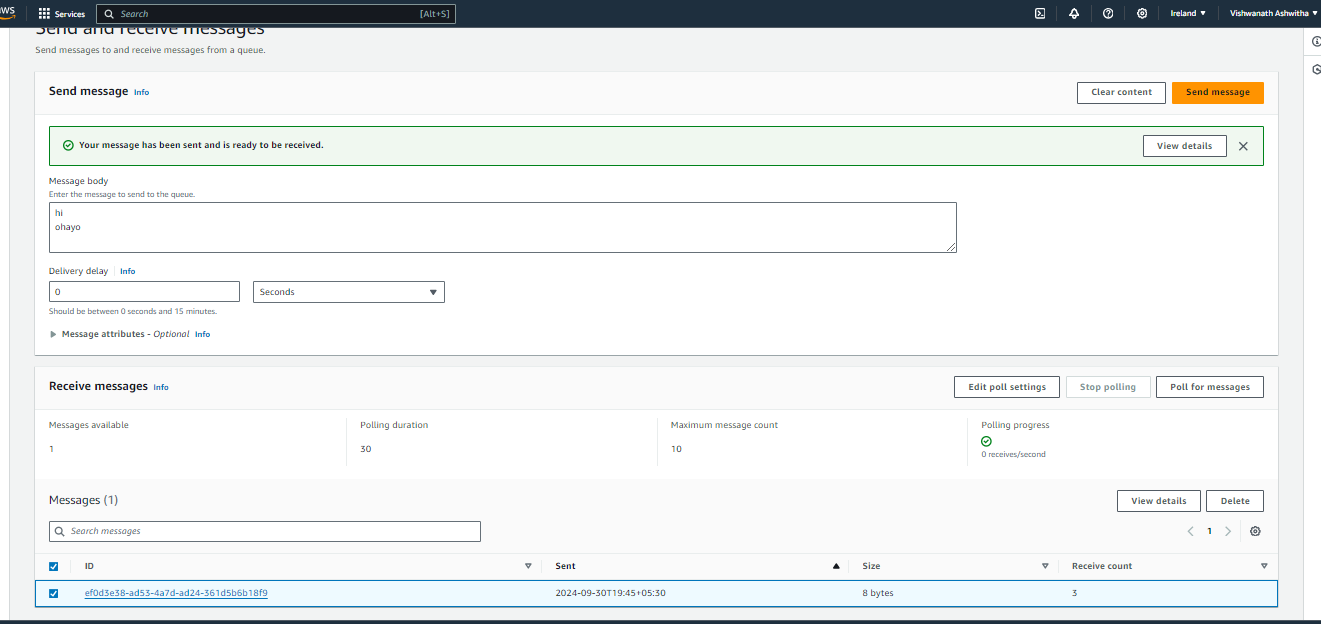
It was offering in AWS almost 10 years old.

There are two types of Queue:

* Standard Queues (default)
* FIFO Queues (First-In-First-Out)

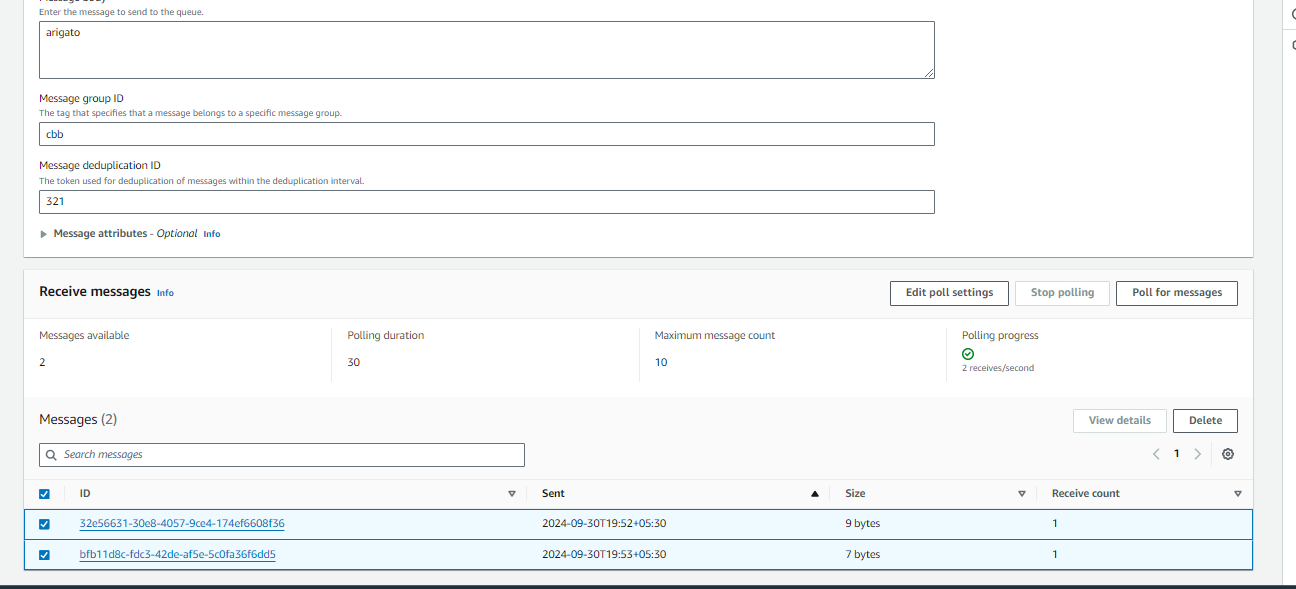
 

* There will be the producer (send message to the SQS) and consumer (poll messages)
* The message will be lost by default within 4 to 14 days.
* Read and delete messages after retention period.
* Low latency (i.e., responds within less than 10 seconds on publish and receive message)
* The message shall be less than 256 KB.
* Can have duplicate messages.
* SQS- produce messages using SDK (Send Message API)
* SQS – consuming messages: consumers running on EC2 servers or lambda pools SQS for messages
* Receive up to 10 messages at a time
* Delete message using delete API



SQS – FIFO Queue:

* First in First Out
* Limited through put: 300 messages
* Exactly one send capability by removing duplicates
* Messages are ordered / processed by consumer
* Name ends with .fifo



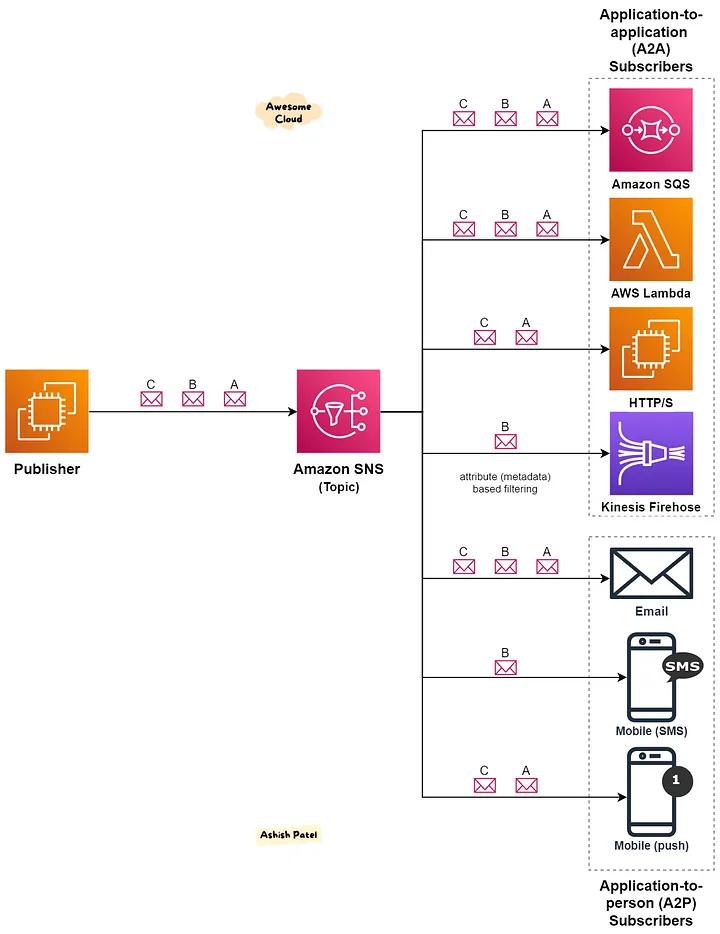
**SNS**

* Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).
* Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).

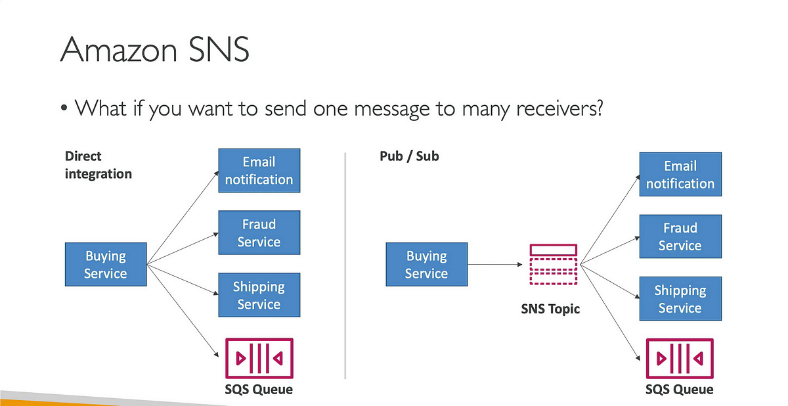


**Pub/sub service**

* Only event producer send messages to SNS topics
* Every subscriber will get every message
* Up to 12, 500, 000 subscriptions per topic
* Up to 100, 000 topics limit



How to publish?

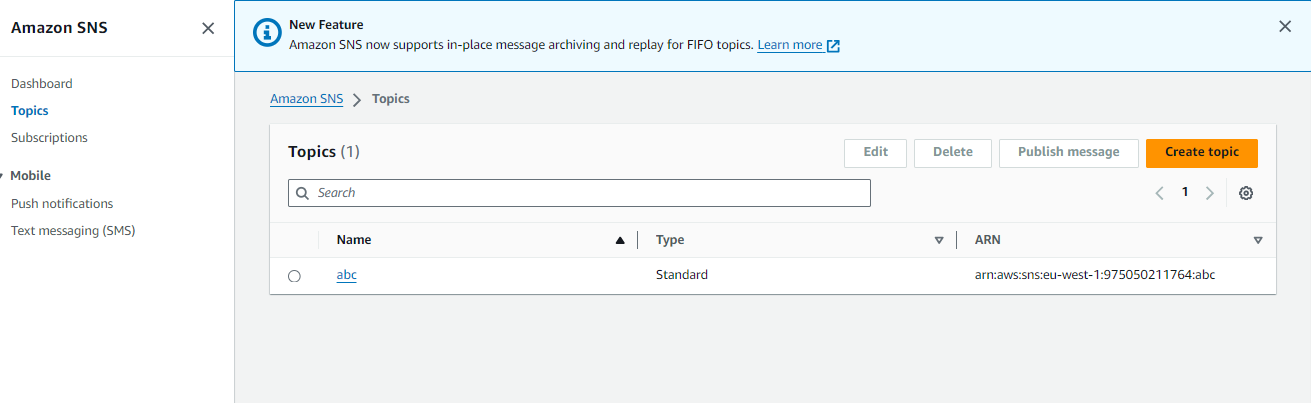


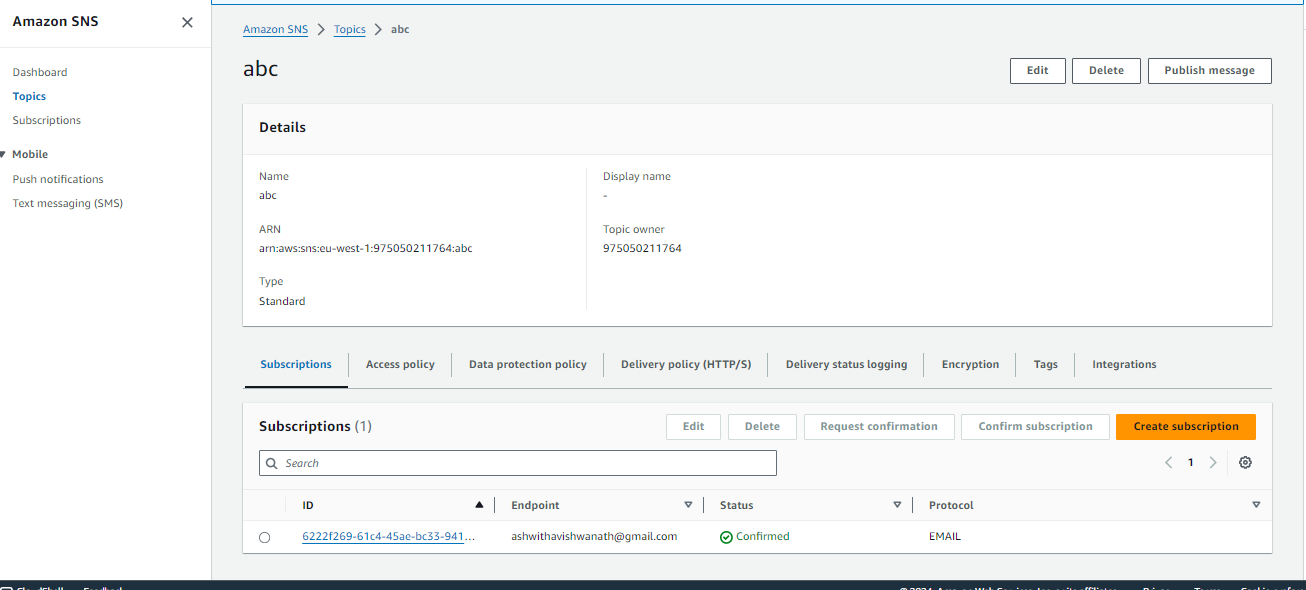
**Topic publish**: (using SDK)

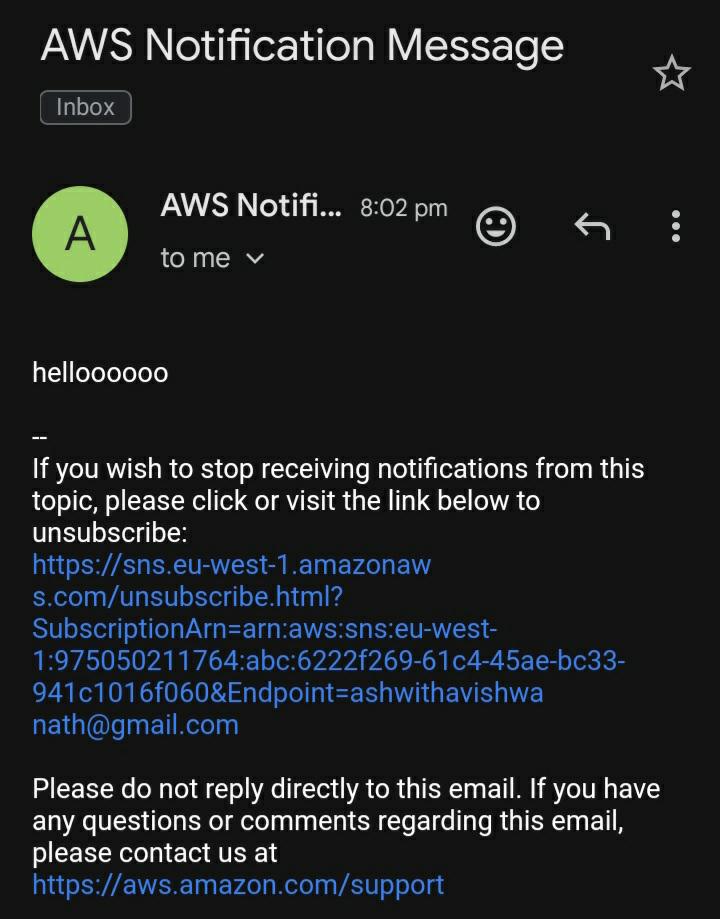
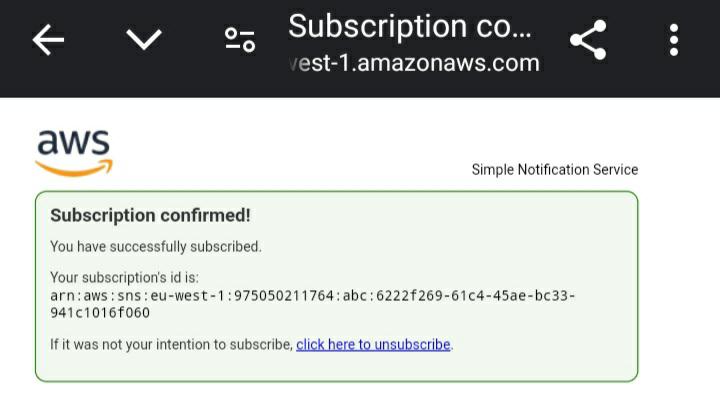
Create topic, create subscription, publish to topic

**Direct publish**: (for mobile apps SDK)

Create platform applications, create platform end point, publish to the platform endpoint





Amazon CloudWatch is a service that monitors applications, responds to performance changes, optimizes resource use, and provides insights into operational health.

**CLOUD WATCH**

**Cloud watch alarm**:

AWS CloudWatch Alarms is a monitoring tool that helps you to create alarm. This enables you to be alerted immediately when issues occur in your infrastructure

Trigger notifications for any metric

Alarms states: ok, insufficient-data, in-alarm

Period: length of time in seconds to evaluate metric

High resolution custom metrics, multiple of 60 seconds

**Alarm types**:

**Metric**- A *metric alarm* watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics.

**Composite**- A *composite alarm* includes a rule expression that takes into account the alarm states of other alarms that you have created

Can use and or conditions, help to reduce the alarm noise

To test alarm and notifications, set the alarm state to alarm using cli



Create EC2 instance with t2. micro and launch

Create alarm, choose metric, select pre-instance-metrics, CPU utilization -select metric

Statistics-average, period-5min, threshold type-statistic, CPU utilization- greater,

Threshold value-95, data-3,3

Next

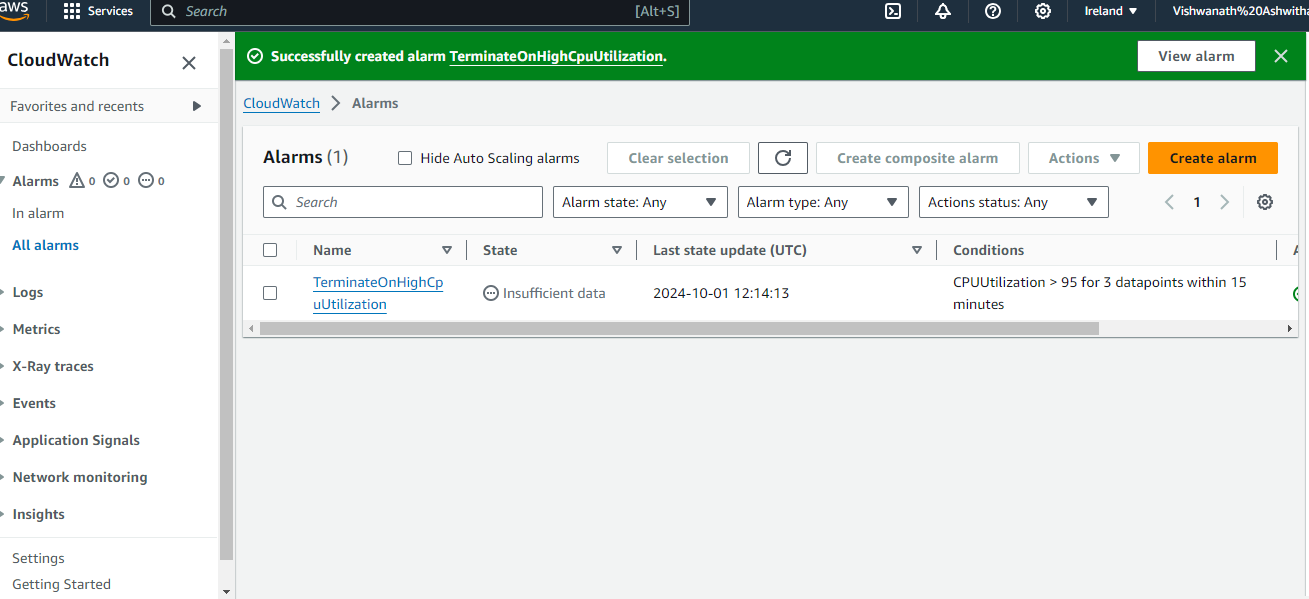
Ec2-action, in alarm, terminate instance

Next

Alarm name

Next

Create alarm (which is in insufficient state)

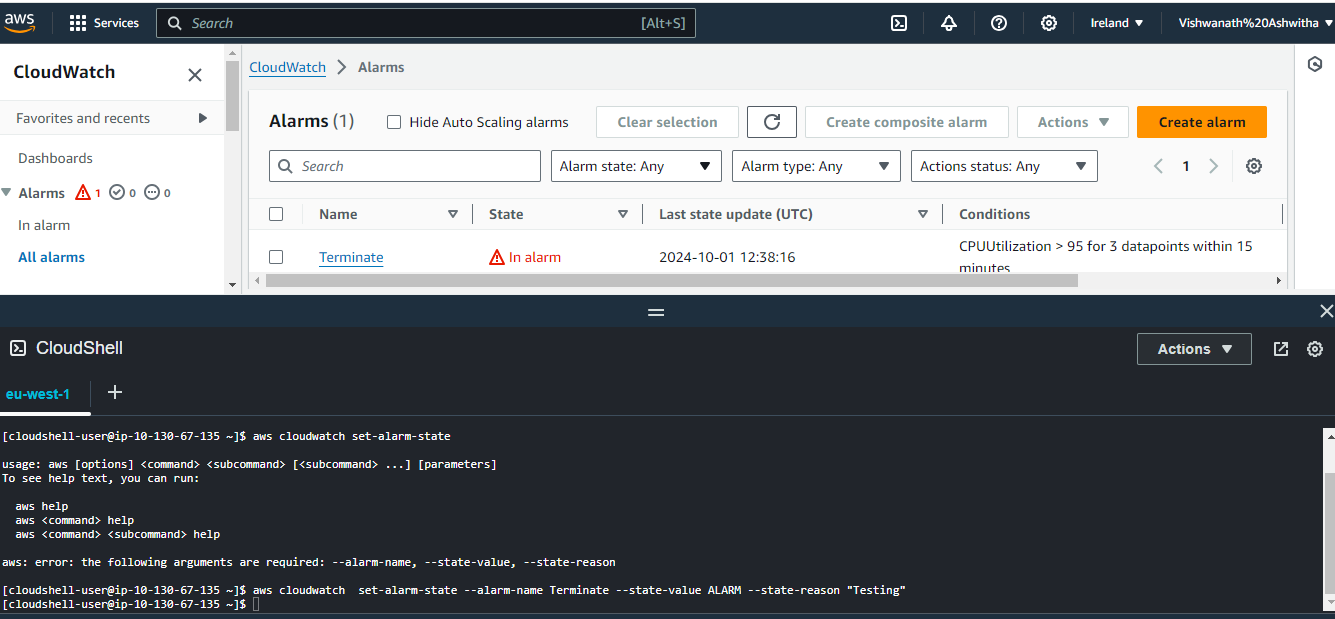


To test alarms and notifications set the alarm state from insufficient- data to in- alarm state using cli as cloud watch set-alarm-state – alarm-name “alarm name” --state -value ALARM --state -region “testing”

Open AWS CLI

Then the alarm sets as in- alarm state

To see what would happen if this alarm went in to breach phase



Now get into EC2 instance we see that it is going to terminate automatically as there was alarm that was triggered on top of instance as we setup alarm to terminate when reaches 95% of CPU utilisation

