Serverless Data Processing (CSCI 5410)

Dr. Saurabh Dey

Images and contents used in these slides are from web search and/or text books, and are used for academic purposes only

Outline

Message Queue

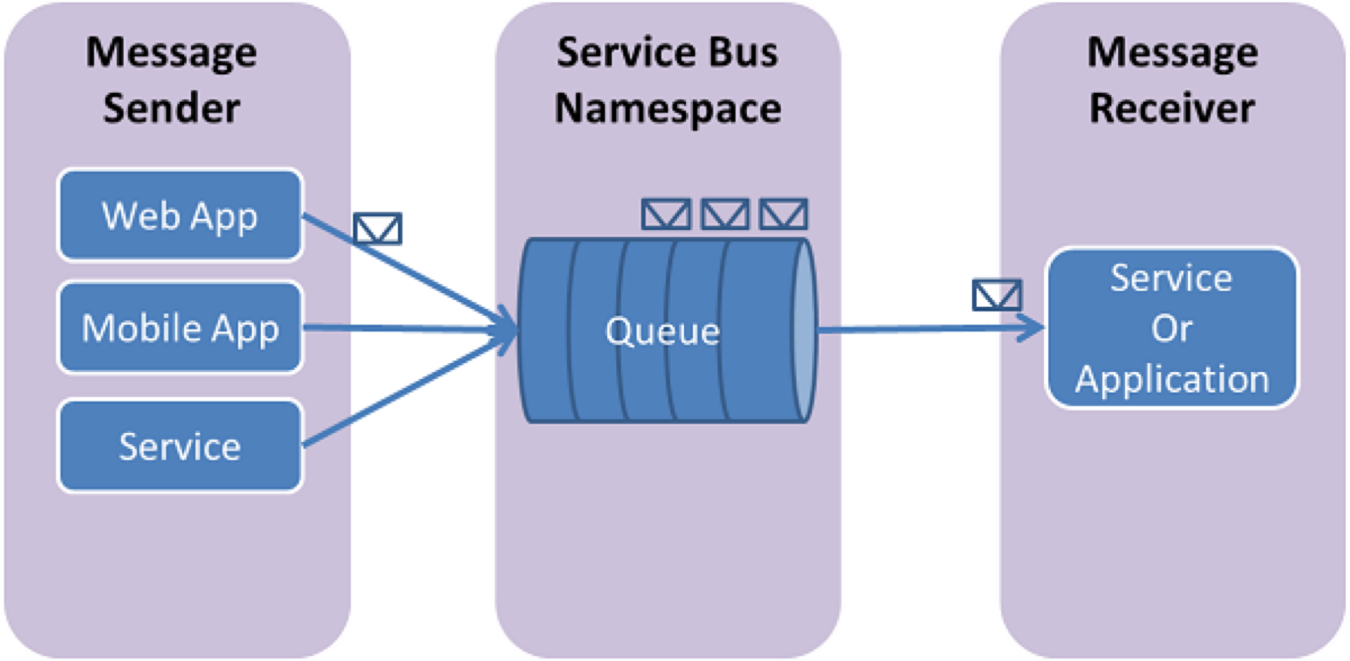
Amazon SNS

Google Pub/Sub

What is a message queue?

“A message queue is a component of messaging middleware solutions that enables independent applications and services to exchange information.”

-IBM



https://dalwax.com/azure-service-bus-queue-vs-rabbitmq/

Popular Message Queue Services

- IBM MQ
- AWS MQ
- Apache Kafka
- RabbitMQ

Benefits of Message Queues

- Better Performance
- Highly Reliable
- Scalability
- Decoupling

Better Performance

- Message queues enable asynchronous communication, which means that the endpoints that are producing and consuming messages interact with the queue, not with each other. Data flow is optimized in this scenario. Because the producers can add requests to queue without worrying about the consumer, and the consumers process messages only when they are available.

Highly Reliable

- By separating different components with message queues, more fault tolerant system can be designed. If one part of the system is ever unreachable, the other can continue to interact with the queue. The queue itself can also be mirrored for even more availability. Using a message queue can ensure that business-critical messages between applications will not be lost and that they will be only be delivered to the recipient once

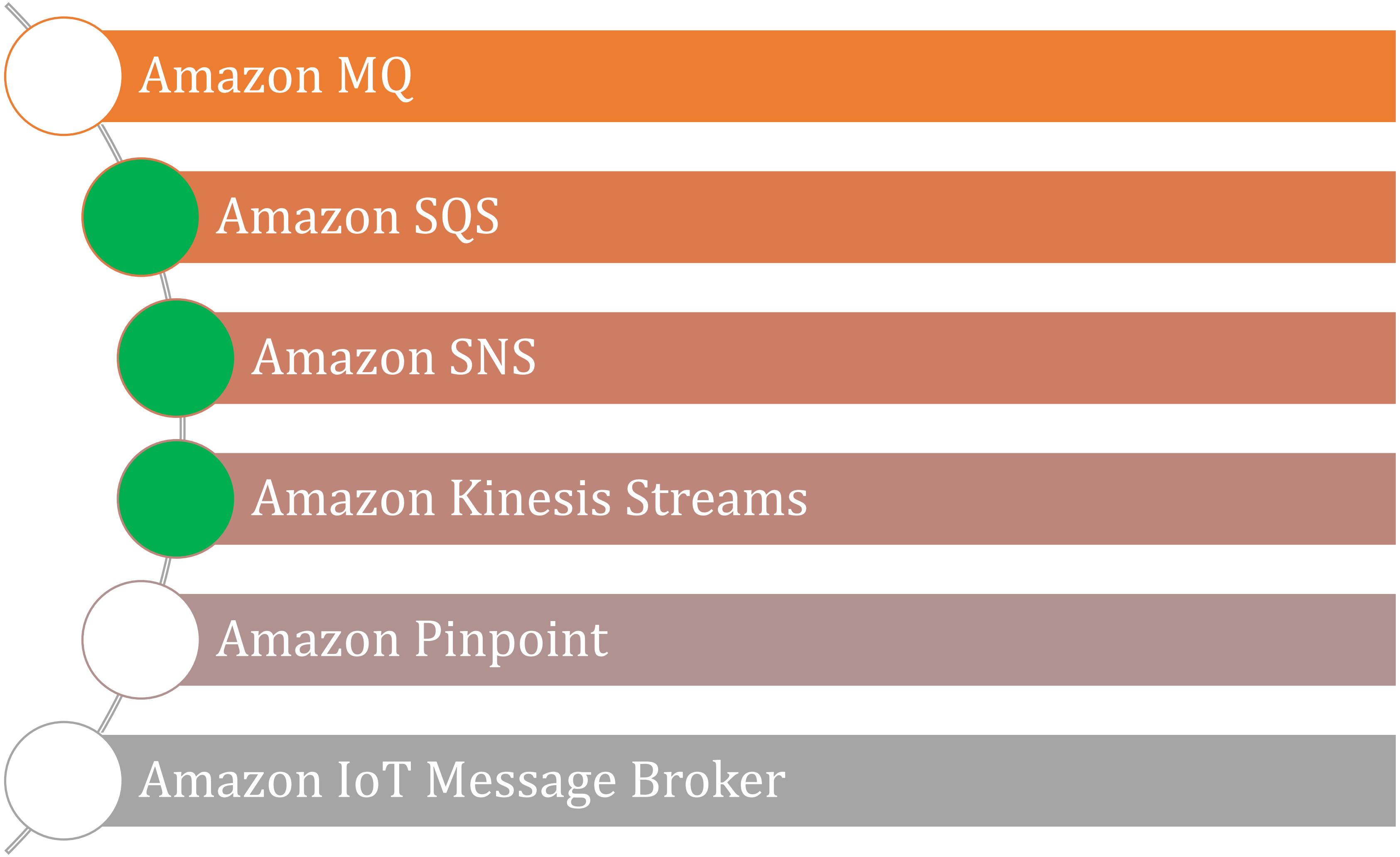
Scalability

- Message queues make it possible to scale precisely where it is needed. Multiple instances of an application can all add requests to the queue without risk of collision. Producers, consumers, and the queue itself can all grow and shrink on demand.

Decoupling

- Message queues remove dependencies between components and significantly simplify the coding of decoupled applications. Message queues are an elegantly simple way to decouple distributed systems, whether we are using monolithic applications, microservices or serverless architectures.

Amazon Messaging Services



- ○ Amazon MQ
- ● Amazon SQS
- ● Amazon SNS
- ● Amazon Kinesis Streams
- ○ Amazon Pinpoint
- ○ Amazon IoT Message Broker

Amazon MQ is used to migrate to a managed message broker to automate software administration and maintenance, without having to re-write existing applications

Amazon SQS is used to Build decoupled, highly scalable microservices, distributed systems, and serverless applications in the cloud

Amazon SNS is used to Push messages to a variety of endpoints and clients in distributed systems, microservices, and serverless applications and enable event-driven architecture

Amazon Kinesis Streams is used to Build custom, real-time applications that process data streams using popular stream processing frameworks

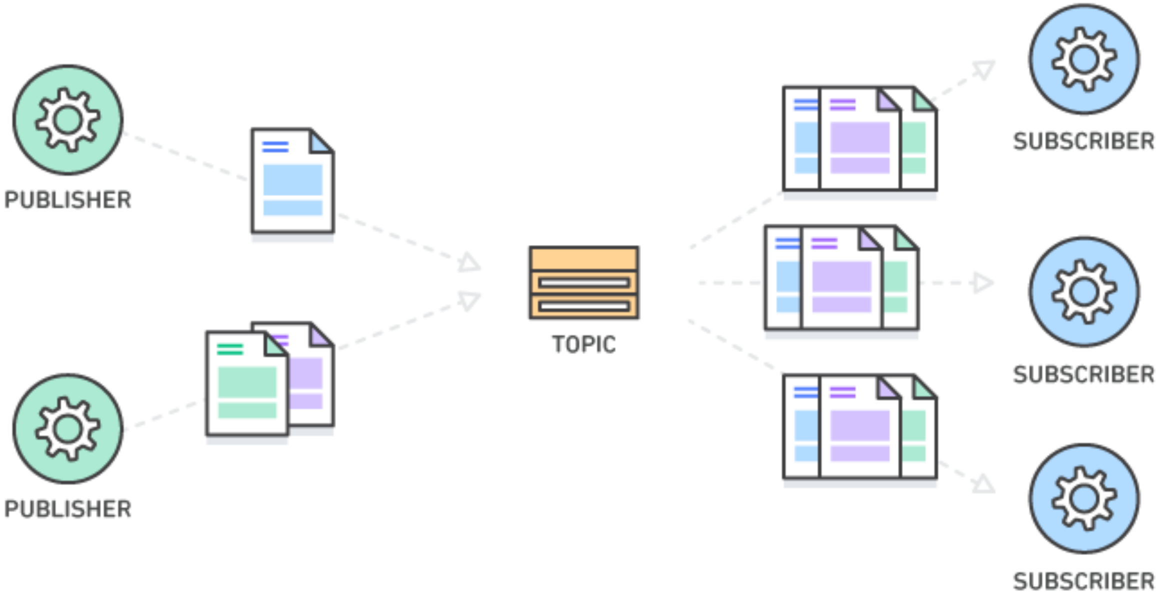
Amazon Pinpoint is used to Deliver the right message to the right customer at the right time to improve engagement and conversion

Amazon IoT Message Broker is used to Send messages to/from devices and AWS IoT apps in a secure fashion using MQTT, HTTP, and WebSockets

Publish/Subscribe Messaging

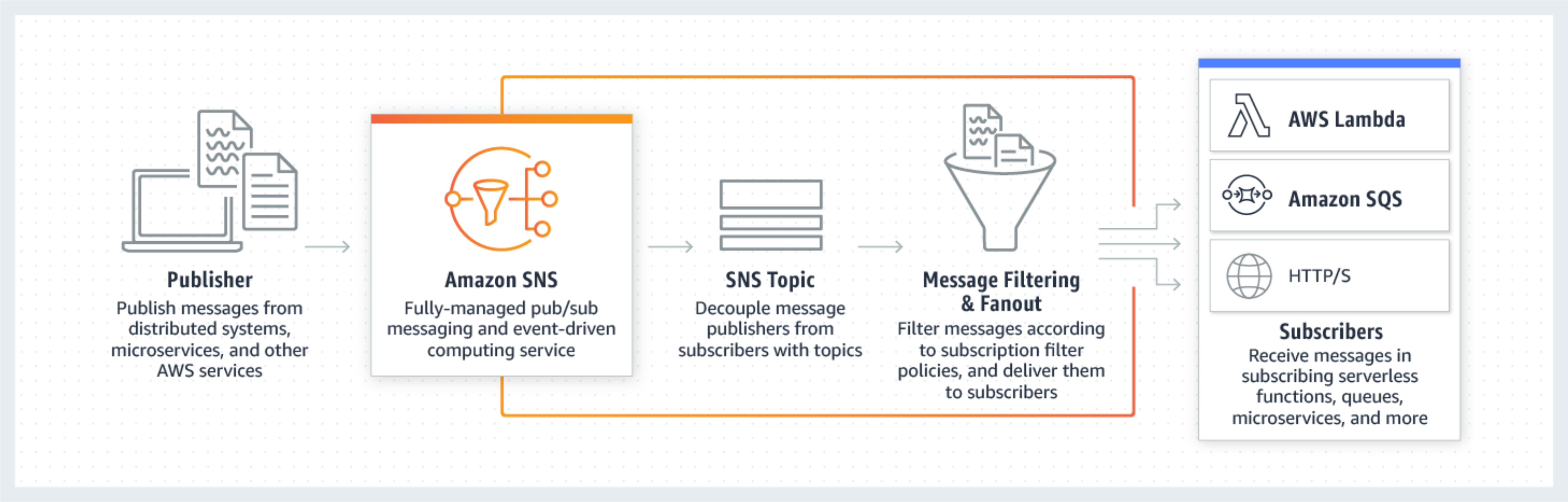
- A form of asynchronous service-to-service communication used in serverless, and microservices architectures.

- In modern cloud architecture, applications are decoupled into smaller, independent building blocks that are easier to develop, deploy and maintain. Publish/Subscribe (Pub/Sub) messaging provides instant event notifications for these distributed applications.



https://aws.amazon.com/pub-sub-messaging/

Amazon Simple Notification Service



Publisher
Publish messages from distributed systems, microservices, and other AWS services

Amazon SNS
Fully-managed pub/sub messaging and event-driven computing service

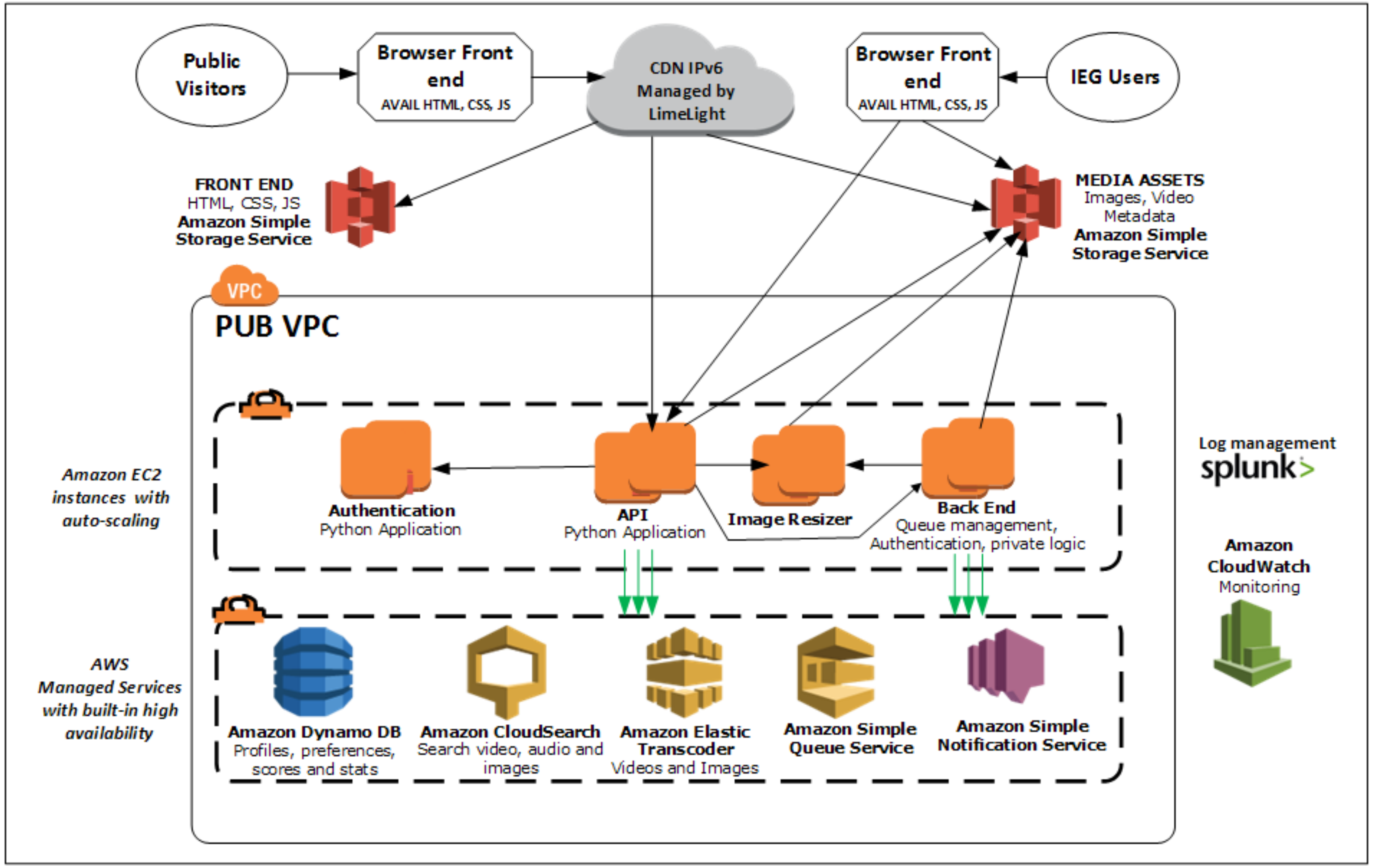
SNS Topic
Decouple message publishers from subscribers with topics

Message Filtering & Fanout
Filter messages according to subscription filter policies, and deliver them to subscribers

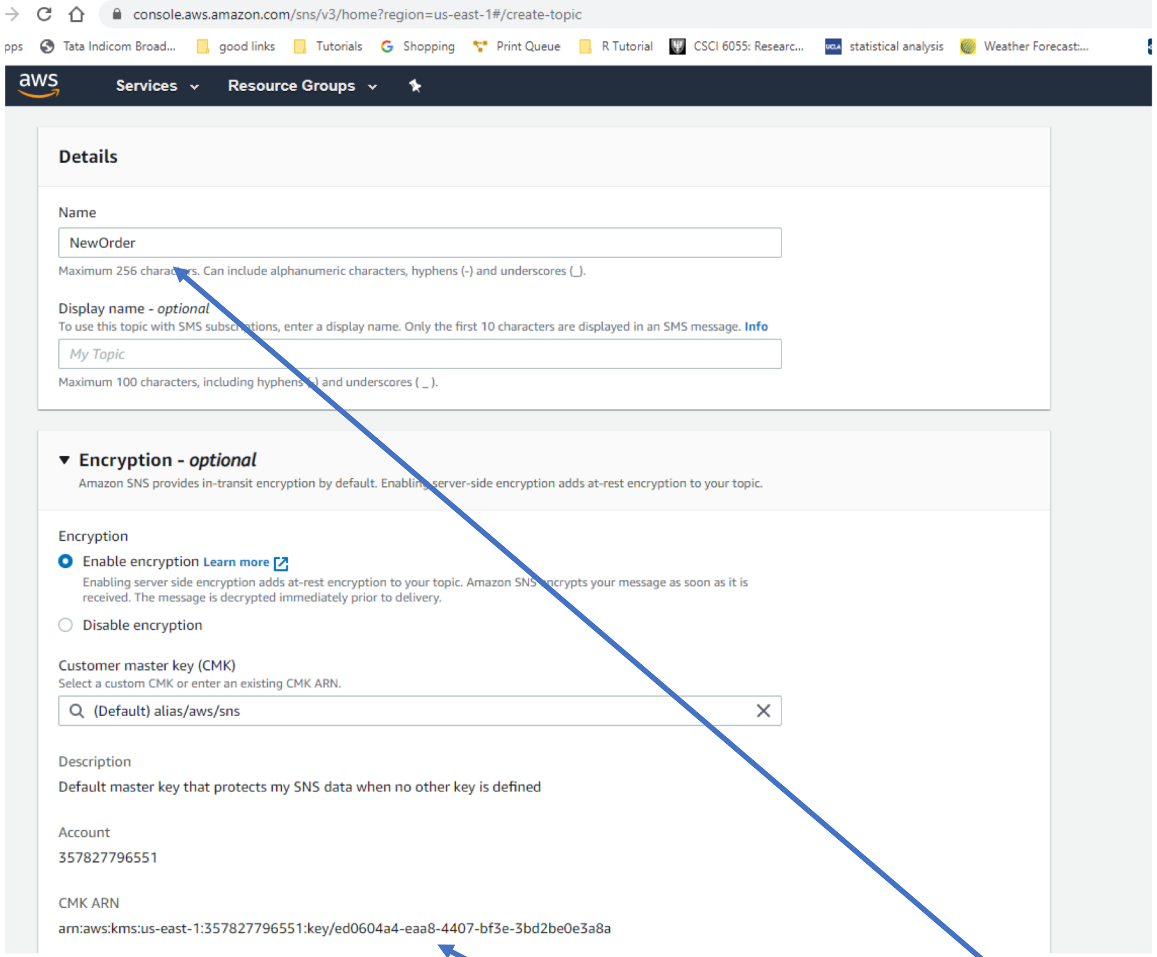
Subscribers
Receive messages in subscribing serverless functions, queues, microservices, and more

Amazon SNS Case Study

- The NASA Image and Video Library is a cloud-native solution


Public Visitors
Browser Front end
AVAIL HTML, CSS, JS
CDN IPv6 Managed by LimeLight
Browser Front end
AVAIL HTML, CSS, JS
IEG Users
FRONT END
HTML, CSS, JS
Amazon Simple Storage Service
MEDIA ASSETS
Images, Video Metadata
Amazon Simple Storage Service
VPC
PUB VPC
Amazon EC2 instances with auto-scaling
Authentication
Python Application
API
Python Application
Image Resizer
Back End
Queue management, Authentication, private logic
AWS Managed Services with built-in high availability
Amazon Dynamo DB
Profiles, preferences, scores and stats
Amazon CloudSearch
Search video, audio and images
Amazon Elastic Transcoder
Videos and Images
Amazon Simple Queue Service
Amazon Simple Notification Service
Log management
splunk>
Amazon CloudWatch
Monitoring

https://aws.amazon.com/partners/success/nasa-image-library/



Details

Name

NewOrder

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional*

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. Info

My Topic

Maximum 100 characters, including hyphens (-) and underscores (_).

▼ **Encryption - *optional***

Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Encryption

● Enable encryption Learn more

Enabling server side encryption adds at-rest encryption to your topic. Amazon SNS encrypts your message as soon as it is received. The message is decrypted immediately prior to delivery.

○ Disable encryption

Customer master key (CMK)

Select a custom CMK or enter an existing CMK ARN.

(Default) alias/aws/sns

Description

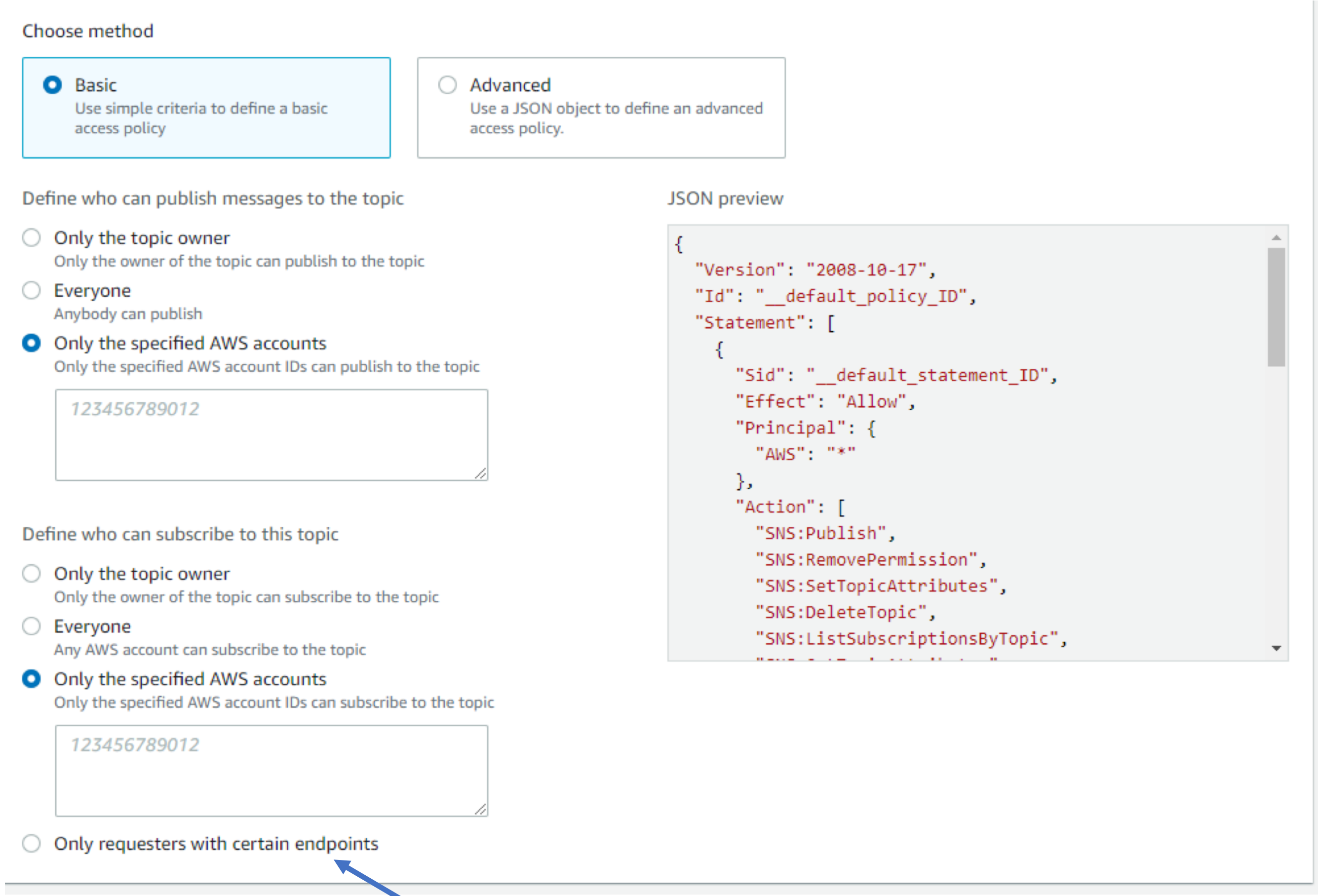
Default master key that protects my SNS data when no other key is defined

Account

357827796551

CMK ARN

arn:aws:kms:us-east-1:357827796551:key/ed0604a4-eaa8-4407-bf3e-3bd2be0e3a8a



Choose method

● Basic
Use simple criteria to define a basic access policy

○ Advanced
Use a JSON object to define an advanced access policy.

Define who can publish messages to the topic

○ Only the topic owner
Only the owner of the topic can publish to the topic

○ Everyone
Anybody can publish

● Only the specified AWS accounts
Only the specified AWS account IDs can publish to the topic

123456789012

Define who can subscribe to this topic

○ Only the topic owner
Only the owner of the topic can subscribe to the topic

○ Everyone
Any AWS account can subscribe to the topic

● Only the specified AWS accounts
Only the specified AWS account IDs can subscribe to the topic

123456789012

○ Only requesters with certain endpoints

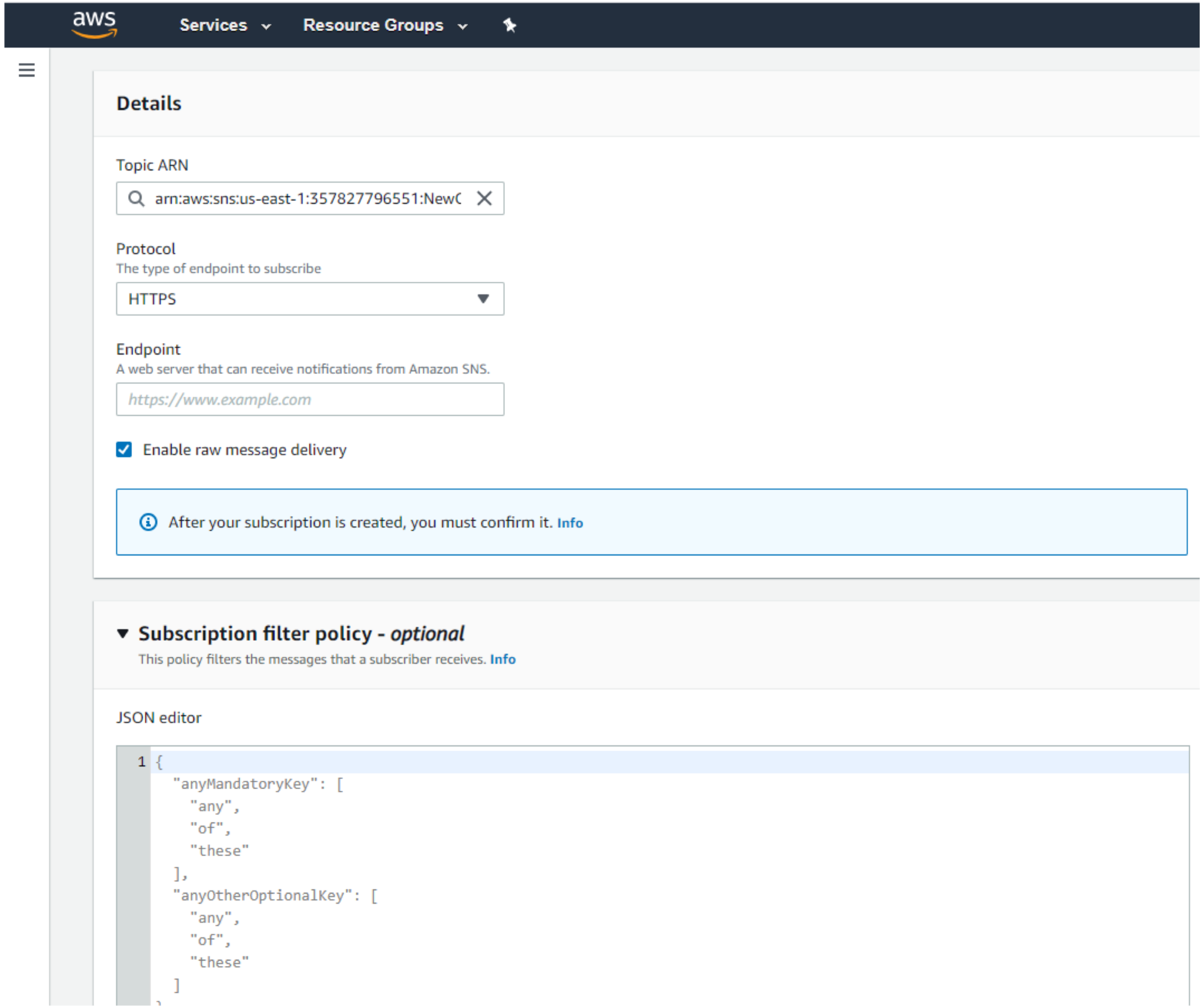
JSON preview

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:Publish",
        "SNS:RemovePermission",
        "SNS:SetTopicAttributes",
        "SNS:DeleteTopic",
        "SNS:ListSubscriptionsByTopic",
```

We can add endpoints or define it open.

New Topic

Enabled Encryption



Details

Topic ARN

arn:aws:sns:us-east-1:357827796551:NewC

Protocol
The type of endpoint to subscribe

HTTPS

Endpoint
A web server that can receive notifications from Amazon SNS.

https://www.example.com

☑ Enable raw message delivery

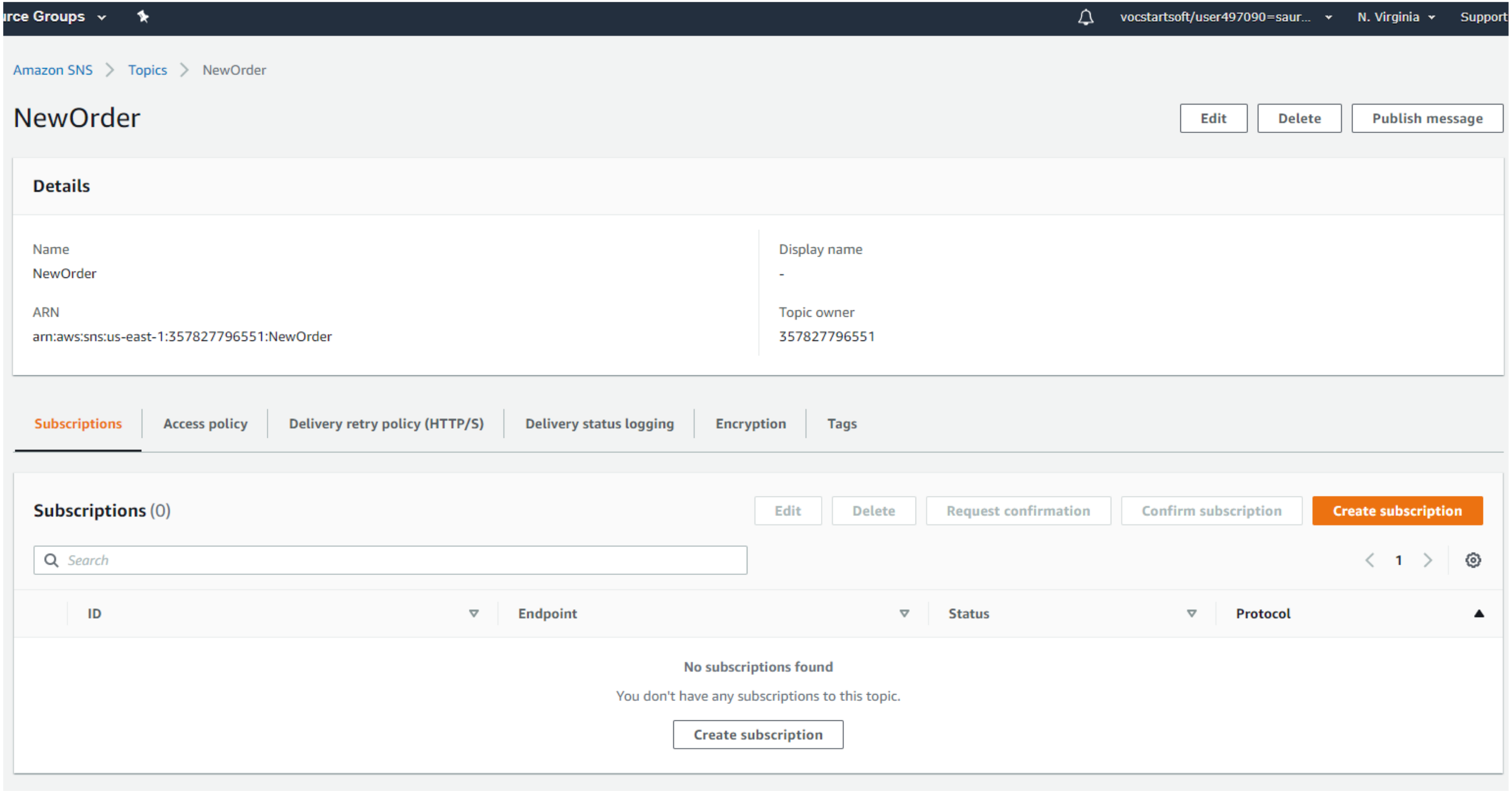
After your subscription is created, you must confirm it. Info

Subscription filter policy - *optional*

This policy filters the messages that a subscriber receives. Info

JSON editor

```
{
  "anyMandatoryKey": [
    "any",
    "of",
    "these"
  ],
  "anyOtherOptionalKey": [
    "any",
    "of",
    "these"
  ]
```



NewOrder

Details

Name
NewOrder

ARN
arn:aws:sns:us-east-1:357827796551:NewOrder

Display name
-

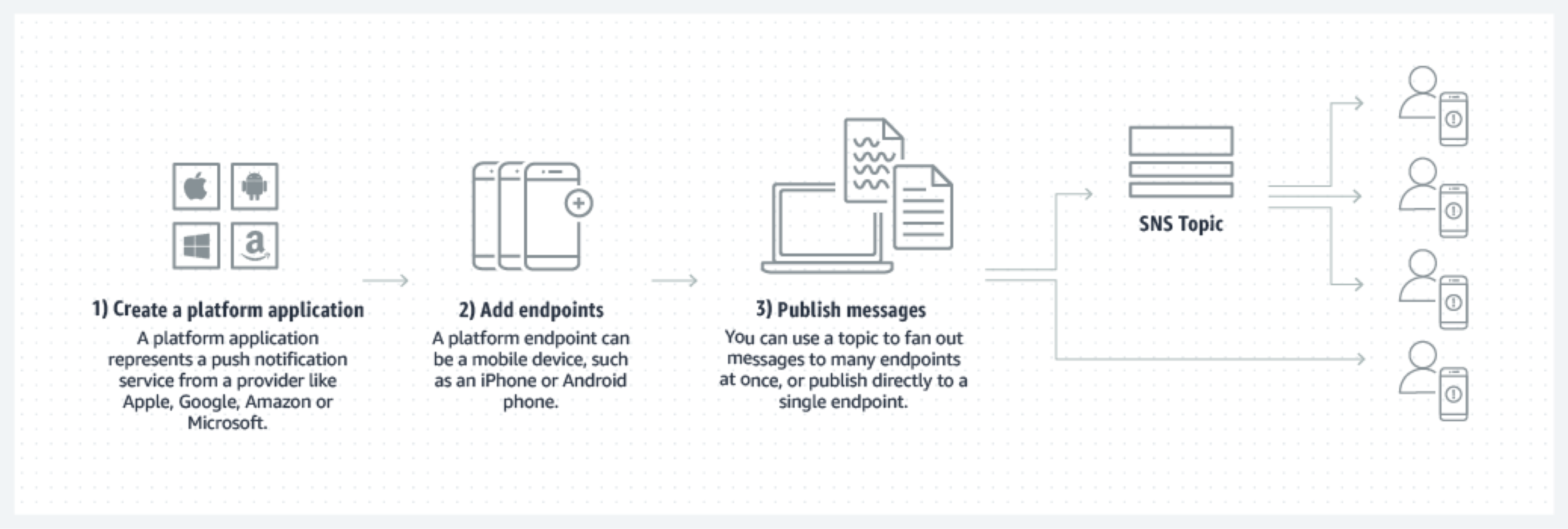
Topic owner
357827796551

Subscriptions (0)

ID	Endpoint	Status	Protocol

No subscriptions found

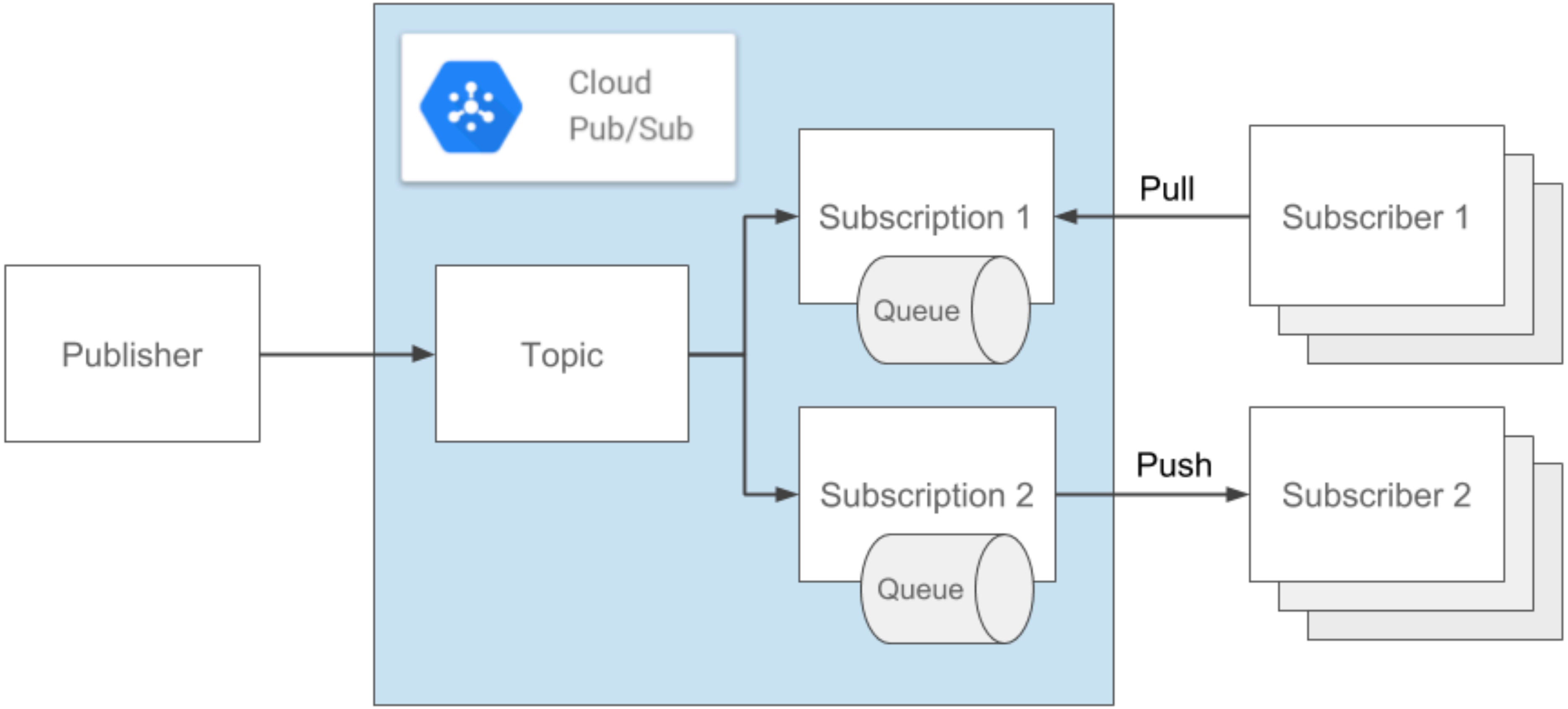
You don't have any subscriptions to this topic.



1) Create a platform application
A platform application represents a push notification service from a provider like Apple, Google, Amazon or Microsoft.
2) Add endpoints
A platform endpoint can be a mobile device, such as an iPhone or Android phone.
3) Publish messages
You can use a topic to fan out messages to many endpoints at once, or publish directly to a single endpoint.
SNS Topic

Amazon SNS lets you send push notifications to apps on mobile devices.

Google Pub/Sub



Source: https://subscription.packtpub.com/book/programming/9781838647438/14/ch14lvl1sec102/google-cloud-pub-sub

Google Pub/Sub (REST)

List Topic

GET https://pubsub.googleapis.com/v1/projects/*PROJECT_ID*/topics Authorization: Bearer *ACCESS_TOKEN*

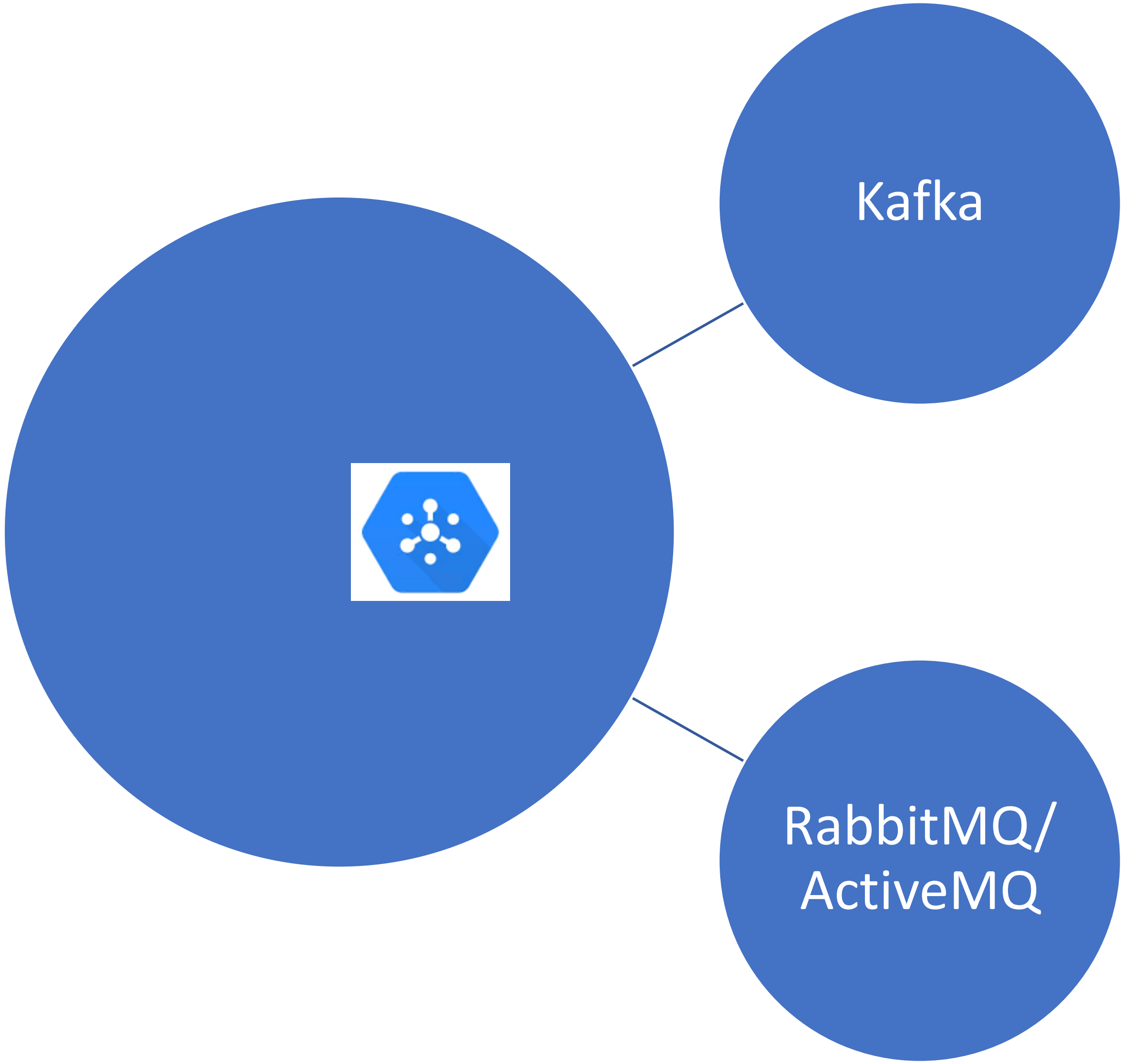
Publish message to a Topic

POST https://pubsub.googleapis.com/v1/projects/PROJECT_ID/topics/TOPIC_ID:publish Content-Type: application/json Authorization: Bearer $(gcloud auth application-default print-access-token)

Request Body

```
{
 "messages": [
  {
   "attributes": {
    "KEY": "VALUE",
    ...
   },
   "data": "MESSAGE_DATA",
  }
 ]
}
```

Why GCP Pub/Sub is better?



Kafka

- Horizontally scalable

RabbitMQ/ ActiveMQ

- Traditional messaging middleware
- per-message parallelism

Questions to Consider

- Can SNS be used as part of instant messaging service?
- How is SNS different than Google Pub/Sub?