

NOTIFICATION USING SNS-SQS

Why notifications?

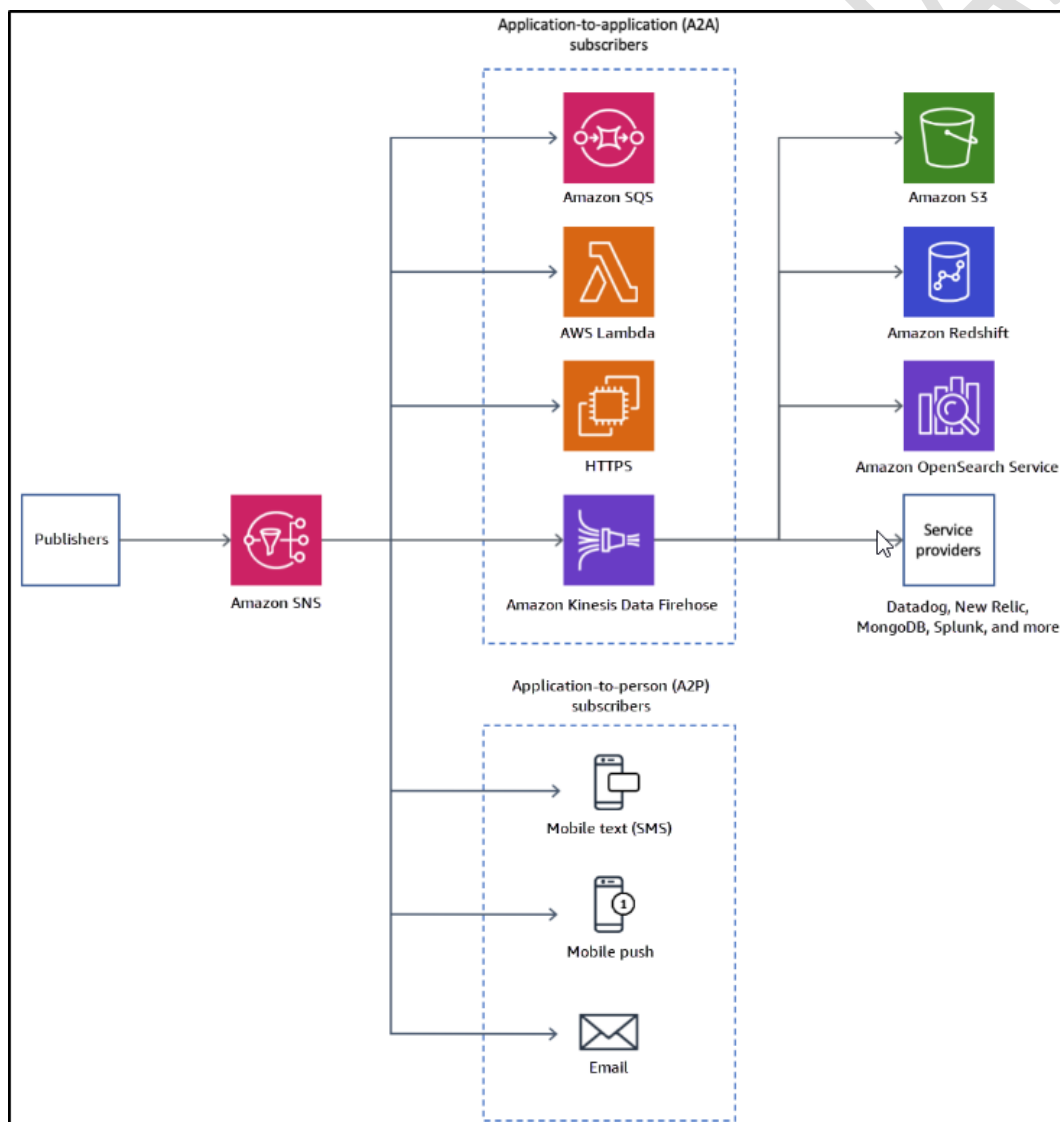
- To know that jobs have run successfully or failed. If failed, details of the failure.

AWS SNS & SQS INTRO: #detailed in AWS

What is SNS?

- Simple notification service.
- It's an alerting service.

AWS official documentation: <https://docs.aws.amazon.com/sns/latest/dg/welcome.html>



SNS topics are used to enable communication:

- Producers publish messages to topics, and

- Consumers subscribe to these topics to receive messages.

What is SQS?

- Simple Queue Service.
- AWS SQS is **queueing** service used for communication between applications, micro services, and distributed systems. # in short, software components
- Components:
 - **Producers** (components that send messages to the queue).
 - **Queue** (which stores messages).
 - **Consumers** (other components that receive messages from the queue).

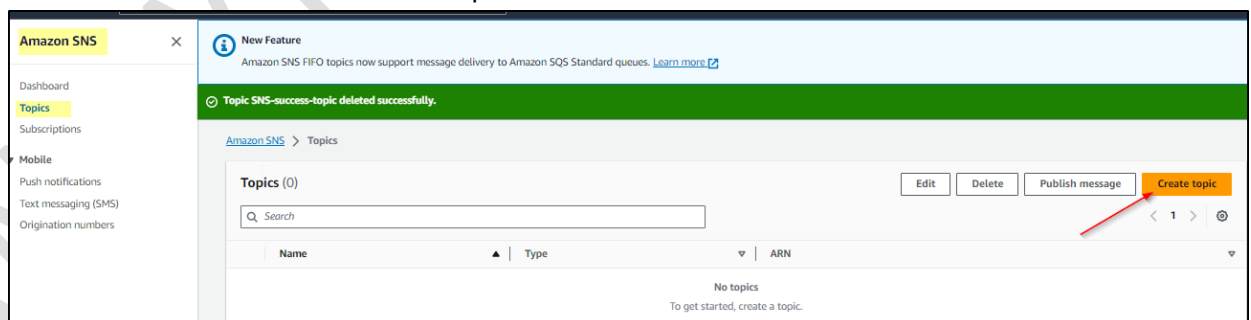
Differences:

SNS	SQS
SNS supports A2A and A2P communication	SQS supports only A2A communication
<ul style="list-style-type: none"> ➤ SNS is a pub/sub system ➤ Typically use SNS to send the same message to multiple consumers via topics. 	<ul style="list-style-type: none"> ➤ SQS is a queueing system. ➤ Each message in an SQS queue is processed by only one consumer.
<ul style="list-style-type: none"> ➤ SNS does not persist messages - it delivers them to subscribers that are present, and then deletes them. 	<ul style="list-style-type: none"> ➤ SQS can persist messages (from 1 minute to 14 days).

AWS SNS JOB:

Email and SMS alert using SNS:

- In AWS:
 - Create an SNS topic in AWS
 - Go to Amazon SNS and click Create topic



- 1.1.2. Select the 'Standard' and update the name fields highlighted below. Let the defaults be and click on *Create topic*

Details

Type | [Info](#)

Topic type cannot be modified after topic is created

☐ FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

☒ Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

Ren-SNS-Topic

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

Display name - optional | [Info](#)

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

Ren-SNS-Topic

Maximum 100 characters.

Topic is created:

Topic Ren-SNS-Topic created successfully.
You can create subscriptions and send messages to them from this topic.

[Amazon SNS](#) > [Topics](#) > Ren-SNS-Topic

Ren-SNS-Topic

EditDeletePublish message

Details

Name

Ren-SNS-Topic

Display name

Ren-SNS-Topic

ARN

arn:aws:sns:us-west-1:275156976860:Ren-SNS-Topic

Type

Standard

Topic owner

275156976860

Subscriptions

Access policy

Data protection policy

Delivery policy (HTTP/S)

Delivery status logging

Encryption

Tags

Integrations

Subscriptions (0)

EditDeleteRequest confirmationConfirm subscriptionCreate subscription

1.2. Create a subscription to the topic with Email alerts and SMS

1.2.1. Click on create subscriptions in the same page

Ren-SNS-Topic

EditDeletePublish message

Details

Name

Ren-SNS-Topic

Display name

Ren-SNS-Topic

ARN

arn:aws:sns:us-west-1:275156976860:Ren-SNS-Topic

Type

Standard

Topic owner

275156976860

Subscriptions

Access policy

Data protection policy

Delivery policy (HTTP/S)

Delivery status logging

Encryption

Tags

Integrations

Subscriptions (0)

EditDeleteRequest confirmationConfirm subscriptionCreate subscription

Search

< 1 > ⚙

1.2.2. Add protocol as 'Email', endpoint click 'create subscription'

Create subscription

Details

Topic ARN

Q X

Protocol

The type of endpoint to subscribe

Email ▼

Endpoint

An email address that can receive notifications from Amazon SNS.

After your subscription is created, you must confirm it. [Info](#)

► **Subscription filter policy - optional** [Info](#)

This policy filters the messages that a subscriber receives.

► **Redrive policy (dead-letter queue) - optional** [Info](#)

Send undeliverable messages to a dead-letter queue.

Cancel

Create subscription

1.2.2.1. Displays as pending confirmation

Subscriptions (1)

Edit

Delete

Request confirmation

Confirm subscription

Create subscription

Q Search

<

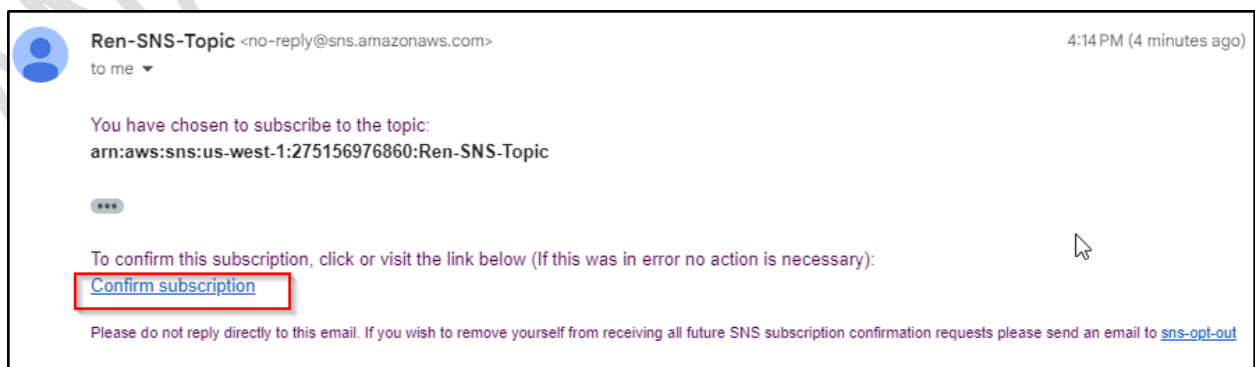
1

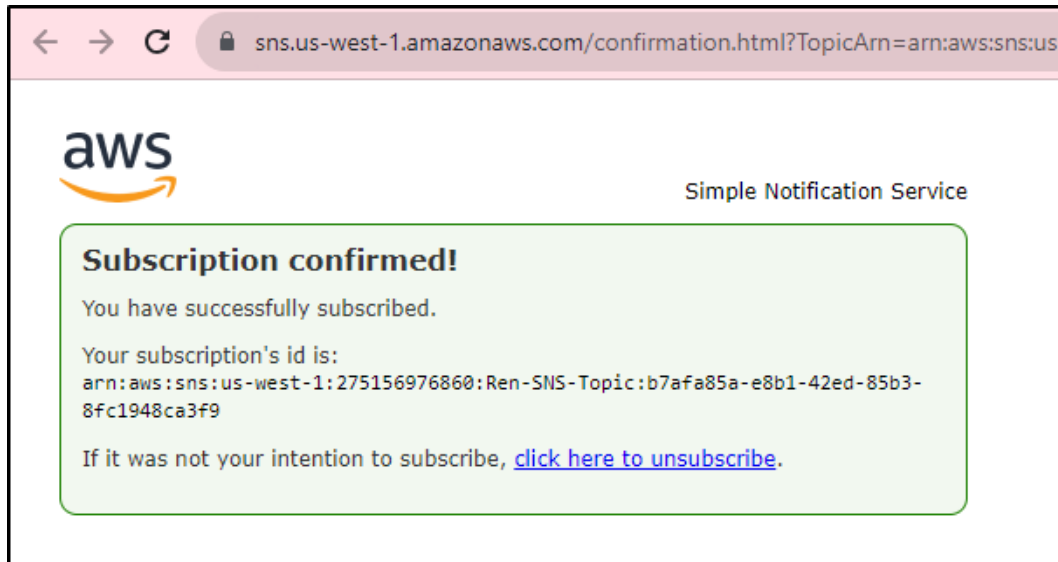
>

⊙

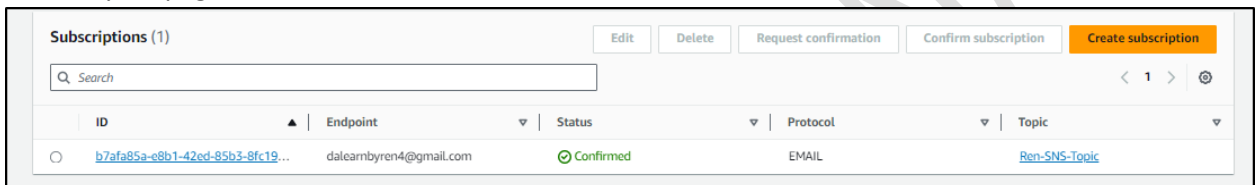
	ID	Endpoint	Status	Protocol	Topic
<input type="radio"/>	Pending confirmation	dalearnbyren4@gmail.com	<input checked="" type="radio"/> Pending confirmation	EMAIL	Ren-SNS-Topic

1.2.3. Check your mailbox for the confirmation email (could be in Spam as well) and click 'confirm subscriptions'.

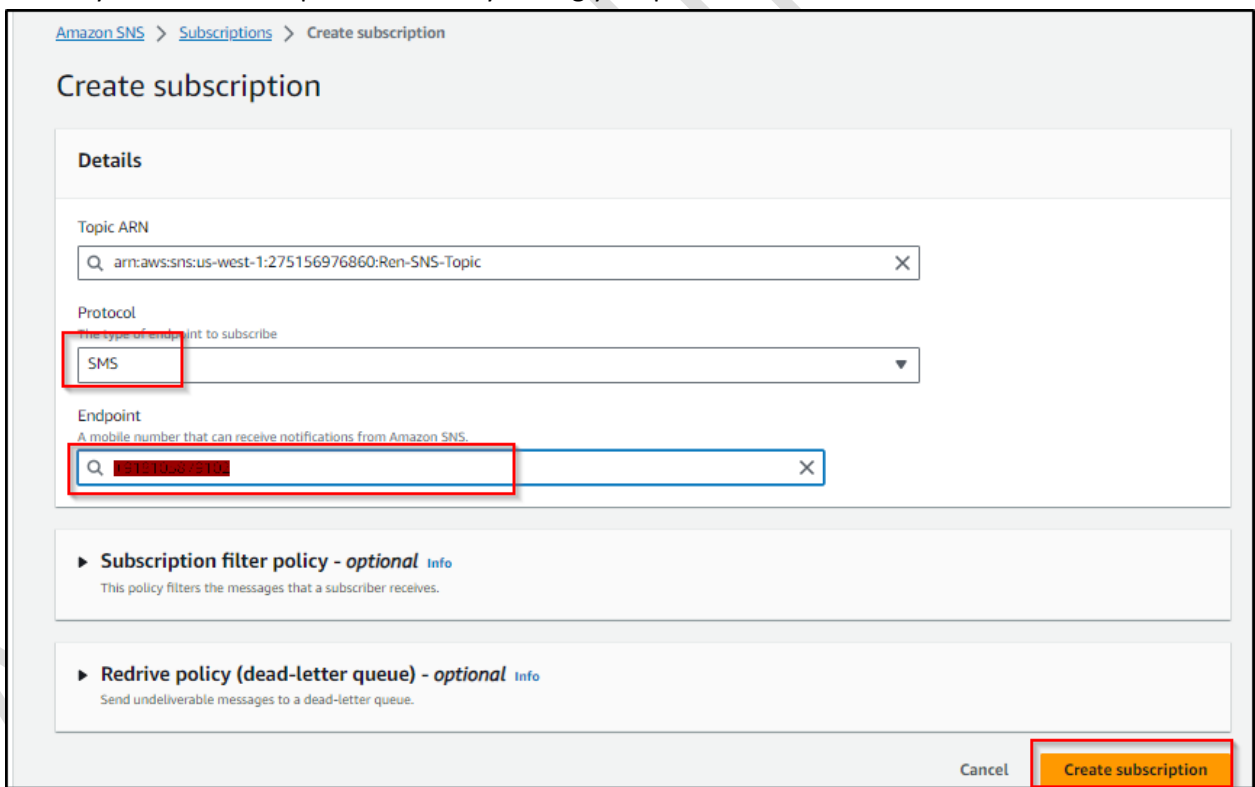




1.3. Refresh your page in Amazon SNS . Status will be confirmed.



1.4. Similarly create a subscription for SMS by adding your phone number



2. In matillion,

2.1. Create an orchestration project

2.2. Add python script component

```
1 ###
2 # Variables are directly accessible:
3 # print (myvar)
4 # Updating a variable:
5 # context.updateVariable('myvar', 'new-value')
6 # Grid Variables are accessible via the context:
7 # print (context.getGridVariable('mygridvar'))
8 # Updating a grid variable:
9 # context.updateGridVariable('mygridvar', [['list','of
10 # A database cursor can be accessed from the context (Jy
11 # cursor = context.cursor()
12 # cursor.execute('select count(*) from mytable')
13 # rowcount = cursor.fetchone()[0]
14 ###
15 print("running python job - Ren")
```

Run Python 3 Timeout(seconds):

running python job - Ren
Script duration: 0.1s.

2.3. Add SNS Message component to your job and configure it for Job Success

2.3.1.Details :

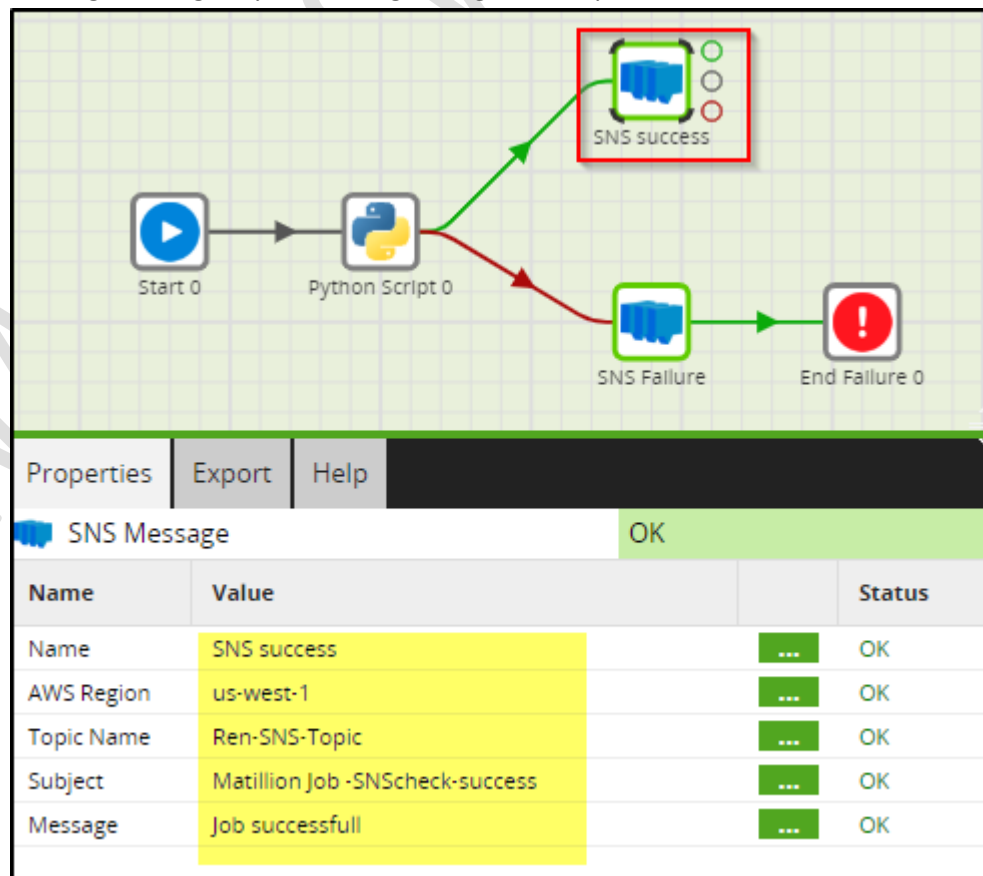
Name: Give a name for the component

AWS Region: Select the region where we created the SNS topic in AWS

Topic Name: Select the topic name. It should come in the drop-down.

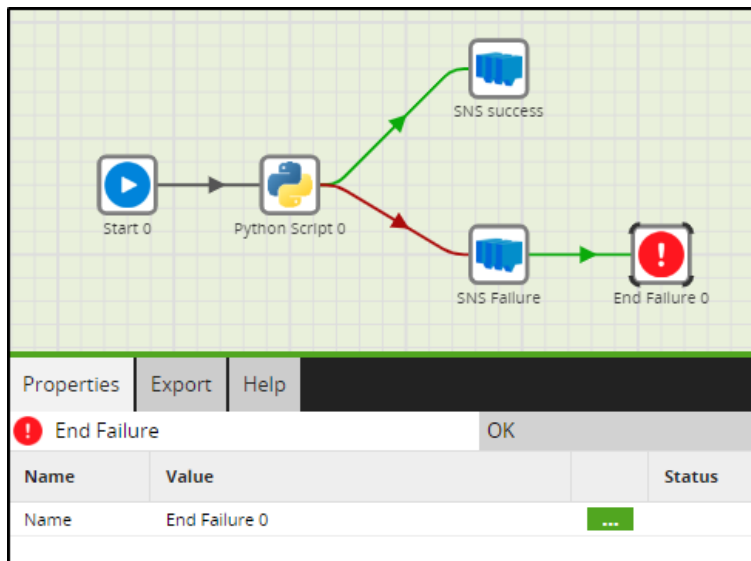
Subject: Give the Email Alert a Subject line.

Message: Configure your message using the Component Metadata and Environment variables.



3. Similarly, Add SNS Message component to your job and configure it for Job Failure

4. Add an End Failure component for the job (optional)
 - 4.1. The overall status of the job will be failure, even if it would have been otherwise successful.
 - 4.2. End Failure component



SNS Alert with Job details using automatic variables:

Automatic variables – Already created in the Matillion account by default. It holds the metadata.

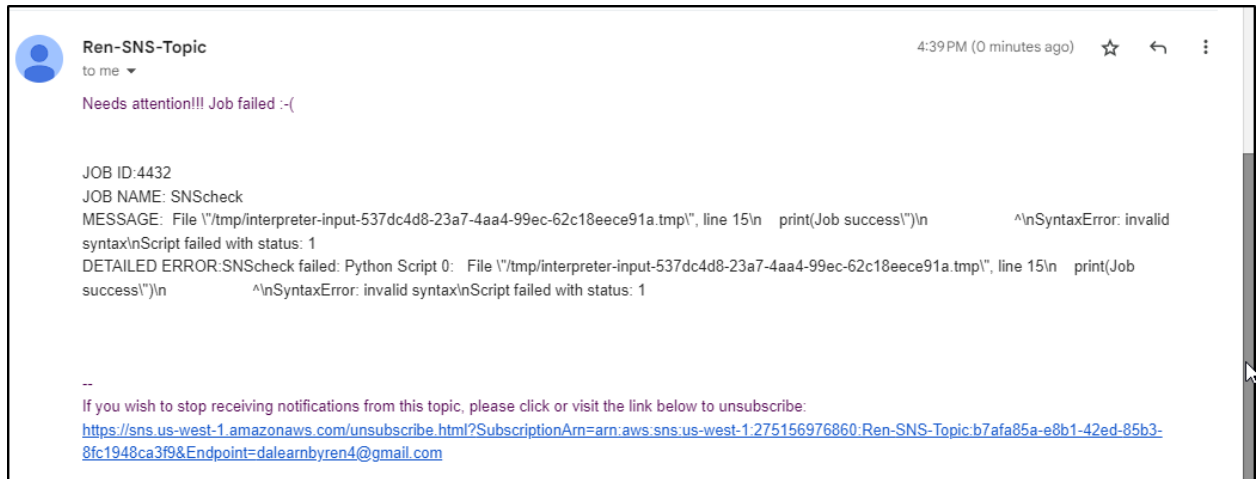
- In the SNS component > message, add below variables
 - JOB ID:\${job_id}
 - JOB NAME: \${job_name}
 - MESSAGE:\${component_message}
 - DETAILED ERROR:\${detailed_error}

```

1 Needs attention!!! Job failed :-(<
2
3
4 JOB ID:${job_id}
5 JOB NAME: ${job_name}
6 MESSAGE:${component_message}
7 DETAILED ERROR:${detailed_error}
8
9

```

- Run Job
 - Email received on failure:



SQS Notification:

Run a Job B based on result in Job A:

1. We will continue from the Job that we already created. We consider it as Job A
2. Create a simple job B
 - 2.1. Add a python script component with simple script .
 - 2.2. Added a SNS message component to receive a email on completion (optional)

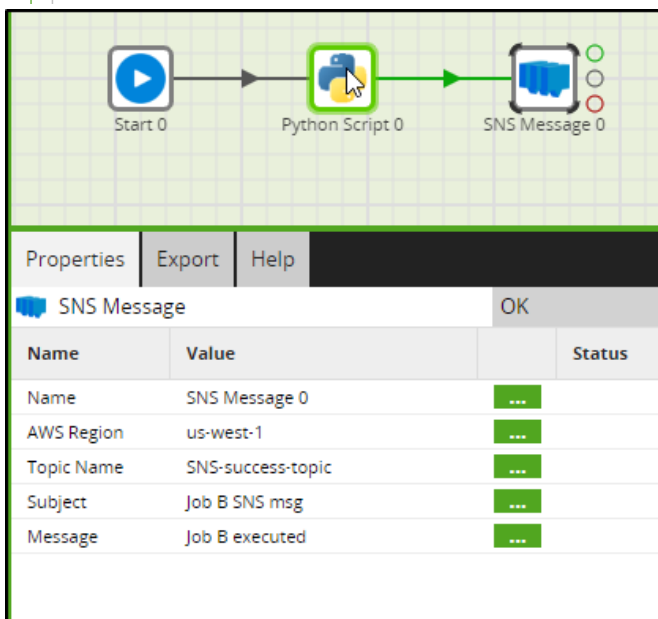
The screenshot shows a workflow diagram with three components: Start 0, Python Script 0, and SNS Message 0, connected in sequence. Below the diagram is a table with tabs for Properties, Export, and Help. The Properties tab is active, showing the configuration for the Python Script component.

Name	Value	Status
Name	Python Script 0	...
Script	### # Variables are directly acc...	...
Interpreter	Python 3	...
Timeout	360	...


```
1 ###
2 # Variables are directly accessible:
3 # print(myvar)
4 # Updating a variable:
5 # context.updateVariable('myvar', 'new-value')
6 # Grid Variables are accessible via the context:
7 # print(context.getGridVariable('mygridvar'))
8 # Updating a grid variable:
9 # context.updateGridVariable('mygridvar', [['list']])
10 # A database cursor can be accessed from the context:
11 cursor = context.cursor()
12 cursor.execute('select count(*) from mytable')
13 rowcount = cursor.fetchone()[0]
14 ###
15 print("JobB ran successfully")
```

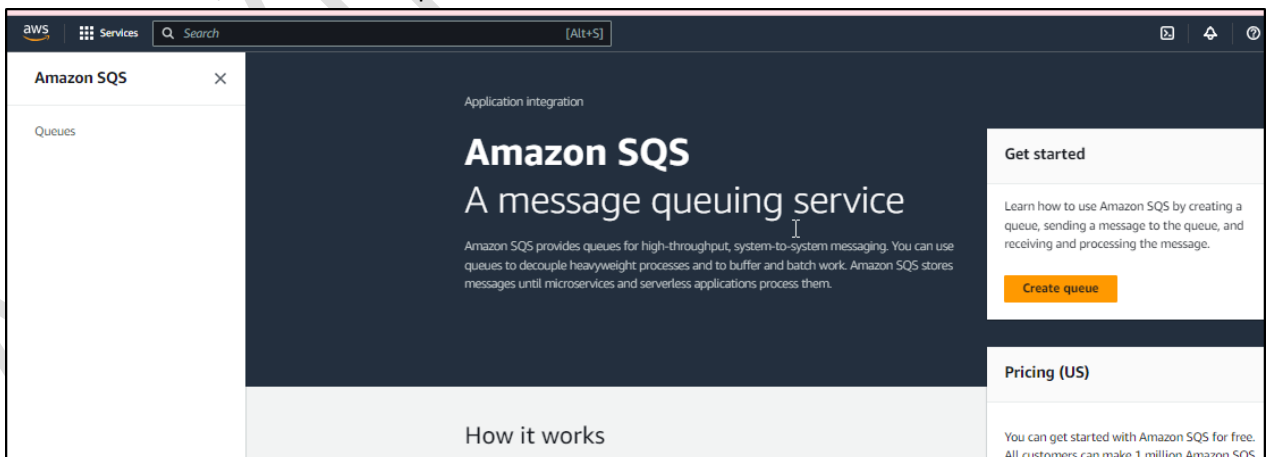
Run Python 3 Timeout (secor)

Press "Run" to test script execution.



3. Create queue:

3.1. Go to SQS in AWS, click on create queue



3.2. Write the name and click on 'create queue', We will take this as 'listen queue'.

Create queue

Details

Type

Choose the queue type for your application or cloud infrastructure.

☒ **Standard** [Info](#)
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

☐ **FIFO** [Info](#)
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

i You can't change the queue type after you create a queue.

Name

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Configuration

Set the maximum message size, visibility to other consumers, and message retention.

Visibility timeout [Info](#)

Message retention period [Info](#)

Should be between 0 seconds and 12 hours.

Should be between 1 minute and 14 days.

3.3. Similarly create 2 more queues for Success and Failure

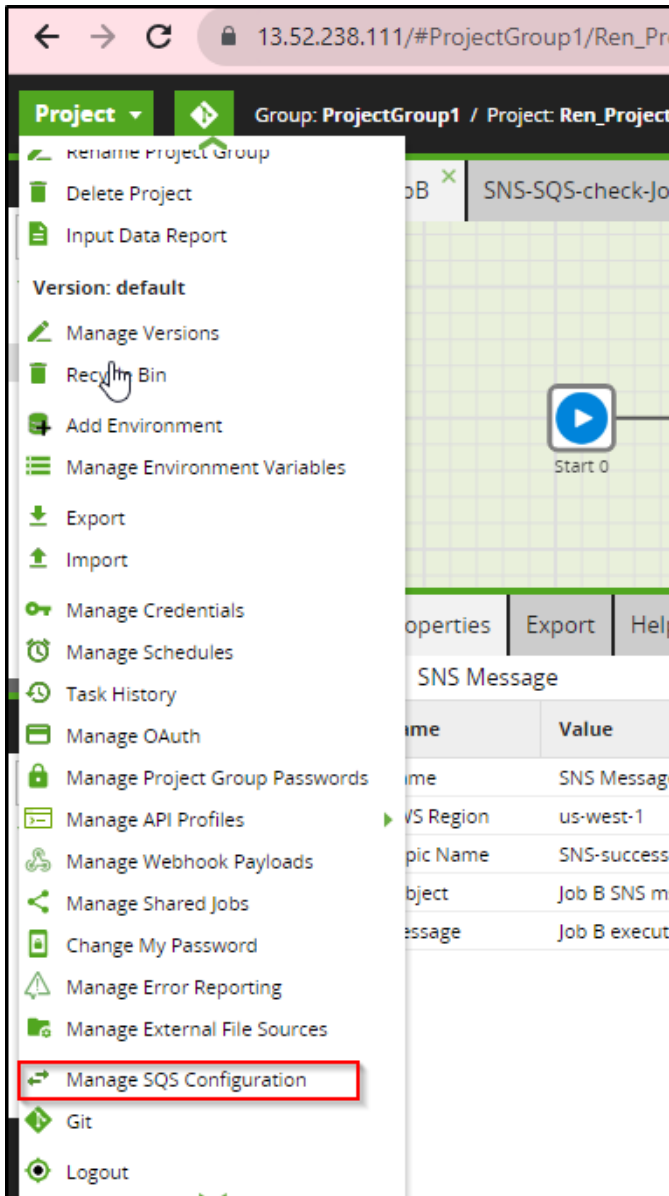
Queues (3)

	Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
<input type="radio"/>	Matillion-start-jobB-queue	Standard	2023-10-05T10:17+05:30	0	0	Amazon SQS key (SSE-SQS)	-
<input type="radio"/>	Test_Failure	Standard	2023-10-05T10:32+05:30	4	0	Amazon SQS key (SSE-SQS)	-
<input type="radio"/>	Test_Success	Standard	2023-10-05T10:31+05:30	1	0	Amazon SQS key (SSE-SQS)	-

4. In Matillion,

We need to Configure in 'Manage SQS Credentials ':

Go to Project > 'Manage SQS Credentials '



Fill in the details

Manage SQS Configuration

Listen

Enable SQS: ☒

Credentials: Ren_AWS_CRed Manage

Region: us-west-1

Listen Queue: Matillion-start-jobB-queue

Success

Enable Success: ☒

Success Queue: Test_Success

Compress: ☐

Failure

Enable Failure: ☒

Failure Queue: Test_Failure

Compress: ☐

Cancel

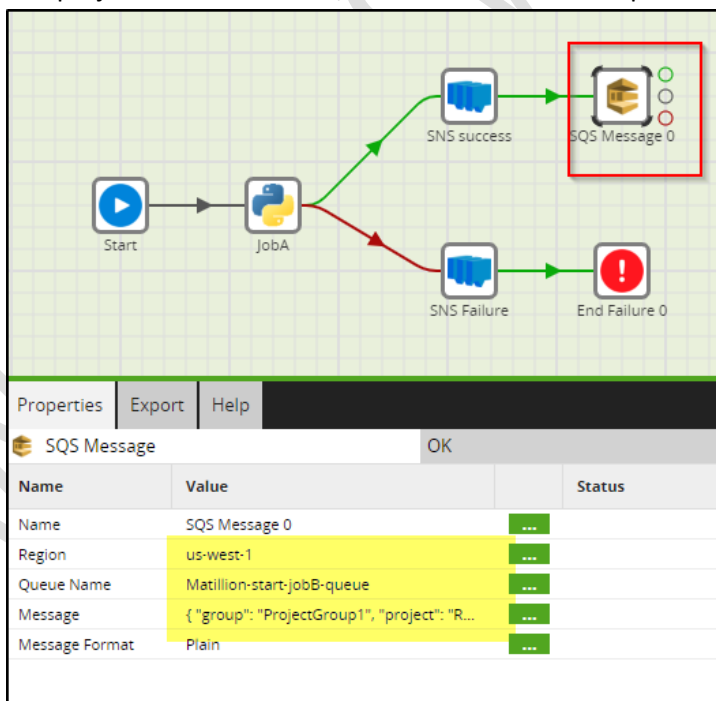
Enable Cancel: ☐

Cancel Queue: queue name

OK Cancel

5. Add SQS message component:

5.1. To our SNS project that we created, add SQS MESSAGE component for success.



Region – select you AWS region

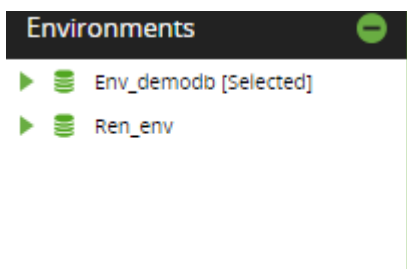
Queue Name: this is the SQS queue we created as a listener

Message: Add below script and update accordingly

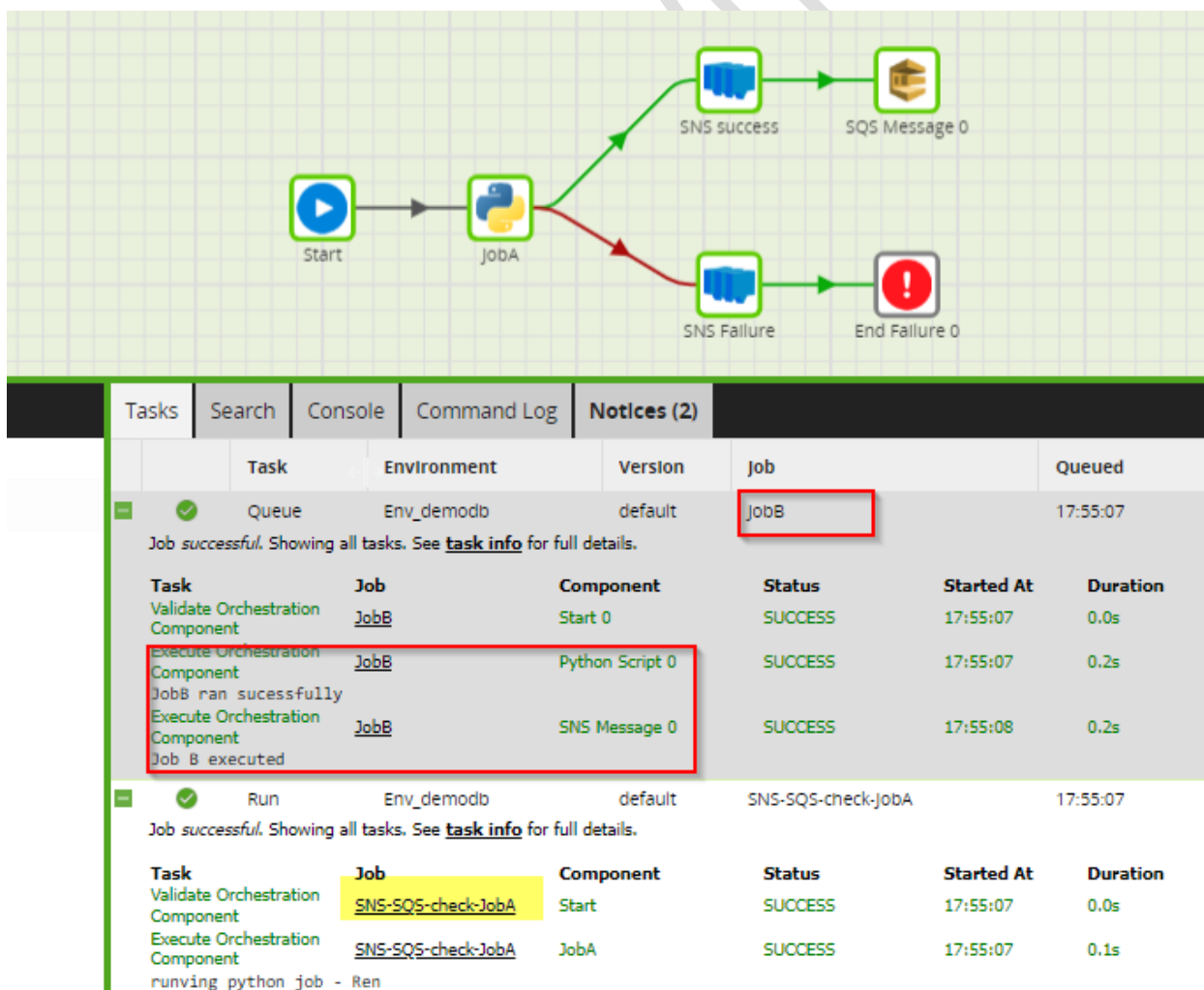
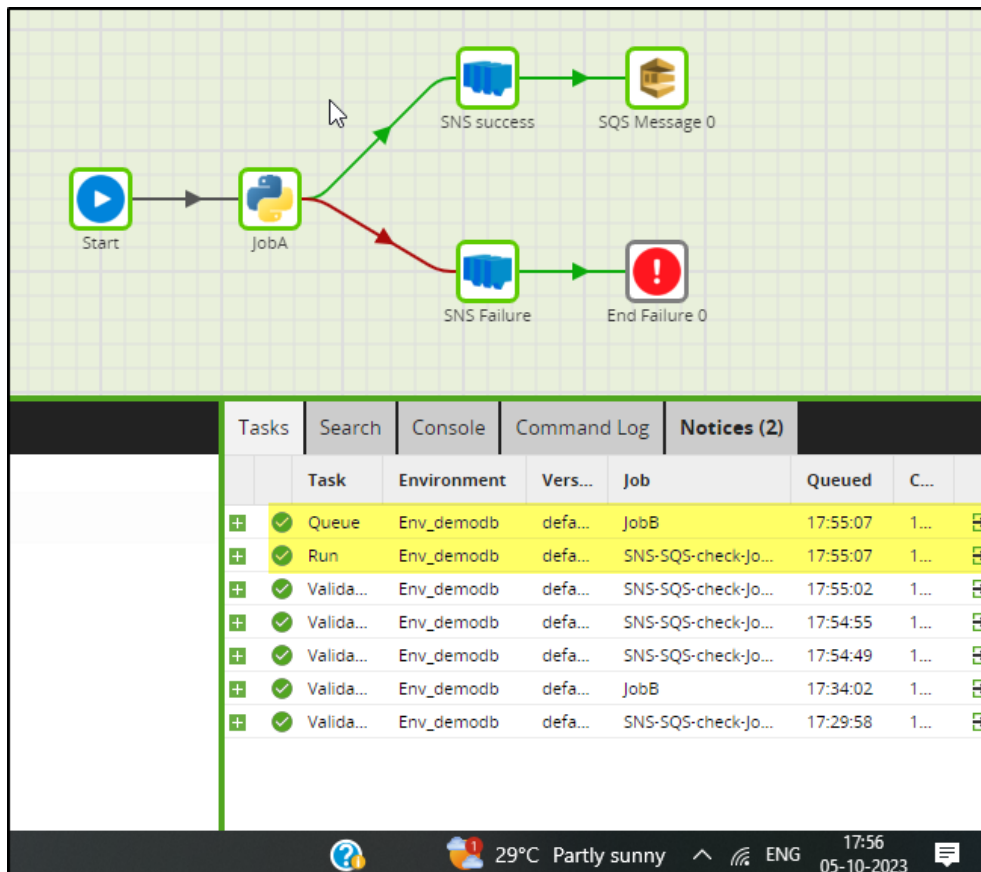
```
{
  "group": "ProjectGroup1",
  "project": "Ren_Project",
  "version": "default",
  "environment": "Env_demodb",
  "job": "JobB"
}
```



Environment is the name of your matillion environment



6. Now run Job:



Received emails for Jobs executed.

Primary		Promotions	Social
<input type="checkbox"/>	☆ SNS-suc., Ren-SNS. 8	Matillion Job -SNScheck-success - Job successfull!! Will trigger job B -- If you wish to stop receivi...	
<input type="checkbox"/>	☆ SNS-succ., Ren-SNS-. 2	Job B SNS msg - Job B executed -- If you wish to stop receiving notifications from this topic, please ...	

Congratulations you have successfully configured notification for Job status in matillion using SNS and SQS.