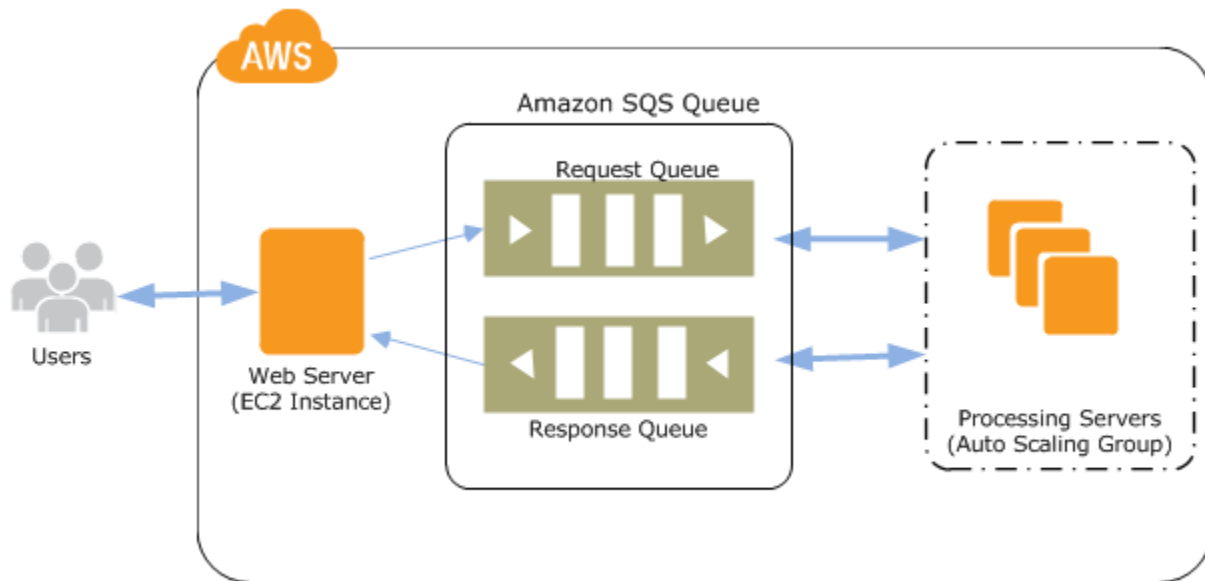


## Amazon Simple Queue Service (Amazon SQS)

Amazon SQS is a fast, reliable, scalable, and fully managed message queuing service. Amazon SQS makes it simple and cost effective to decouple the components of a cloud application.

- Amazon SQS is a web service that gives you access to a message queue that can be used to store messages while waiting for a computer to process them.
- A queue is a temporary repository for messages that are awaiting processing.
- An Amazon SQS queue is basically a buffer between the application components that receive data and those components that process the data in your system.
- Messages can contain up to 256 KB of text in any format.
- Amazon SQS ensures delivery of each message at least once, and supports multiple readers and writers interacting with the same queue.
- Message Retention period is 14 Days
- Amazon SQS is engineered to provide "**at least once**" delivery of all messages in its queues. Although most of the time each message will be delivered to your application exactly once.
- A single queue can be used simultaneously by many distributed application components, with no need for those components to coordinate with each other to share the queue.
- Maximum message size 256kb now available
- AWS will Billed as Chunks, Each Chunk size is 64kb, That means a 256kb message will be 4 x 64kb "chunks".
- First 1 million Amazon SQS Requests per month are free
- \$0.50 per 1 million Amazon SQS Requests per month thereafter (\$0.00000050 per SQS Request)
- A single request can have from 1 to 10 messages, up to a maximum total payload of 256KB.
- Each 64KB 'chunk' of payload is billed as 1 request. For example, a single API call with a 256KB payload will be billed as four requests.

For example, suppose that you have a web app that receives orders from customers. The app runs on EC2 instances in an Auto Scaling group that is configured to handle a typical number of orders. The app places the orders in an Amazon SQS queue until they are picked up for processing, processes the orders, and then sends the processed orders back to the customer. The following diagram illustrates the architecture of this example.



Amazon SQS is a distributed queue system that enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component.

Using Amazon SQS, you can store application messages on reliable and scalable infrastructure, enabling you to move data between distributed components to perform different tasks as needed.

Amazon SQS ensures delivery of each message at least once and supports multiple readers and writers interacting with the same queue. A single queue can be used simultaneously by many distributed application components, with no need for those components to coordinate with one another to share the queue. Although most of the time each message will be delivered to your application exactly once, you should design your system to be idempotent

SQL service does not guarantee First In, First Out (FIFO) delivery of messages.

Amazon SQS supports up to 12 hours' maximum visibility timeout.

When creating a new queue, you must provide a queue name that is unique within the scope of all of your queues. Amazon SQS assigns each queue an identifier called a queue URL, which includes the queue name and other components that Amazon SQS determines. Whenever you want to perform an action on a queue, you must provide its queue URL.

- TO create a Queue, Navigate to “Messaging” section and select the “Simple Queue Service”.
- Here is the default values we are getting with the Queue.

## Create New Queue

### What do you want to name your queue?

Queue Name ⓘ

Type the queue name.

Region ⓘ Asia Pacific (Mumbai)

For more information, see the [Amazon SQS FAQs](#) and the [Amazon SQS Developer Guide](#).

You can change these default parameters.

#### Queue Attributes

Default Visibility Timeout ⓘ  seconds ▾ Value must be between 0 seconds and 12 hours.

Message Retention Period ⓘ  days ▾ Value must be between 1 minute and 14 days.

Maximum Message Size ⓘ  KB Value must be between 1 and 256 KB.

Delivery Delay ⓘ  seconds ▾ Value must be between 0 seconds and 15 minutes.

Receive Message Wait Time ⓘ  seconds Value must be between 0 and 20 seconds.

## Amazon Simple Workflow Service (Amazon SWF)

Amazon Simple Workflow Service (Amazon SWF) is a web service that makes it easy to coordinate work across distributed application components.

Amazon SWF enables applications for a range of use cases, including media processing, web application back-ends, business process workflows, and analytics pipelines, to be designed as a coordination of tasks.

Amazon SWF makes it easy to build applications that coordinate work across distributed components. In Amazon SWF, a task represents a logical unit of work that is performed by a component of your application.

Amazon SWF gives you full control over implementing and coordinating tasks without worrying about underlying complexities such as tracking their progress and maintaining their state.

We have three SWF Actors:

- Workflow Starters - An application that can initiate (start) a workflow. Could be your e-commerce website when placing an order.
- Deciders - Control the flow of activity tasks in a workflow execution. If something has finished in a workflow (or fails) a Decider decides what to do next.
- Activity Workers - Carry out the activity tasks

## **Amazon Simple Notification Service (Amazon SNS)**

Amazon Simple Notification Service (Amazon SNS) is a web service that makes it easy to set up, operate, and send notifications from the cloud.

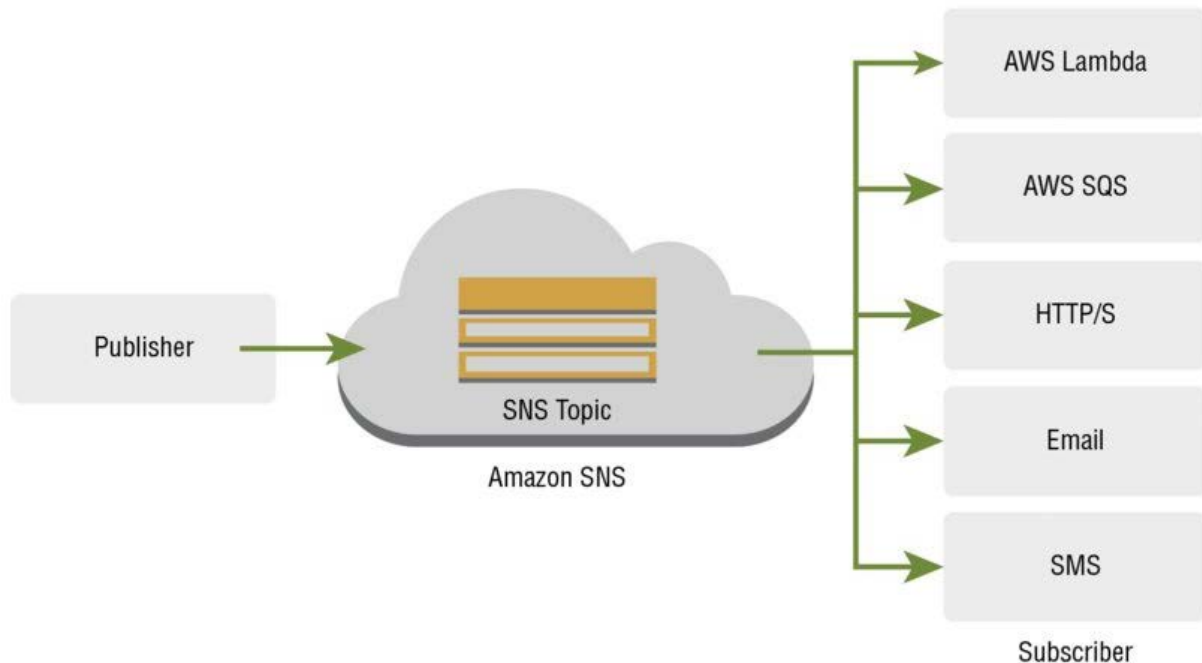
It provides developers with a highly scalable, flexible, and cost-effective capability to publish messages from an application and immediately deliver them to subscribers or other applications.

Push notifications to Apple, Google, Fire OS, and Windows devices, as well as Android devices in China with Baidu Cloud Push.

Amazon SNS consists of two types of clients: publishers and subscribers (sometimes known as producers and consumers).

- Publishers communicate to subscribers asynchronously by sending a message to a topic.
- A topic is simply a logical access point/communication channel that contains a list of subscribers and the methods used to communicate to them.
- When you send a message to a topic, it is automatically forwarded to each subscriber of that topic using the communication method configured for that subscriber.

Besides pushing cloud notifications directly to mobile devices, Amazon SNS can also deliver notifications by SMS text message or email, to Amazon Simple Queue Service (SQS) queues, or to any HTTP endpoint.



To prevent messages from being lost, all messages published to Amazon SNS are stored redundantly across multiple availability zones.

SNS allows you to group multiple recipients using topics. A topic is an "access point" for allowing recipients to dynamically subscribe for identical copies of the same notification.

### **Application and System Alerts**

Application and system alerts are SMS and/or email notifications that are triggered by predefined thresholds. For example, we can receive immediate notification when an event occurs, such as a specific change to your Auto Scaling group in AWS.

### **Push Email and Text Messaging**

Push email and text messaging are two ways to transmit messages to individuals or groups via email and/or SMS. For example, you can use Amazon SNS to push targeted news headlines to subscribers by email or SMS. Upon receiving the email or SMS text, interested readers can then choose to learn more by visiting a website or launching an application.

### **Mobile Push Notifications**

Mobile push notifications enable you to send messages directly to mobile applications. For example, you can use Amazon SNS for sending notifications to an application, indicating that an update is available. The notification message can include a link to download and install the update.

## **SNS Benefits**

- Instantaneous, push-based delivery (no polling)
- Simple APIs and easy integration with applications
- Flexible message delivery over multiple transport protocols
- Inexpensive, pay-as-you-go model with no up-front costs
- Web-based AWS Management Console offers the simplicity of a point-and-click interface

## **SNS vs SQS**

- Both Messaging Services in AWS
- SNS - Push
- SQS - Polls (Pulls)

## **Creating SNS Topic and Publishing:**

- Sign in to AWS account and Navigate to Mobile Services and then Amazon SNS to load the Amazon SNS dashboard.

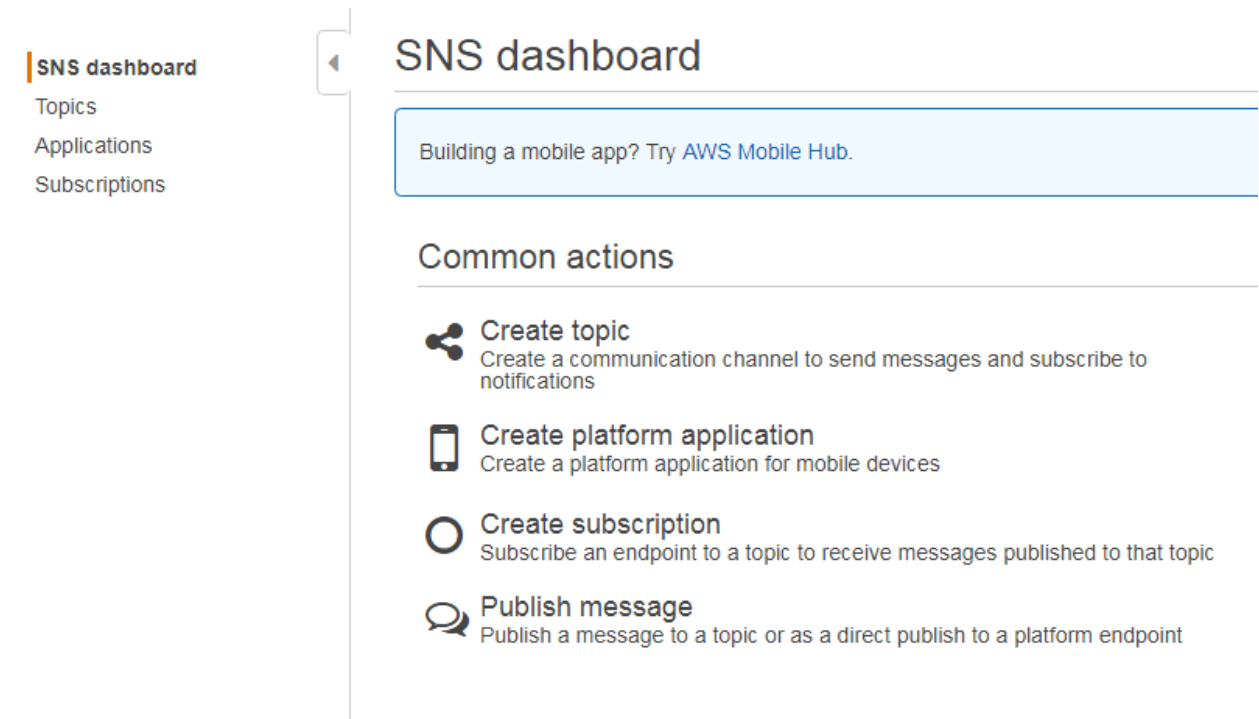


### **Messaging**

Simple Queue Service

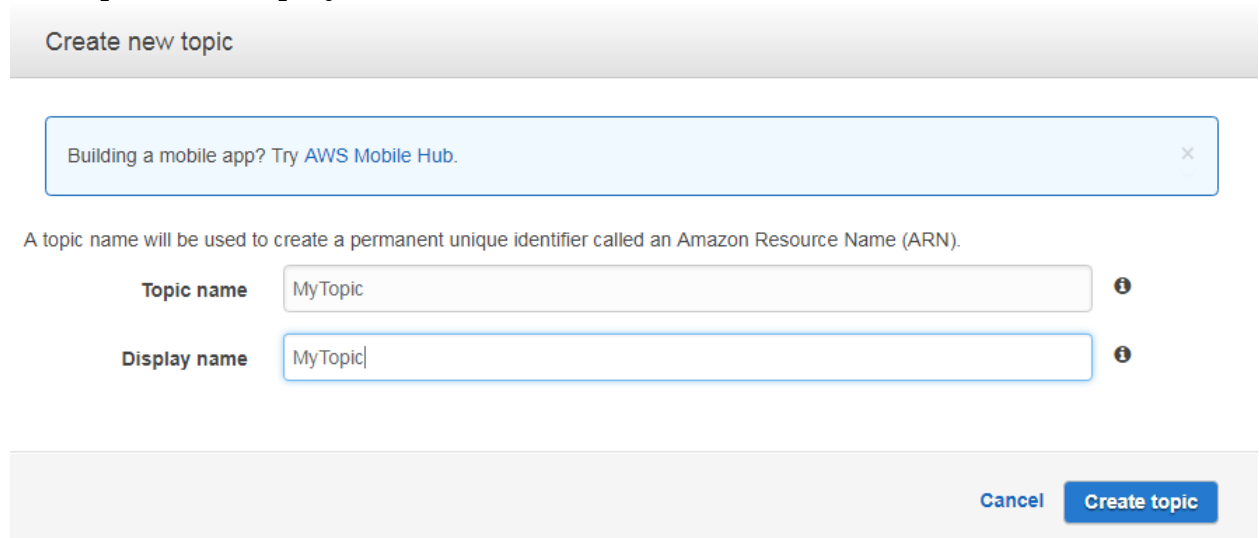
**Simple Notification Service**

Simple Email Service



The screenshot shows the AWS SNS dashboard. On the left is a sidebar with 'SNS dashboard' highlighted, and links for 'Topics', 'Applications', and 'Subscriptions'. The main area is titled 'SNS dashboard' and contains a blue banner with the text 'Building a mobile app? Try [AWS Mobile Hub](#).' Below this is a section 'Common actions' with four options: 'Create topic' (with a share icon), 'Create platform application' (with a mobile phone icon), 'Create subscription' (with a circle icon), and 'Publish message' (with a speech bubble icon). Each option has a brief description of its function.

- Create a new topic by selecting “**Create topic**” option, and give a name for Topic and Display Name.



The screenshot shows the 'Create new topic' form. At the top is a blue banner with the text 'Building a mobile app? Try [AWS Mobile Hub](#).' Below this is a note: 'A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN).' There are two input fields: 'Topic name' with the value 'MyTopic' and 'Display name' with the value 'MyTopic'. Both fields have an information icon (i) to their right. At the bottom right are two buttons: 'Cancel' and 'Create topic'.



- Here is Topic Details screen

## Topic details: MyTopic

**Publish to topic** **Other topic actions ▾**

**Topic ARN** am:aws:sns:ap-south-1:518084852393:MyTopic  
**Topic owner** 518084852393  
**Region** ap-south-1  
**Display name** MyTopic

## Subscriptions

**Create subscription** **Request confirmations** **Confirm subscription** **Other subscription actions ▾**  

Filter

<input type="checkbox"/>	Subscription ID	Protocol	Endpoint	Subscriber
--------------------------	-----------------	----------	----------	------------

- We can publish to Topic, but we don't have any Subscribers to this topic, we need to add the subscribers then we can publish to all the Subscribers at a time.
- Click on Create Subscription option and choose the Protocol as Email and Enter the Email ID you want to subscribe to this topic.

**Create subscription**

**Topic ARN**



**Protocol**

**Endpoint**

**Cancel** **Create subscription**

- Here is the status below user subscribed to the topic.

**Subscriptions**

**Create subscription** **Request confirmations** **Confirm subscription** **Other subscription actions ▾**  

Filter

<input type="checkbox"/>	Subscription ID	Protocol	Endpoint	Subscriber
<input checked="" type="checkbox"/>	PendingConfirmation	email	ry@gmail.com	



- Now login to the mentioned Email ID and verify the Email from AWS SNS, and it'll ask you to subscribe to the topic.
- You'll get an Email as mentioned below image.

## AWS Notification - Subscription Confirmation



Inbox x



**MyTopic** no-reply@sns.amazonaws.com via amazonses.com  
to me ▾

You have chosen to subscribe to the topic:  
**arn:aws:sns:ap-south-1:518084852393:MyTopic**

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):  
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

- Click on the Confirm Subscription Link, it'll redirect to another page which shows the subscription status page.



## Simple Notification Service

**Subscription confirmed!**

You have subscribed : ay@gmail.com to the topic:  
**MyTopic**

Your subscription's id is:  
**arn:aws:sns:ap-south-1:518084852393:MyTopic:4d705236-7595-4fac-a0ea-e1645819435e**

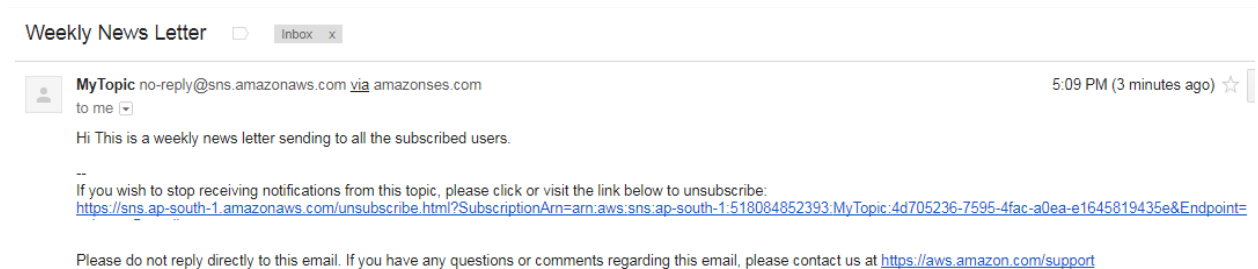
If it was not your intention to subscribe, [click here to unsubscribe](#).

- Now we can publish to the Topic, all the subscribed users will get the email/notification.
- Now select the **“Publish to Topic”** Option Then provide the Subject to the email and enter the Message to send to all the subscribers.

Amazon SNS enables you to publish notifications to all subscriptions associated with a topic as well as to an individual endpoint associated with a platform application.

Topic ARN	arn:aws:sns:ap-south-1:518084852393:MyTopic	i
Subject	Weekly News Letter	i
Message format	<input checked="" type="radio"/> Raw <input type="radio"/> JSON	
Message	<div>Hi This is a weekly news letter sending to all the subscribed users.</div>	

- Give TTL value as 300 Seconds and click on Publish message, immediately all the subscribed users will get the email.



- We can unsubscribe to the Topic at any time, and in every email we'll get unsubscribed URL, we can click on that when we want to opt-out from the topic.



