Jean Monnet University

Ecole des Mines de Saint-Etienne

Master of Science
Cyber-Physical and Social Systems - M2

Cloud and Edge Infrastructures - Project

PROPOSE A SOLUTION ARCHITECTURE USING AWS SERVICES

Student:Arun Raveendran Nair Sheela
Mail ID:arun.raveendran@emse.fr,
raveendran.nair.arun@etu.univ-st-etienne.fr

Lecturer:Luis Gustavo NARDIN
Mail ID:luisgustavo.nardin@emse.fr

Saint-Etienne, October 2022

Contents

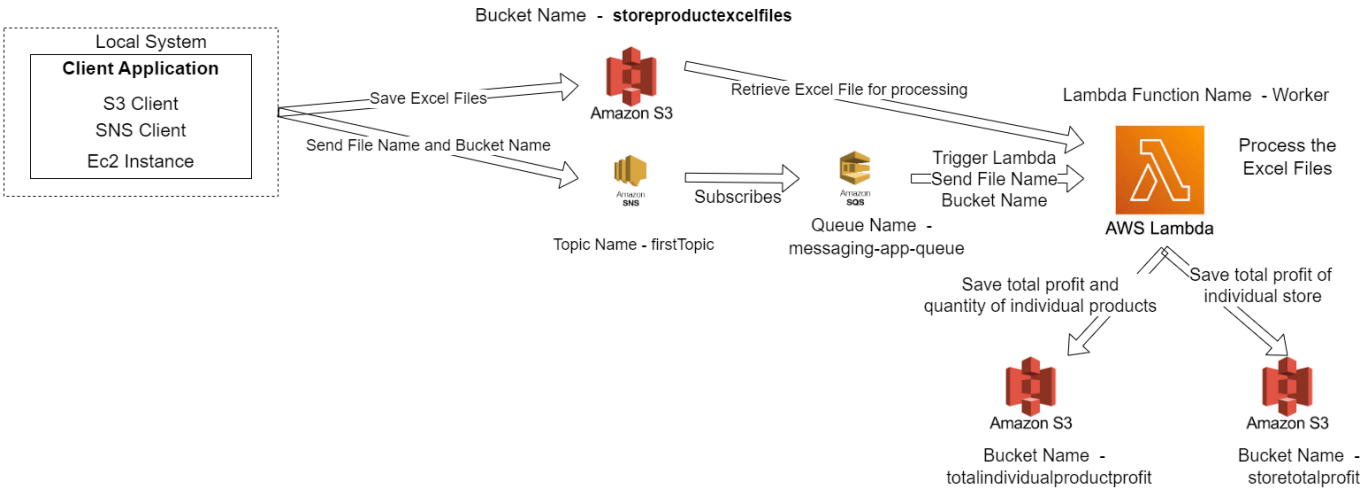


Figure 1: Solution Architecture for Client and Worker

1 Proposed Architecture Using Lambda Function

The figure 1 and 2 illustrates the solution architecture proposed to develop Client, Worker and Consolidator as per requirement. Here, I used following services for the development process.

1. Amazon Simple Storage Service(S3)[1] - It is an object storage service offering industry-leading scalability, data availability, security, and performance. Here I used four buckets which are explained below:

   (a) Bucket name - storeproductexcelfiles - This bucket is used to store the excel file from the individual store which contain product type, unit profit, unit quantity and unit cost. The excel files are send to this bucket via Client Application.

   (b) Bucket name - totalindividualproductprofit - This bucket contains the total profit and quantity of individual product from each stores. This bucket is connected to the Worker Application.

   (c) Bucket name - storetotalprofit - This bucket stores the total profit of each store. This bucket is connected to the Worker Application.

   (d) Bucket name - finalconsolidatoroutput - This bucket store excel files of total profit of all the storea and also the total profit and quantity of individual product from all the store. This bucket is connected to the Consolidator Application.

2. Amazon Simple Notification Service[2] - Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel.

   Here, I created a topic name called "firstTopic" and also a SNS client is running in the Client Application to send the file name and bucket name. Also, this service is coupled with Simple Queuing System(SQS) to make more powerful messaging service. Moreover using this the lambda function named Worker was triggered.

   Name of the Subscribers -

   - SQS name - messaging-app-queue

3. Amazon Simple Queuing System(SQS [3] - 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.

   I created the queue name called "messaging-app-queue" and the message was coming from the SNS service(topic name - "firstTopic"). Using this service a lambda function named "testWorker" is triggered and also feed the bucket name and filename to the same Worker lambda function.

[1] https://aws.amazon.com/s3/
[2] https://aws.amazon.com/sns/
[3] https://aws.amazon.com/sqs/

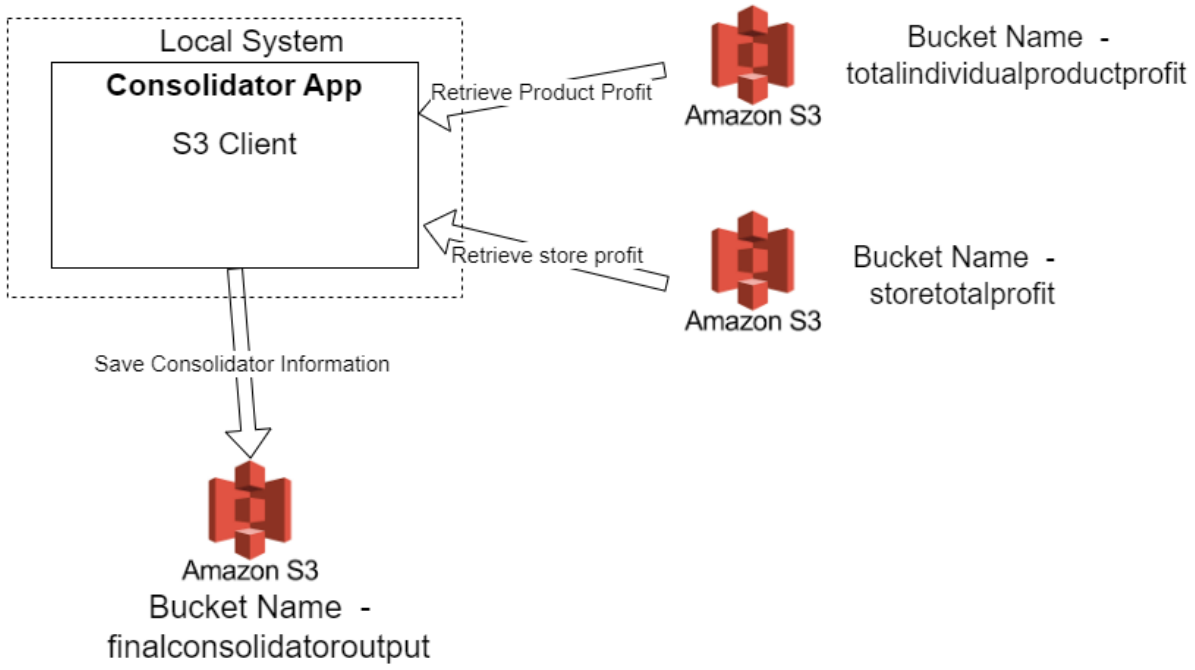


Figure 2: Solution Architecture for Consolidator

4. AWS Lambda Function[4] - AWS Lambda is a serverless, event-driven computer service that lets you run code for virtually any type of application or backend service without provisioning or managing servers.

   Here, the worker application is run as a lambda function which process csv files from all the stores and then the output is stored in the S3 buckets as csv files.

1.1 Justification for the Selected Services -

- Simple Notification Service - Here SNS is used to send the filename and bucket name to the Worker lambda function via SQS service. and also used to trigger the same lambda function to process the excel files.
- Simple Queue Service - Here SQS subscribe the message from SNS and Worker Lambda function receives this message to get the file name and bucket name.This helps to handle multiple request from different stores at the same time.
- S3 - Here I used four buckets and delete the contents in the bucket after processing. I only maintain the final consolidated file(From the application called Consolidator) which will be saved in the bucket named as "finalconsolidatoroutput".
- Lambda Function - Lambda function is the best choice for event driven application. Also, billings for the lambda function is based on the invocations. Also we only need to pay for what we are using.Moreover, the lambda function has strong security support from AWS.

Here I coupled SNS and SQS service So a message published to an SNS topic is distributed to a number of SQS queues in parallel. By using this pattern, we can build applications that take advantage parallel, asynchronous processing. So the data from multiple stores can be processed at the same time with the help of this architecture.

2 Alternate Architecture Using EC2

The figure 3 describes the alternate architecture of Worker and Client application and here Worker is run as a java application in Ec2 instance. This architecture is not recommended for this application.
Here I replace the lambda function with a Java application which is running on the EC2 instance.This application is run in a infinite loop where every 3 seconds it checks in the SQS("messaging-app-queue) for the availability of the new message. During this search if any new message was found then retrieve that message(which contain file name and bucket name) with help sqs client and run the worker application and save the processed files in the buckets.

[4]https://aws.amazon.com/lambda/

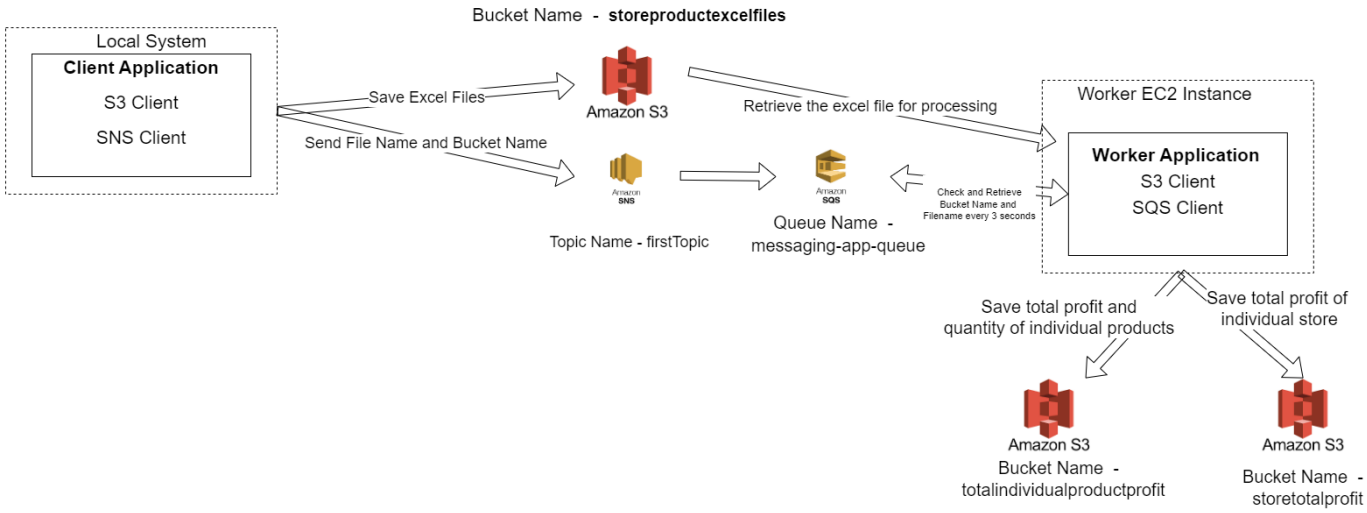


Figure 3: Solution Architecture for Worker - EC2 version

2.1 Justification

Here also I coupled SNS and SQS for better delivery of messages to the Java application. And EC2 instance is running for 24 hours so the java application doesn't miss any request from the client but it more expensive.

3 Comparison of Worker Application run as a Lambda function and as Java application

In terms of Management - For ec2 instance User decides all aspects of the application from designing to deployment. But In lambda user only give code and AWS will handle rest of the things.

In term of cost - For ec2 instance price is based on the running time and selected configuration of the virtual machine. But for lambda, Pay per execution and also the compute time

Performance - Ec2 instance performance is unlimited. From ec2 instance, we can get all the feature that we get from a local computer system.Also configuration of Virtual machines are highly customizable. So we can choose our configuration based on the application complexity and RAM memory it took to run. But in case the Lambda function , runtime is limited to only 15 minutes. So it is not suitable for the application which involve complex computation and also memory cannot be used for than 3008 MB.

Security - For Ec2 instance, it is user responsibility to provide sufficient security for the application deployed.But for lambda function AWS provide the needed security with help of IAM service.

When we consider the worker application, it is best to run as lambda function from my choice due to the following reasons:

1. It is an event driven application. So the application will not run for 24 hours. Whenever a event occurred that means a new file is uploaded to the bucket then lambda file triggered and process the files.
2. Also, Lambda function can run in parallel with help another AWS service called SQS. So multiple files can be processed at the same time.To enable this in Java application require more complex programming.
3. This application doesn't involve much complex computation works. Application will be faster as a lambda function than a java application run in Ec2 instance.
4. This lambda function will run only an event occured. So price is comparatively less than a Ec2 instance which need to be run for 24 hours with out any purpose.
5. Lambda function has inbuilt security feature from AWS. SO there is no need to provide external security. But in the case java application, we need to provide security for restricting unauthorized entry.

4 Results

4.1 Results after implementing the Proposed Architecture

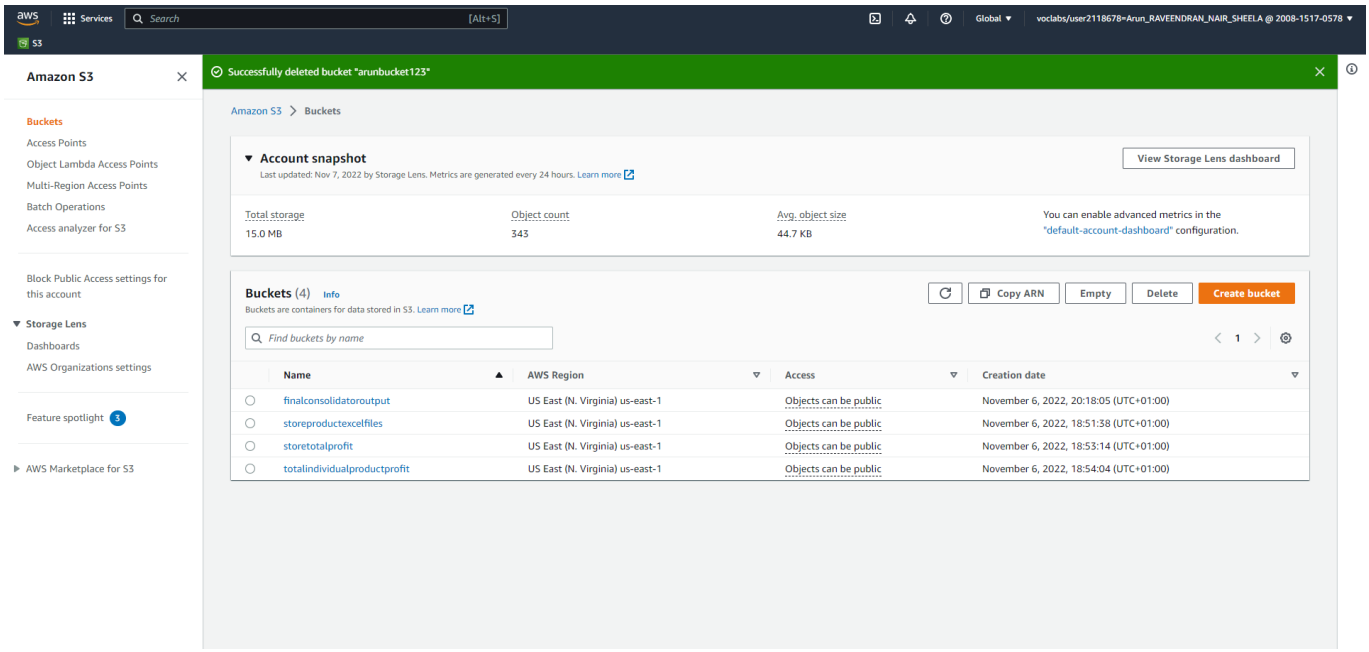


Figure 4: Screenshot to show total bucket list for this application

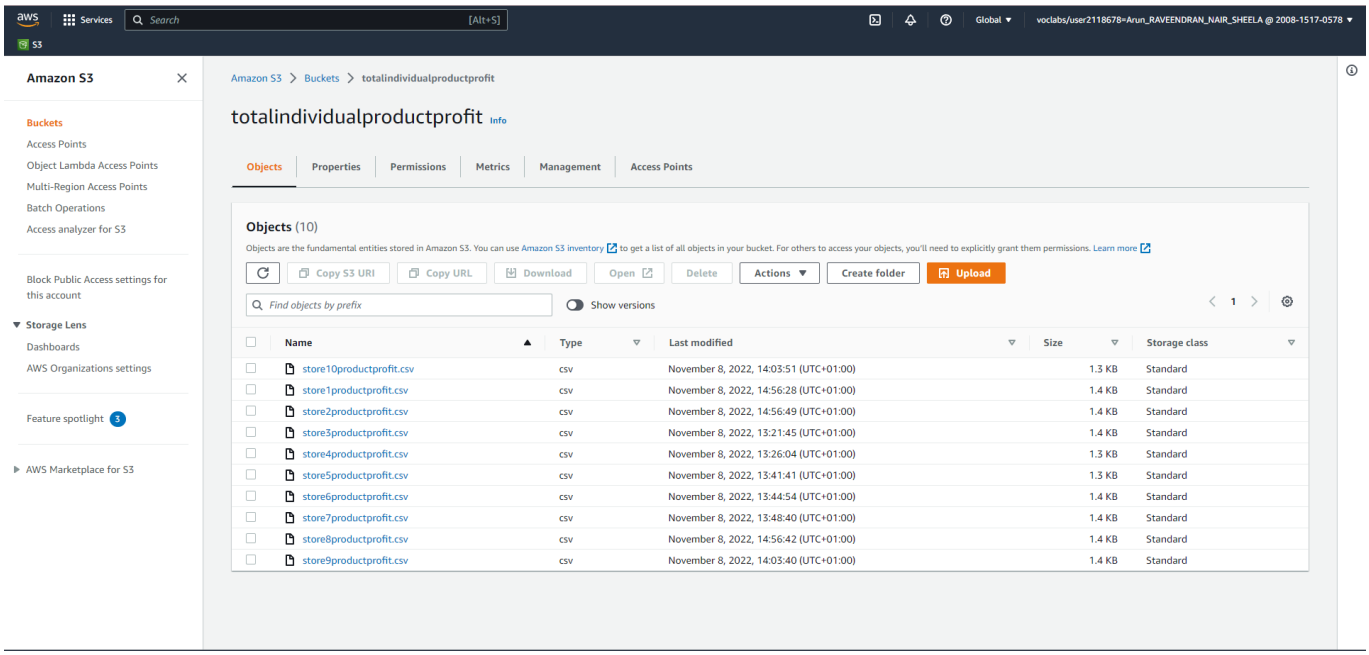


Figure 5: Screenshot of csv files created in the bucket name called totalindividualproductprofit

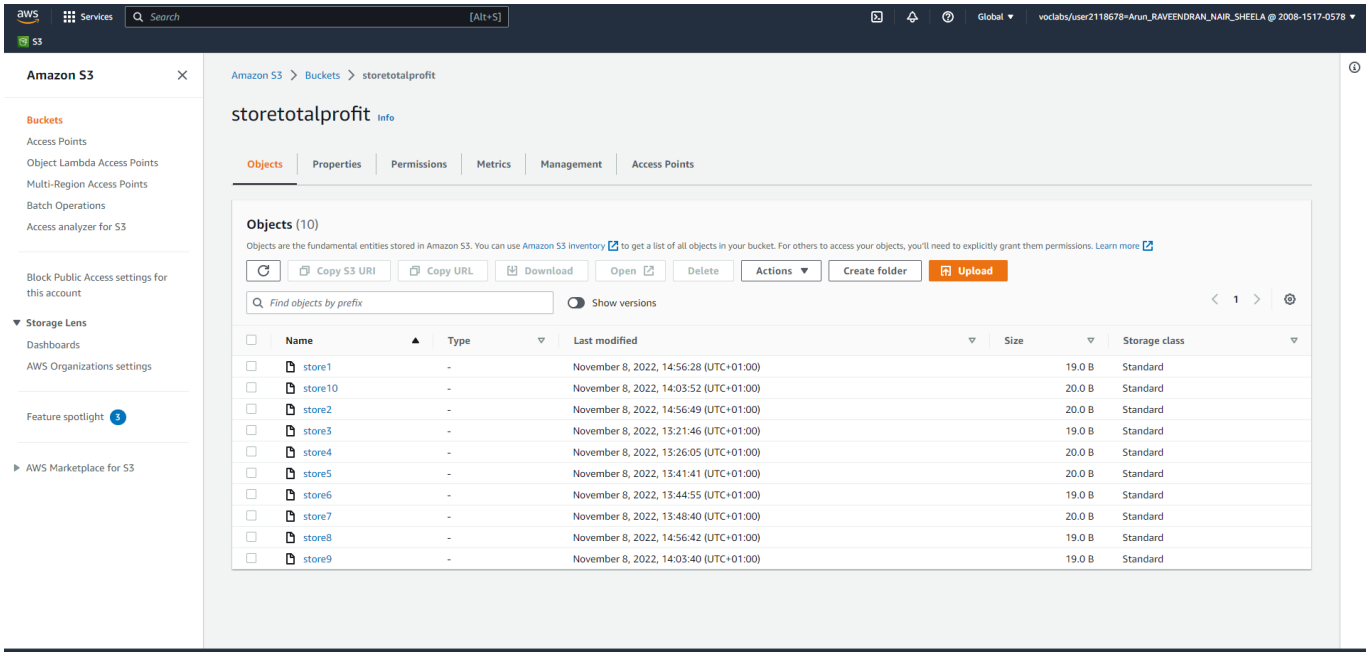


Figure 6: Screenshot of storetotalprofit to store tottal profit of individual stores

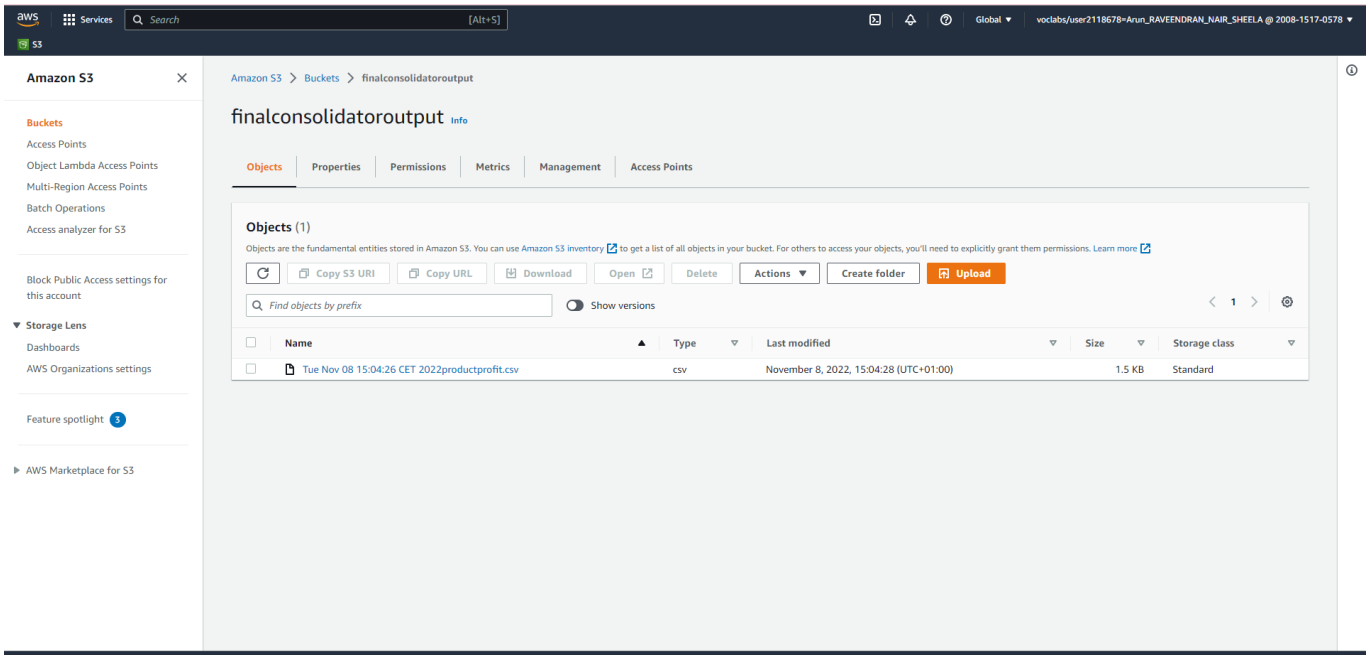


Figure 7: Screenshot of excel file created after running consolidator application

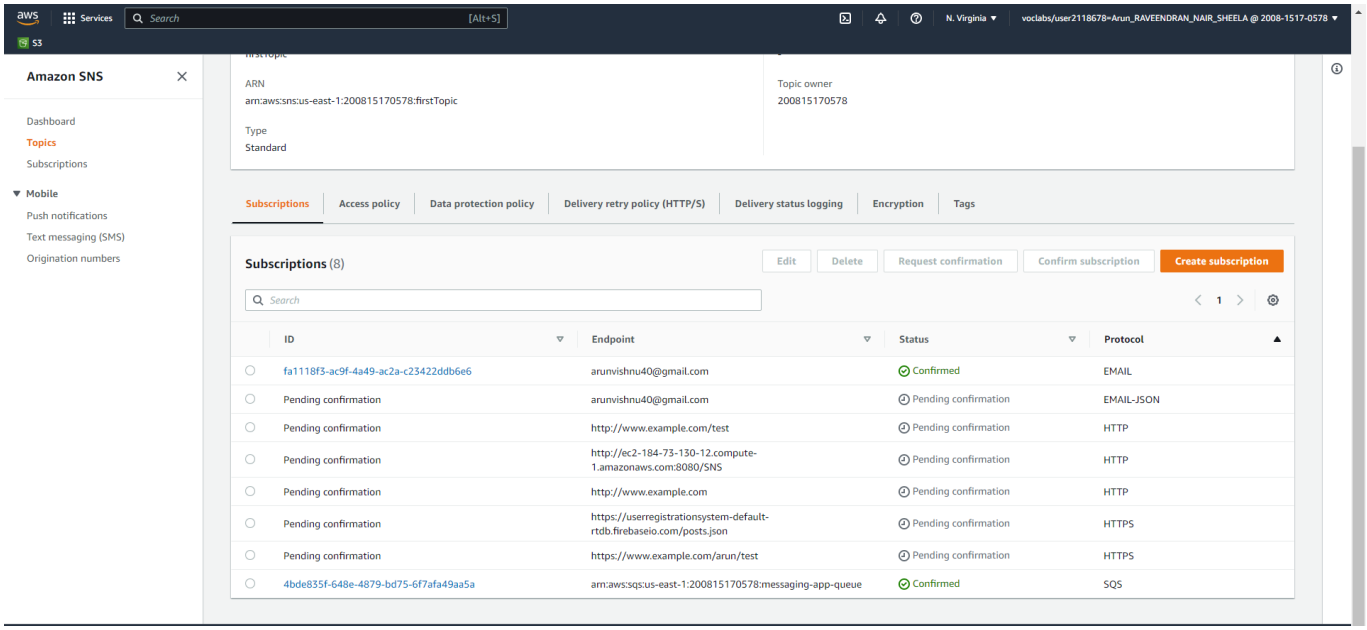


Figure 8: Screenshot SNS topic name and its subscribers

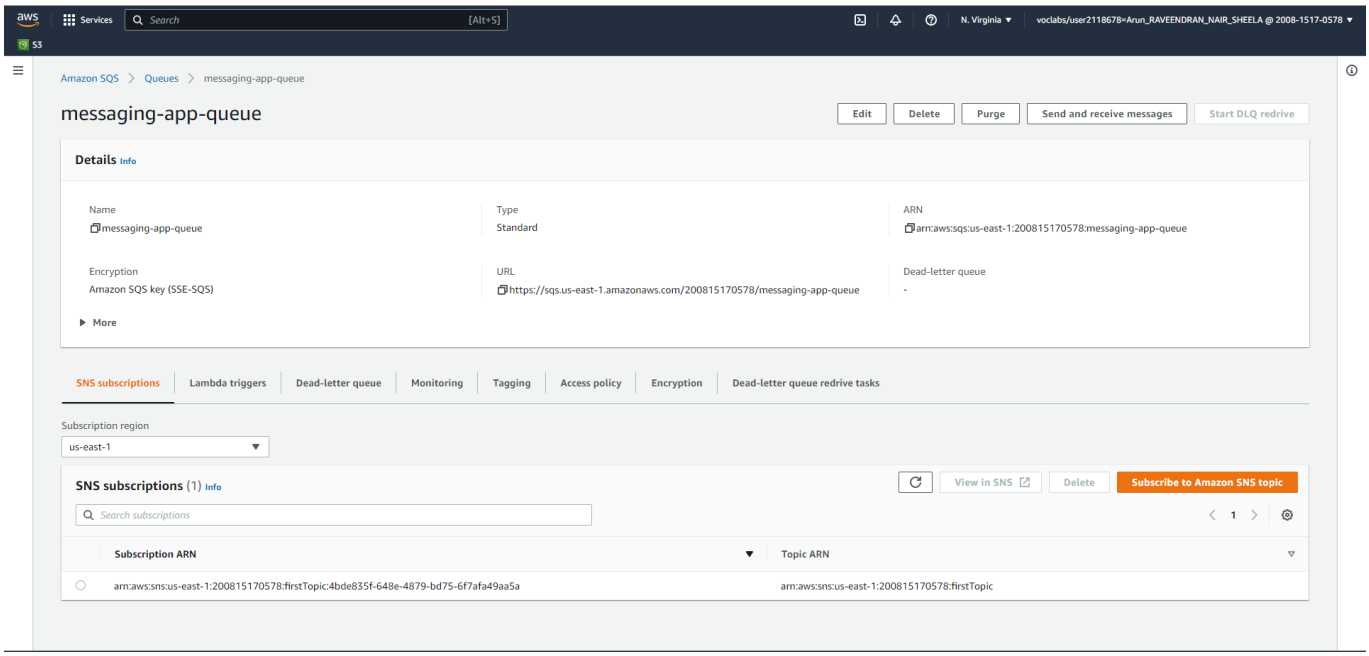


Figure 9: Screenshot of SQS

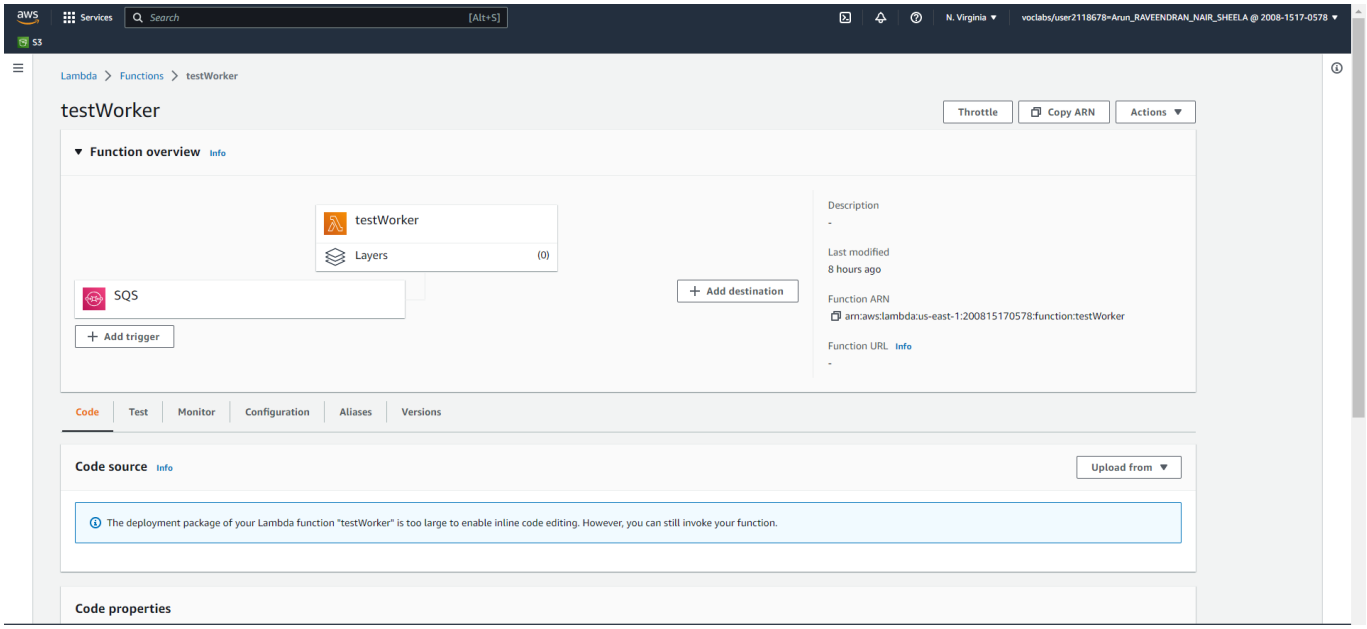
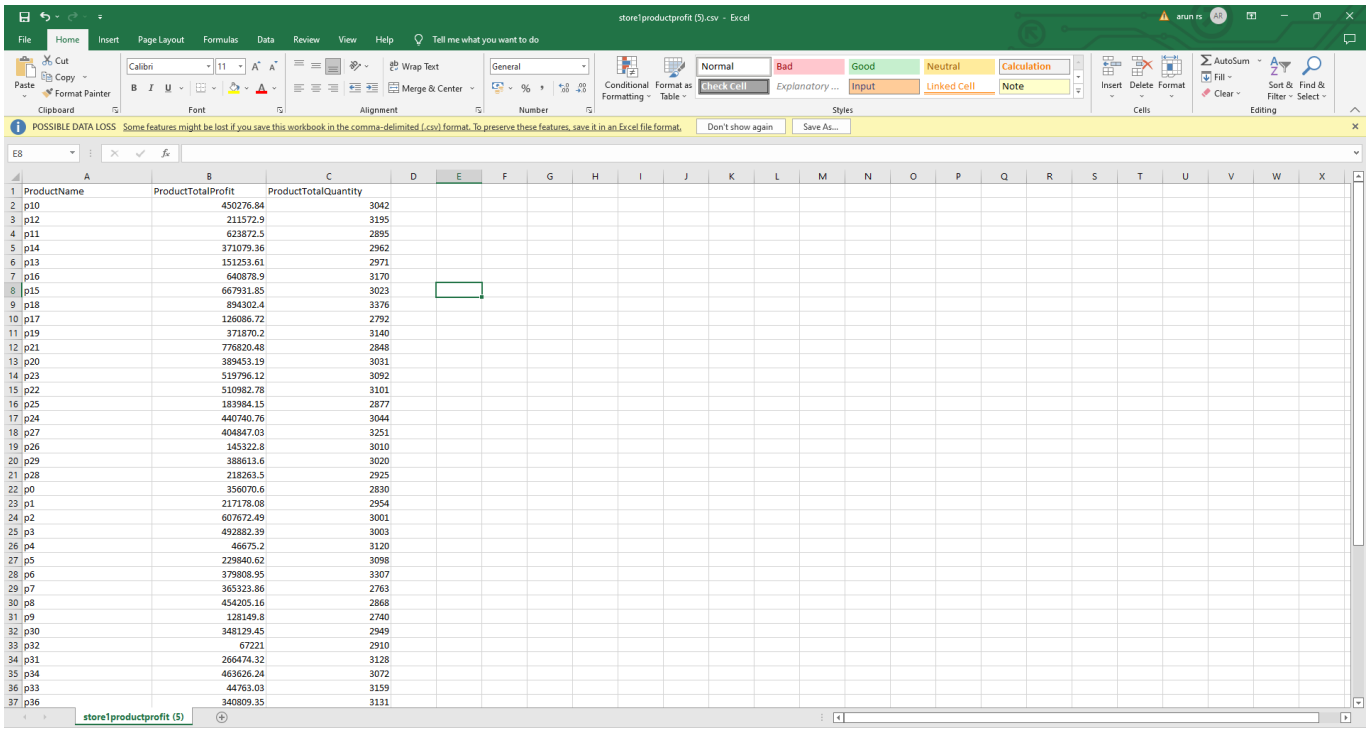
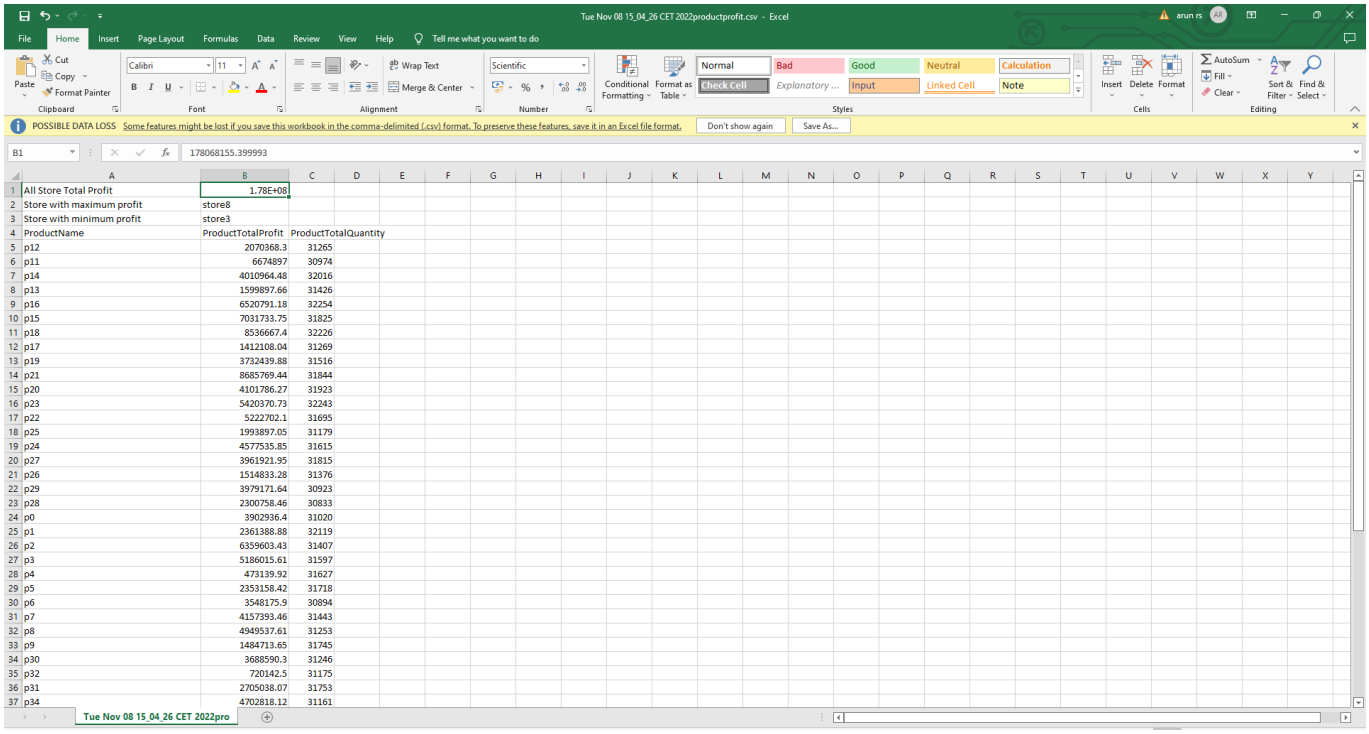


Figure 10: Screenshot of Lambda and its trigger(SQS)



	ProductName	ProductTotalProfit	ProductTotalQuantity
2	p10	450276.84	3042
3	p12	211572.9	3195
4	p11	623872.5	2895
5	p14	371079.36	2962
6	p13	151253.61	2971
7	p16	640878.9	3170
8	p15	667931.85	3023
9	p18	894302.4	3376
10	p17	126086.72	2792
11	p19	371870.2	3140
12	p21	776820.48	2848
13	p20	389453.19	3031
14	p23	519796.12	3092
15	p22	510982.78	3101
16	p25	183984.15	2877
17	p24	440740.76	3044
18	p27	404847.03	3251
19	p26	145322.8	3010
20	p29	388613.6	3020
21	p28	218263.5	2925
22	p0	356070.6	2830
23	p1	217178.08	2954
24	p2	607672.49	3001
25	p3	492882.39	3003
26	p4	46675.2	3120
27	p5	229840.62	3098
28	p6	379808.95	3307
29	p7	365323.86	2763
30	p8	454205.16	2868
31	p9	128149.8	2740
32	p30	348129.45	2949
33	p32	67221	2910
34	p31	266474.32	3128
35	p34	463626.24	3072
36	p33	44763.03	3159
37	p36	340809.35	3131

Figure 11: Screenshot excel file created after running the Worker Application in the bucket called totalindividual-productprofit



	A	B	C
1	All Store Total Profit	1.78E+08	
2	Store with maximum profit	store8	
3	Store with minimum profit	store3	
4	ProductName	ProductTotalProfit	ProductTotalQuantity
5	p12	2070368.3	31265
6	p11	6674897	30974
7	p14	4010964.48	32016
8	p13	1599897.66	31426
9	p16	6520791.18	32254
10	p15	7031733.75	31825
11	p18	8536667.4	32226
12	p17	1412108.04	31269
13	p19	3732439.88	31516
14	p21	8685769.44	31844
15	p20	4101786.27	31923
16	p23	5420370.73	32243
17	p22	5222702.1	31695
18	p25	1993897.05	31179
19	p24	4577535.85	31615
20	p27	3961921.95	31815
21	p26	1514833.28	31376
22	p29	3979171.64	30923
23	p28	2300758.46	30833
24	p0	3902936.4	31020
25	p1	2361388.88	32119
26	p2	6359603.43	31407
27	p3	5186015.61	31597
28	p4	473139.92	31627
29	p5	2353158.42	31718
30	p6	3548175.9	30894
31	p7	4157393.46	31443
32	p8	4949537.61	31253
33	p9	1484713.65	31745
34	p30	3688590.3	31246
35	p32	720142.5	31175
36	p31	2705038.07	31753
37	p34	4702818.12	31161

Figure 12: Screenshot of excel file created after running Consolidator Application in the bucket name called - finalconsolidatoroutput

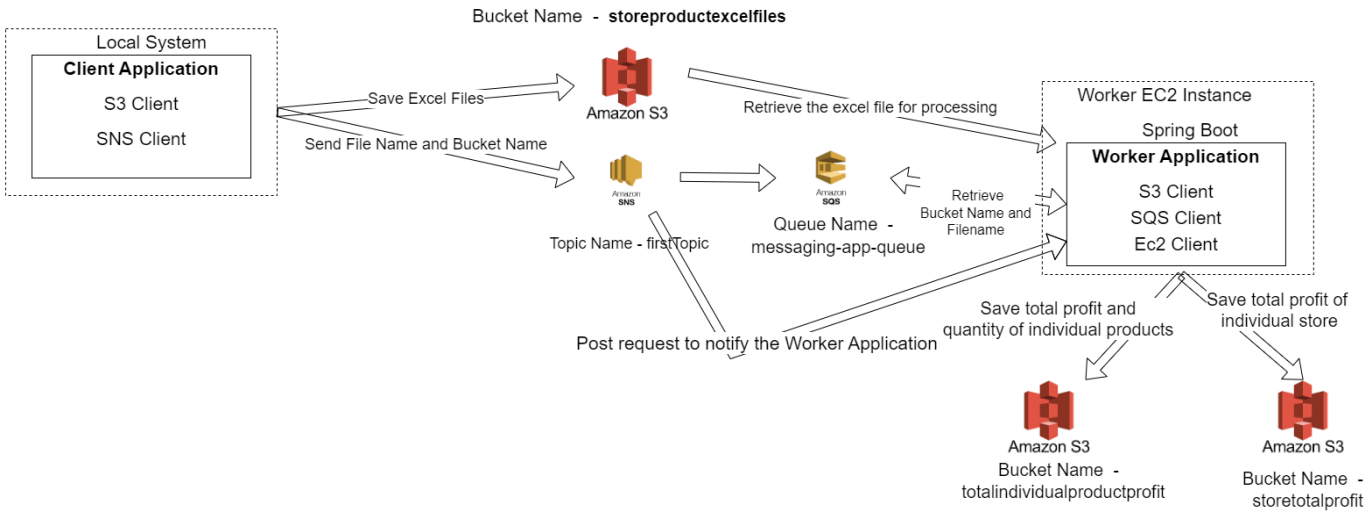


Figure 13: Another approach of Solution Architecture for Worker - EC2 version - Not implemented - (for reference)

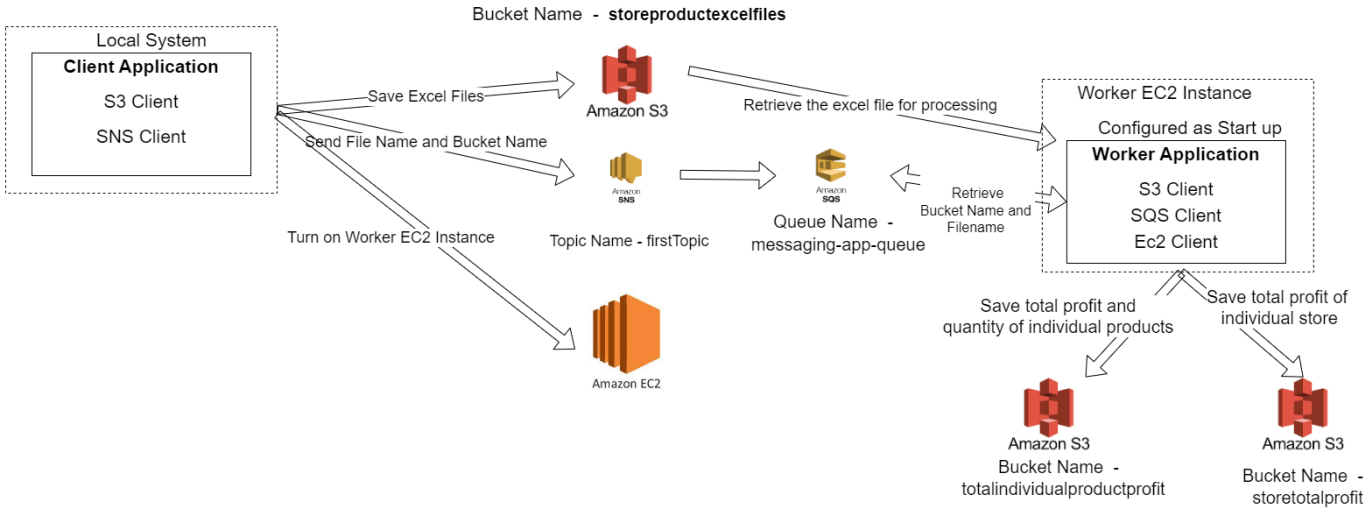


Figure 14: Another approach of Solution Architecture for Worker - EC2 version - Not implemented (for reference)