Cloud Computing In-depth Study

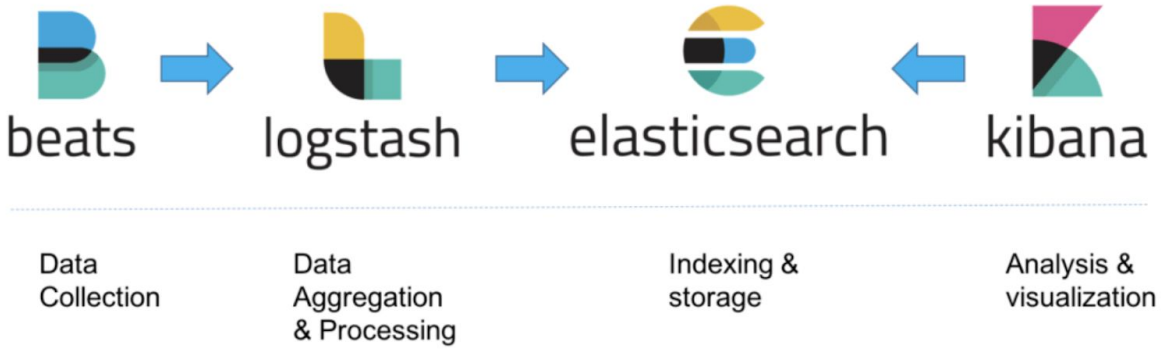
Demo Link:

https://youtu.be/tBtJOxrk-fk

Task 1: ELK stack

ELK STACK:

ELK stands for ElasticSearch, Log Stash and Kibana. Log stash is a server side program(log aggregator,data transformer and a data pipeline) that essentially gathers data from multiple sources to push it on to a stack (Elastic Search). ElasticSearch is the search and analytics engine and Kibana is the data visualization tool which we can automatically integrate with elastic search to visualize the log data. ElasticSearch, Log Stash and Kibana are all open-source products developed and managed by 'Elastic'. A new addition to the package is 'Beats' which is a family of light-weight,single-purpose data-shippers. This can be used for activities such as tailing a file etc.In conclusion, we can use Beats and Logstash for data collection and processing, Elasticsearch to index and store the data, and Kibana to provide a user interface for querying the data and visualizing it.



References:

https://logz.io/learn/complete-guide-elk-stack/

https://www.elastic.co/what-is/elk-stack

https://logz.io/blog/elk-mac/

Steps Followed to setup ELK Stack on my local machine:

Prerequisite:

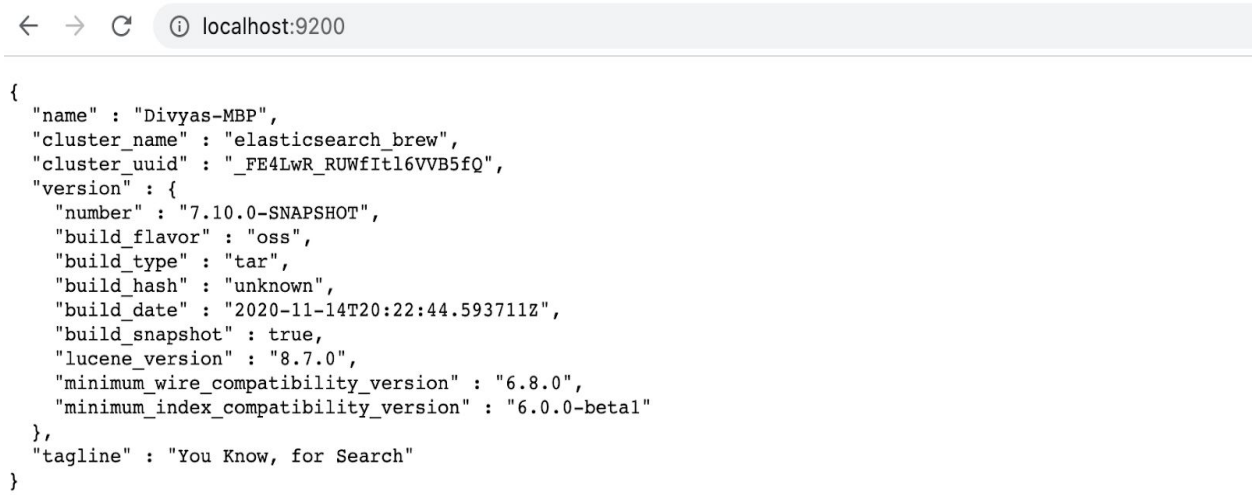
ELK Stack requires Java 8 to be installed
command: java -version

1. Install Elastic Search

Commands:
brew install elasticsearch
brew info elasticsearch
brew services start elasticsearch

By default: The elastic search port number is 9200.
Url: http://localhost:9200

Screenshot:

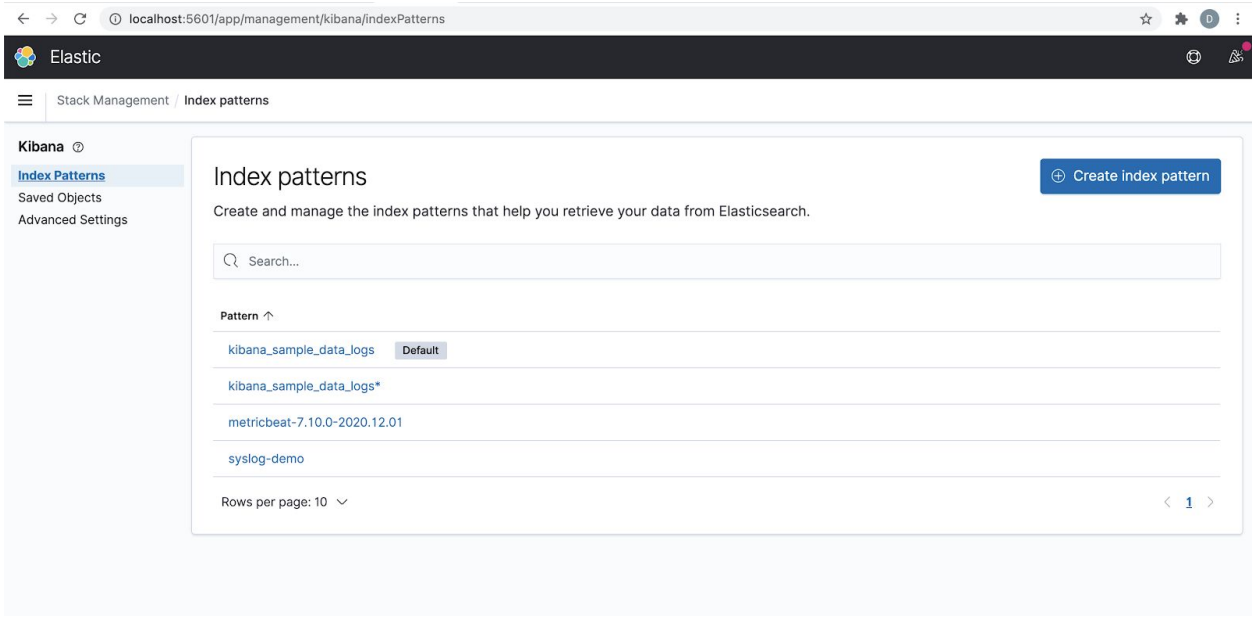


2. Install Kibana

commands:
brew install kibana
brew services start kibana
brew services list

We should the kibana.yml file to uncomment the port number(5601) which is at /usr/local/etc/kibana/kibana.yml

Screenshots:



3. Installed Logstash and Metricbeat

commands:

brew install logstash
brew services start logstash
brew install metricbeat
brew services start metricbeat

screenshot:

```
divya@Divyas-MBP ~ % brew services list
Name          Status  User  Plist
elasticsearch started divya /Users/divya/Library/LaunchAgents/homebrew.mxcl.elas
ticsearch.plist
kibana        started divya /Users/divya/Library/LaunchAgents/homebrew.mxcl.kiba
na.plist
logstash      started divya /Users/divya/Library/LaunchAgents/homebrew.mxcl.logs
tash.plist
metricbeat    started divya /Users/divya/Library/LaunchAgents/homebrew.mxcl.metr
icbeat.plist
unbound       stopped
divya@Divyas-MBP ~ %
```

Shipping some data to ELK stack:

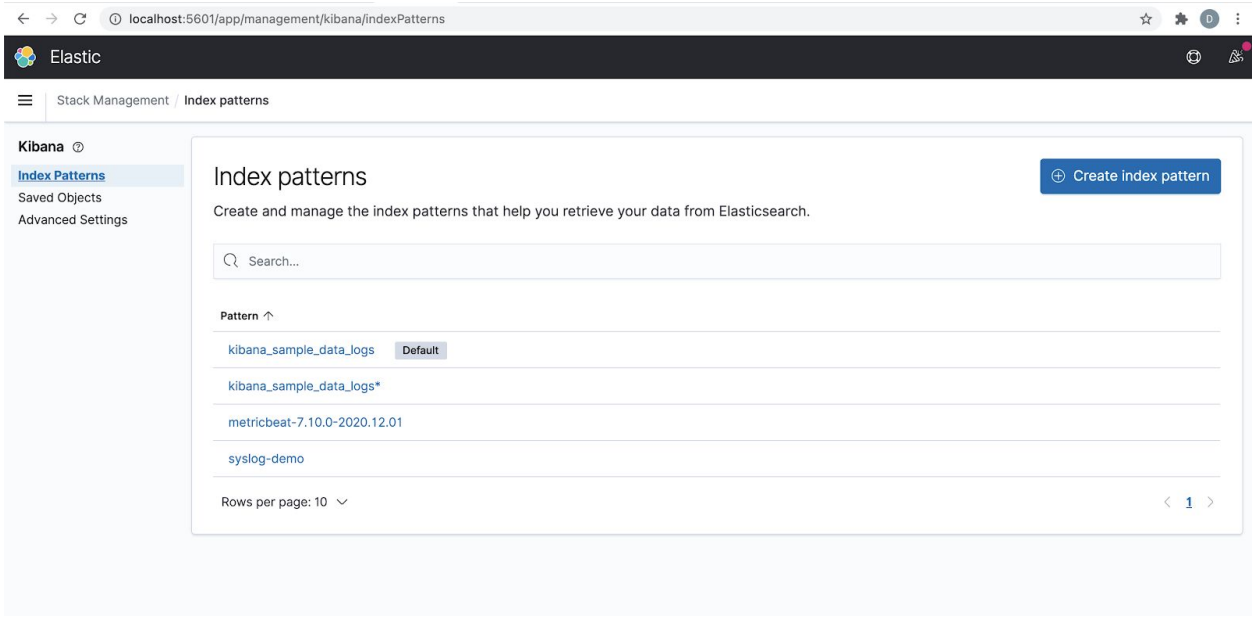
To achieve the same, I added a logstash configuration file namely '/etc/logstash/conf.d/syslog.conf' and added the following configuration to it to

essentially accumulate the system logs(from system log files such as "/var/log/messages", "/var/log/syslog") to the elastic-search index--'syslog-demo'

Code:

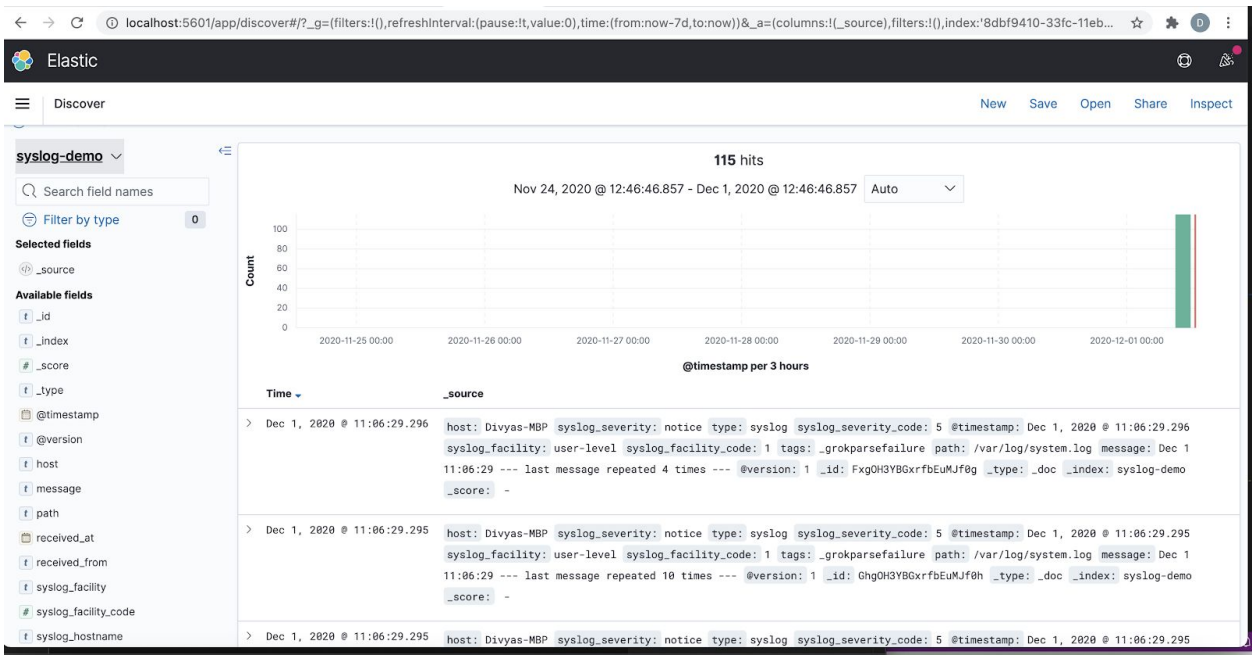
input { file { path => ["/var/log/*.log", "/var/log/messages", "/var/log/syslog"] type => "syslog" }}filter { if [type] == "syslog" { grok { match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: %{GREEDYDATA:syslog_message}" } add_field => ["received_at", "%{@timestamp}"] add_field => ["received_from", "%{host}"] } syslog_pri { } date { match => ["syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss"] } }}output { elasticsearch { hosts => ["127.0.0.1:9200"] index => "syslog-demo" } stdout { codec => rubydebug }}

Output: (index --syslog-demo seen in the elastic search home page)



Correspondingly , the kibana visualizations for the syslog index over some period of time:

Screenshot of ElasticSearch logs and Kibana visualisations:



Task 2: Kafka and Spark Setup

Kafka:

Kafka is primarily used to build real-time streaming data pipelines and applications to the incoming data streams. It is a combination of messaging, storage, and stream processing systems to allow publish-subscribe,storage and analysis of both historical and real-time data.

Installation on local machine:

After downloading kafka, I started the zookeeper service and kafka-brokers to start the kafka application using these commands:

bin/zookeeper-server-start.sh config/zookeeper.properties

bin/kafka-server-start.sh config/server.properties

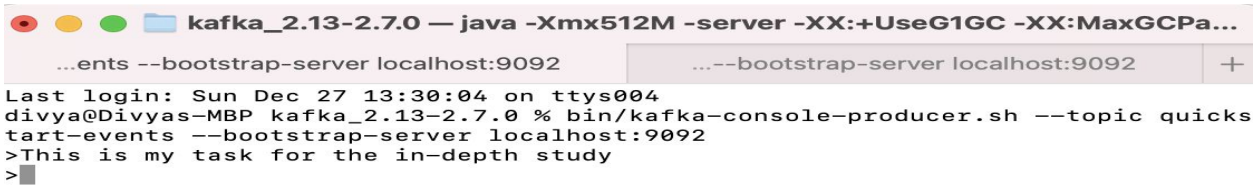
I created a topic namely, quickstart-events using this command :

bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092

I then started the producer in a new tab using this command:(To publish events)

bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092

Screenshot:

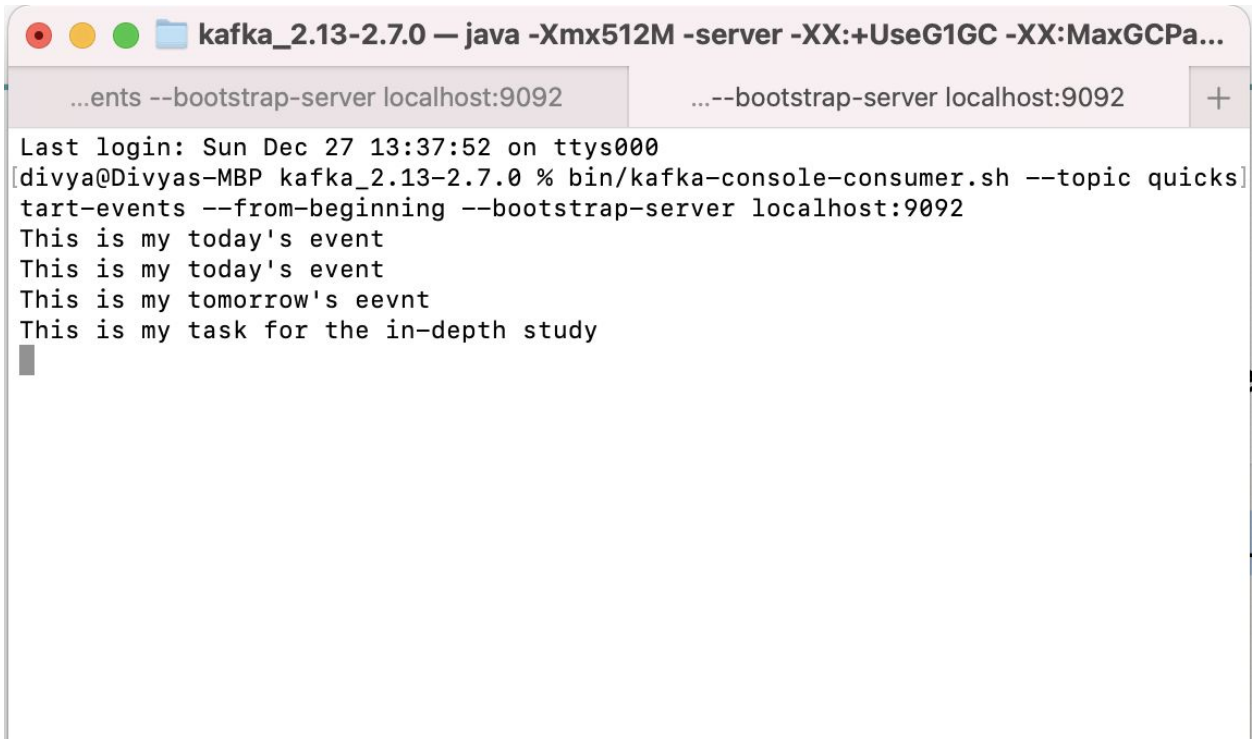


I then started the consumer in a new tab using this command:(To consume events from the **beginning**)

bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning

--bootstrap-server localhost:9092

Screenshot:



Reference:
https://kafka.apache.org/quickstart

I repeated these kind of steps to publish-subscribe events for a kafka-topic on AWS console

Kafka setup on AWS:

AWS implements Apache Kafka through Amazon MSK(Amazon managed streaming service for Apache Kafka) and we can configure lambda to have a MSK trigger to allow MSK as an event source to AWS lambda which acts as a listener to process the incoming events.
Note:
Lambda function's event payload contains an array of records. Each array item contains details of the topic and Kafka partition identifier, together with a timestamp and **base64** encoded message

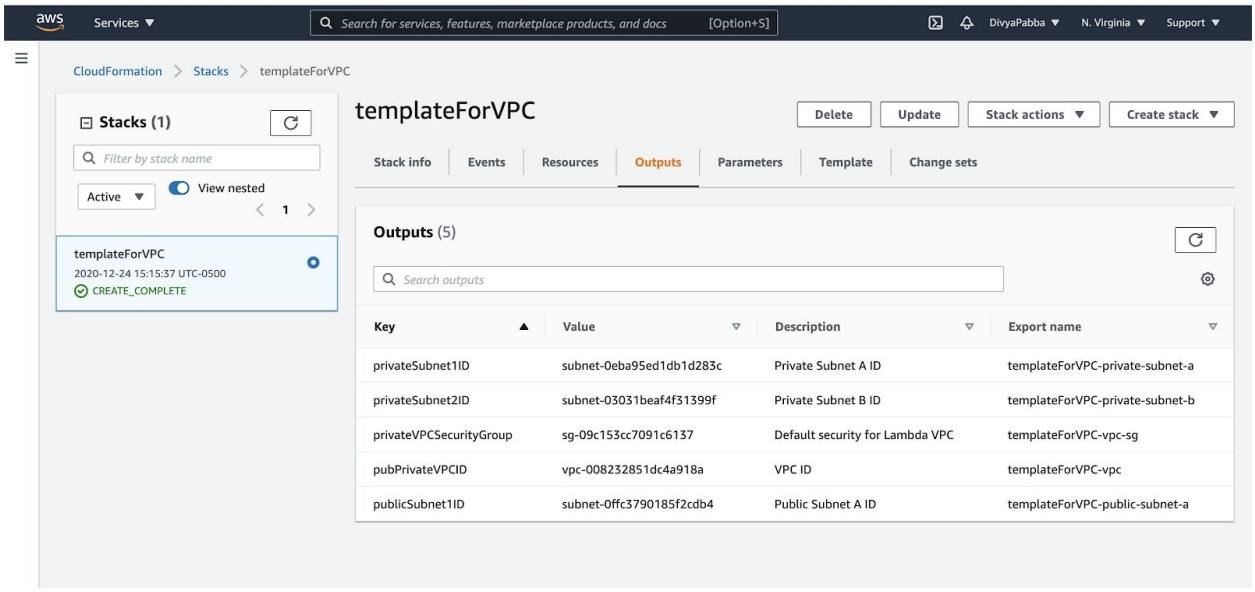
Practices and Behaviors:

For security purposes, the Amazon MSK cluster is configured in a VPC with private subnets.Now for MSK to trigger lambda,a NAT gateway is set up in the public subnet.

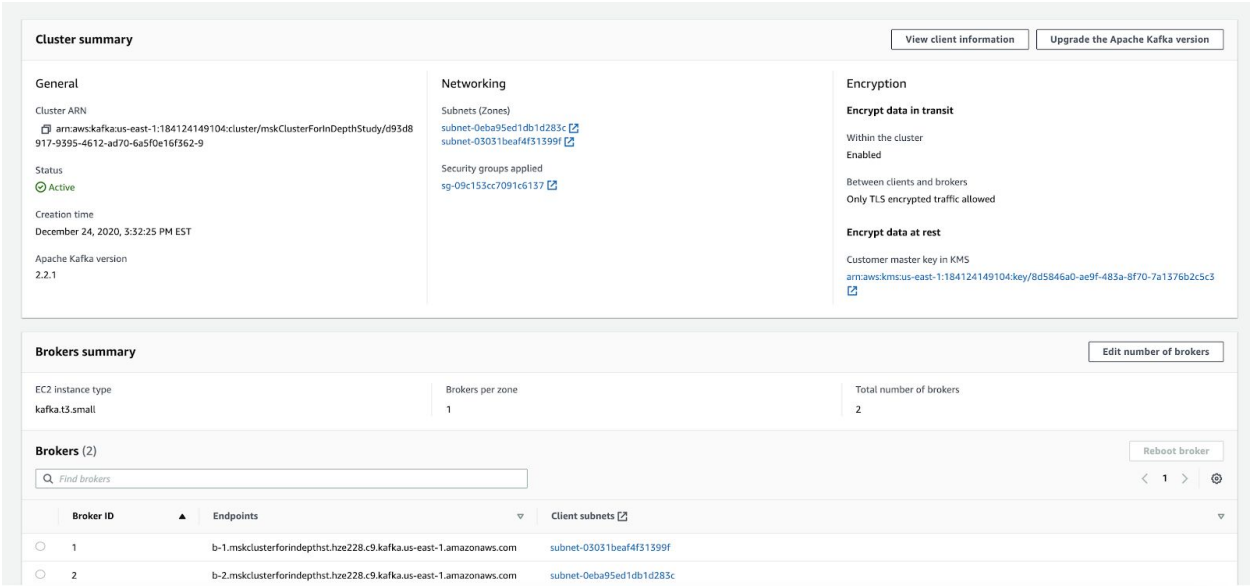
Step 1:

1. Created a MSK cluster using the CloudFormation template(present in the zip folder) with a public subnet,two private subnets and a security group and brokers configured.

   CloudFormation Output:



   MSK cluster:



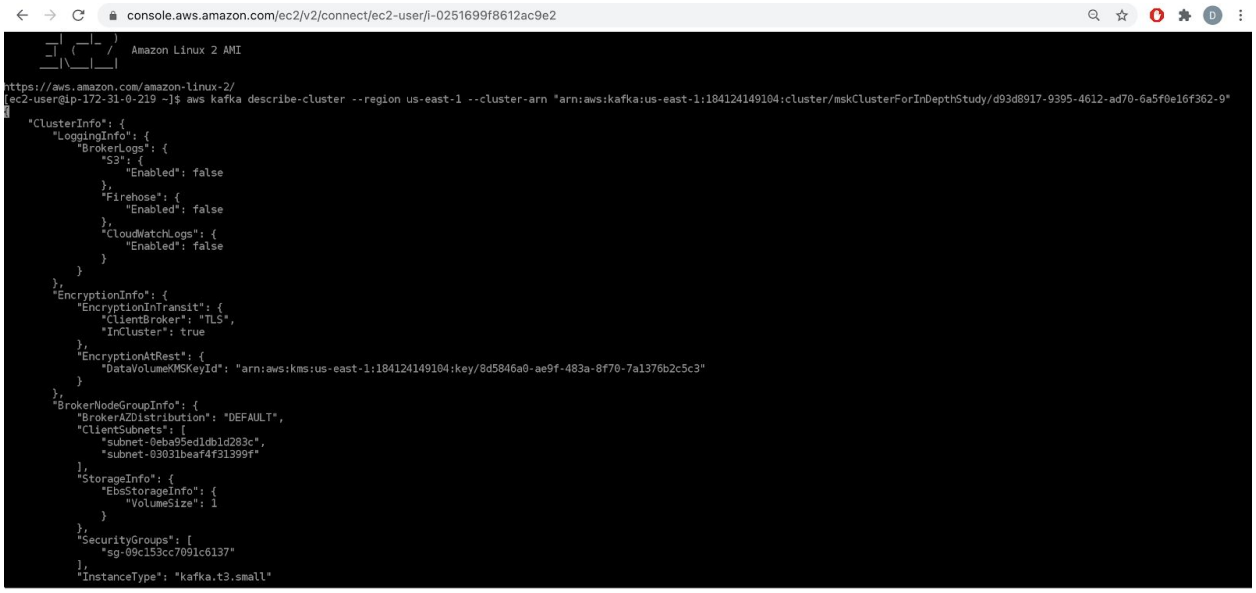
Step 2:

1. Create a Topic on the MSK cluster

   To connect to the remote MSK cluster and create a topic on the brokers, we need to set up the kafka libraries on the client machine.For the same,we can launch an EC2 instance and make a quick-connect to it through aws console.Then we can download the kafka libraries in the ec2 instance and create topics on the MSK cluster from the EC2 instance.

   Screenshot of the following command on the ec2 instance:

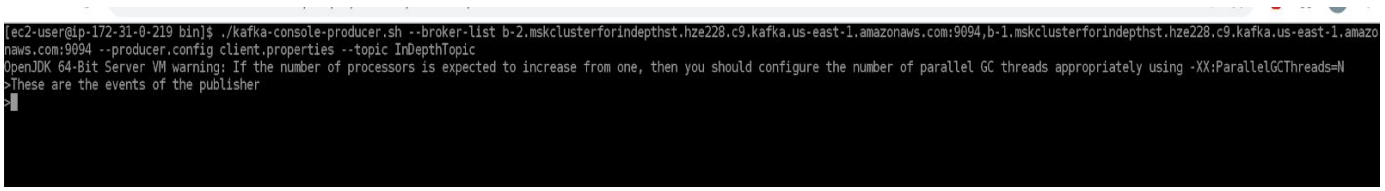
   aws kafka describe-cluster --region us-east-1 --cluster-arn "ClusterArn"



Then I created a topic namely, “InDepthTopic” on the brokers using the following command:

Topic creation:
bin/kafka-topics.sh --create --zookeeper ‘(zookeeper url)’ --replication-factor 3 --partitions 1 --topic InDepthTopic

Start the producer to publish events:
./kafka-console-producer.sh --broker-list b-2.mskclusterforindepthst.hze228.c9.kafka.us-east-1.amazonaws.com:9094,b-1.mskclusterforindepthst.hze228.c9.kafka.us-east-1.amazonaws.com:9094 --producer.config client.properties --topic InDepthTopic



Start the consumer to consume events:

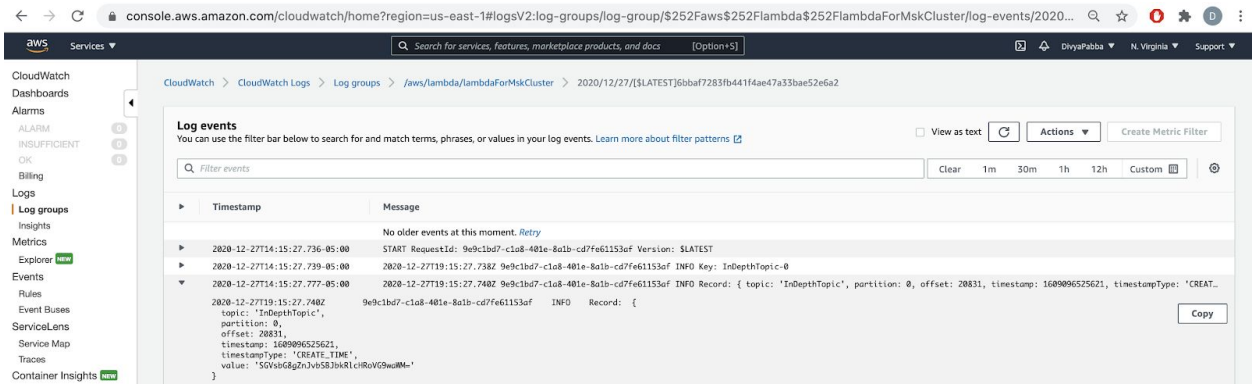
./kafka-console-consumer.sh --bootstrap-server b-2.mskclusterforindepthst.hze228.c9.kafka.us-east-1.amazonaws.com:9094,b-1.mskclusterforindepthst.hze228.c9.kakfa.us-east-1.amazonaws.com:9094 --consumer.config client.properties --topic InDepthTopic --from-beginning

```
[ec2-user@ip-172-31-0-219 bin]$ ./kafka-console-consumer.sh --bootstrap-server b-2.mskclusterforindepthst.hze228.c9.kafka.us-east-1.amazonaws.com:9094,b-1.mskclusterforindepthst.hze228.c9.kakfa.us-east-1.amazonaws.com:9094 --consumer.config client.properties --topic InDepthTopic --from-beginning
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increase from one, then you should configure the number of parallel GC threads appropriately using -XX:ParallelGCThreads=N
[2020-12-27 19:24:24,488] WARN Couldn't resolve server b-1.mskclusterforindepthst.hze228.c9.kakfa.us-east-1.amazonaws.com:9094 from bootstrap.servers as DNS resolution failed for b-1.mskclusterforindepthst.hze228.c9.kakfa.us-east-1.amazonaws.com (org.apache.kafka.clients.ClientUtils)
```

```
Hello from InDepthTopic
This is the event for today
This is my task for indepth study
These are my published events
These arr the published events
These are the events of the publisher
```

I configured a lambda to have a MSK trigger whenever messages are published to a particular topic.(code for the lambda --'lambdaforMSKCluster' can be found on the zip folder)

Output for the lambda Function:



There were around 20k+ messages on this topic..correspondingly offset value is seen.

References:

https://aws.amazon.com/blogs/compute/using-amazon-msk-as-an-event-source-for-aws-lambda/?fbclid=IwAR3Og1roADylOHRV1f6gzSDxSiG7gdHzRuU8ne9c7sS_H6sGf0OG-NHoB8A

https://docs.aws.amazon.com/msk/latest/developerguide/produce-consume.html

Spark:

It is a unified analytics engine for large-scale data processing.

To install this locally i used brew commands and set up the environment variables accordingly to submit a spark-job through jupyter notebook:

commands:
brew install apache-spark
export SPARK_HOME="/usr/local/Cellar/apache-spark/2.3.1/libexec/"

Screenshots:

```
import os
exec(open(os.path.join(os.environ["SPARK_HOME"], 'python/pyspark/shell.py')).read())
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.1
      /_/

Using Python version 3.8.3 (default, Jul  2 2020 11:26:31)
SparkSession available as 'spark'.
```

SparkContext:

```
sc
```

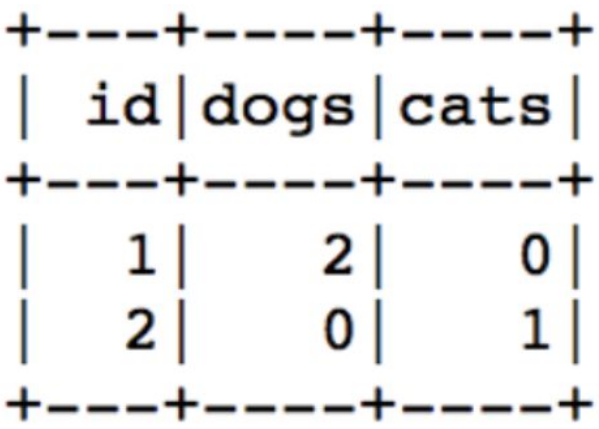
SparkContext

Spark UI
Version
`v3.0.1`
Master
`local[*]`
AppName
`pyspark-shell`

Now this is a small function written to submit a spark-job through jupyter-notebook to create a dataframe for the ‘vals’:

```
import pyspark
from pyspark.sql.session import SparkSession
spark = SparkSession.builder.appName("spark test").getOrCreate()
columns = ['id', 'dogs', 'cats']
vals = [
     (1, 2, 0),
     (2, 0, 1)
]
# create DataFrame
df = spark.createDataFrame(vals, columns)
df.show()
```

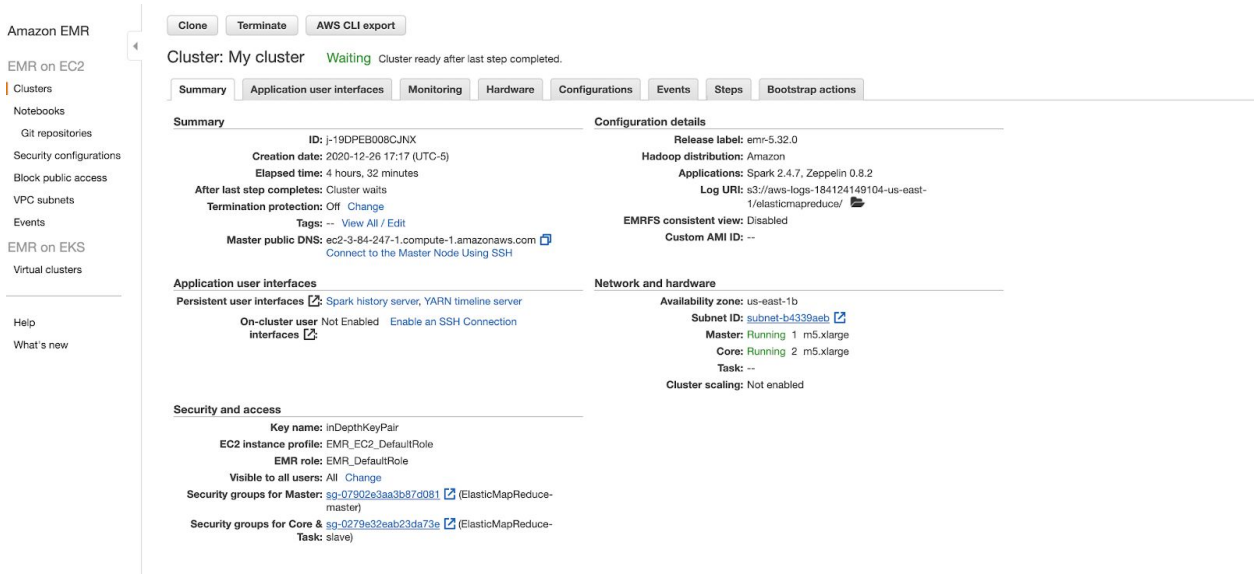
Output:



Now to submit a spark-job to the AWS EMR from remote machine/locally and from AWS console:
Goal:
To submit a spark-job to process the dataset for the StackOverFlowSurvey results dataset to the AWS EMR cluster from remote machine /locally:

Step 1:

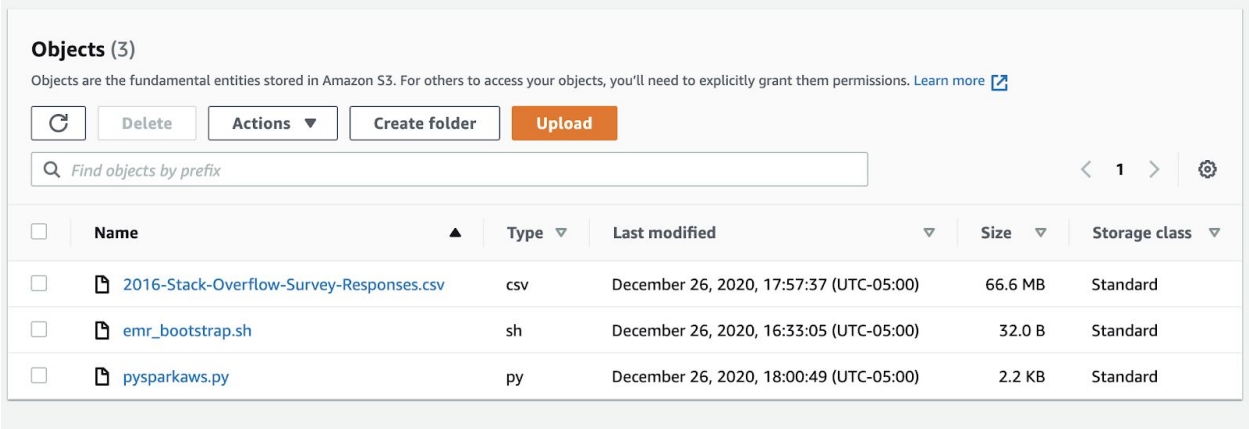
Create a EMR cluster with ‘Spark as one of the in-built applications’



If any error occurs, you will be able to access the logs at the published log URI in the screenshot.
Also I created a key-pair to be able to access the EMR cluster through SSH and also a default security group for master and slave nodes.

Step 2:

Upload the dataset to a s3 bucket so that the pyspark program will be able to access it from the notebook.
Also upload the pyspark program to the same s3 bucket to keep it handy. I also used a emr_bootstrap.sh file to install any python modules such as matplotlib etc on the EMR cluster(however that's not required in my program currently).



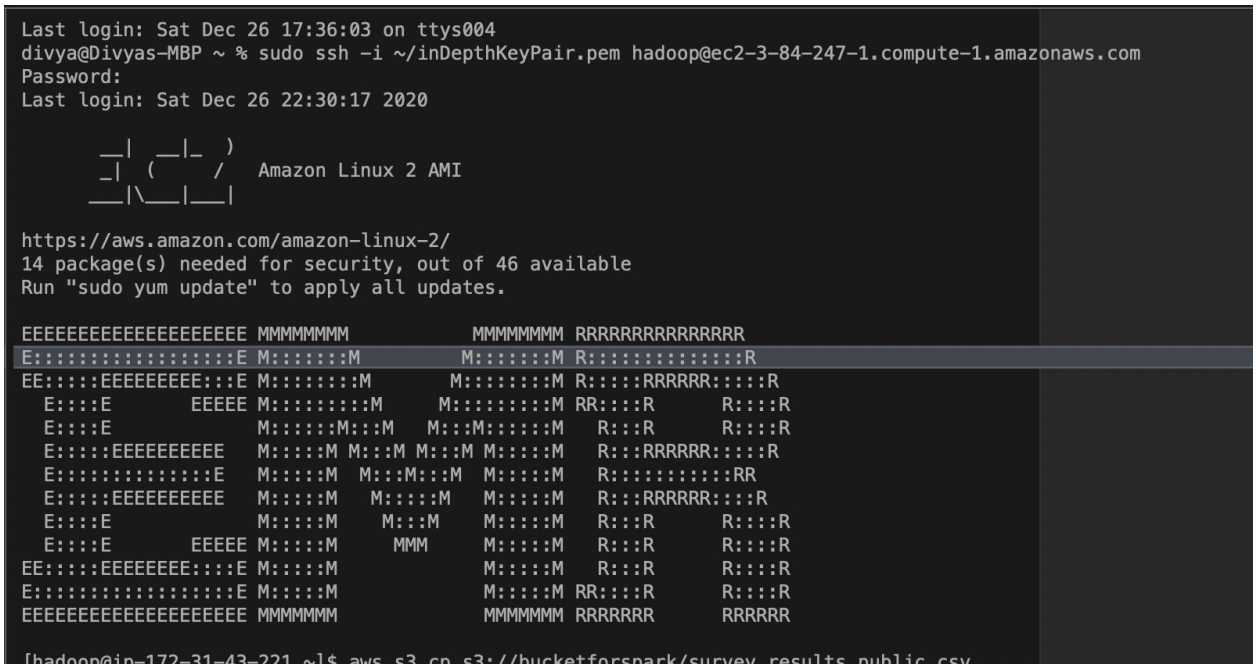
Pyspark program:

```
from pyspark.sql import SparkSession

AGE_MIDPOINT = "age_midpoint"
SALARY_MIDPOINT = "salary_midpoint"
SALARY_MIDPOINT_BUCKET = "salary_midpoint_bucket"

if __name__ == "__main__":

    session = SparkSession.builder.appName("StackOverFlowSurvey").getOrCreate()
    sc=session.sparkContext
    sc.setLogLevel('ERROR')
    dataFrameReader = session.read

    responses = dataFrameReader \
        .option("header", "true") \
        .option("inferSchema", value = True) \
        .csv("s3://bucketforspark/2016-Stack-Overflow-Survey-Responses.csv")

    print("=== Print out schema ===")
    responses.printSchema()

    responseWithSelectedColumns = responses.select("country", "occupation",
        AGE_MIDPOINT, SALARY_MIDPOINT)

    print("=== Print the selected columns of the table ===")
    responseWithSelectedColumns.show()

    print("=== Print records where the response is from Afghanistan ===")
    responseWithSelectedColumns\
        .filter(responseWithSelectedColumns["country"] == "Afghanistan").show()

    print("=== Print the count of occupations ===")
    groupedData = responseWithSelectedColumns.groupBy("occupation")
    groupedData.count().show()

    print("=== Print records with average mid age less than 20 ===")
    responseWithSelectedColumns\
```

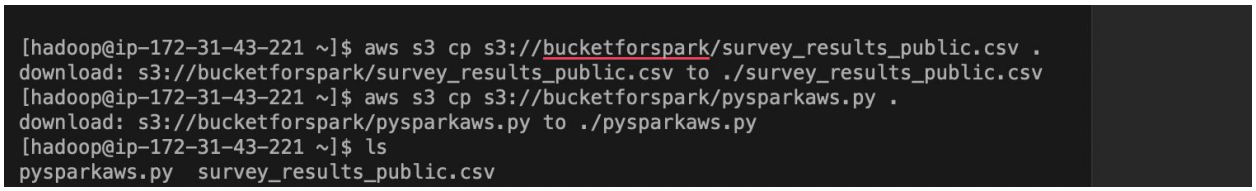
```
print("=== Print records with average mid age less than 20 ===")
responseWithSelectedColumns\
    .filter(responseWithSelectedColumns[AGE_MIDPOINT] < 20).show()

print("=== Print the result by salary middle point in descending order ===")
responseWithSelectedColumns\
    .orderBy(responseWithSelectedColumns[SALARY_MIDPOINT], ascending = False).show()

print("=== Group by country and aggregate by average salary middle point ===")
dataGroupByCountry = responseWithSelectedColumns.groupBy("country")
dataGroupByCountry.avg(SALARY_MIDPOINT).show()

responseWithSalaryBucket = responses.withColumn(SALARY_MIDPOINT_BUCKET,
    ((responses[SALARY_MIDPOINT]/20000).cast("integer")*20000))

print("=== With salary bucket column ===")
responseWithSalaryBucket.select(SALARY_MIDPOINT, SALARY_MIDPOINT_BUCKET).show()

print("=== Group by salary bucket ===")
responseWithSalaryBucket \
    .groupBy(SALARY_MIDPOINT_BUCKET) \
    .count() \
    .orderBy(SALARY_MIDPOINT_BUCKET) \
    .show()

session.stop()
```

Step 3:
Now to connect to EMR cluster from the local machine:

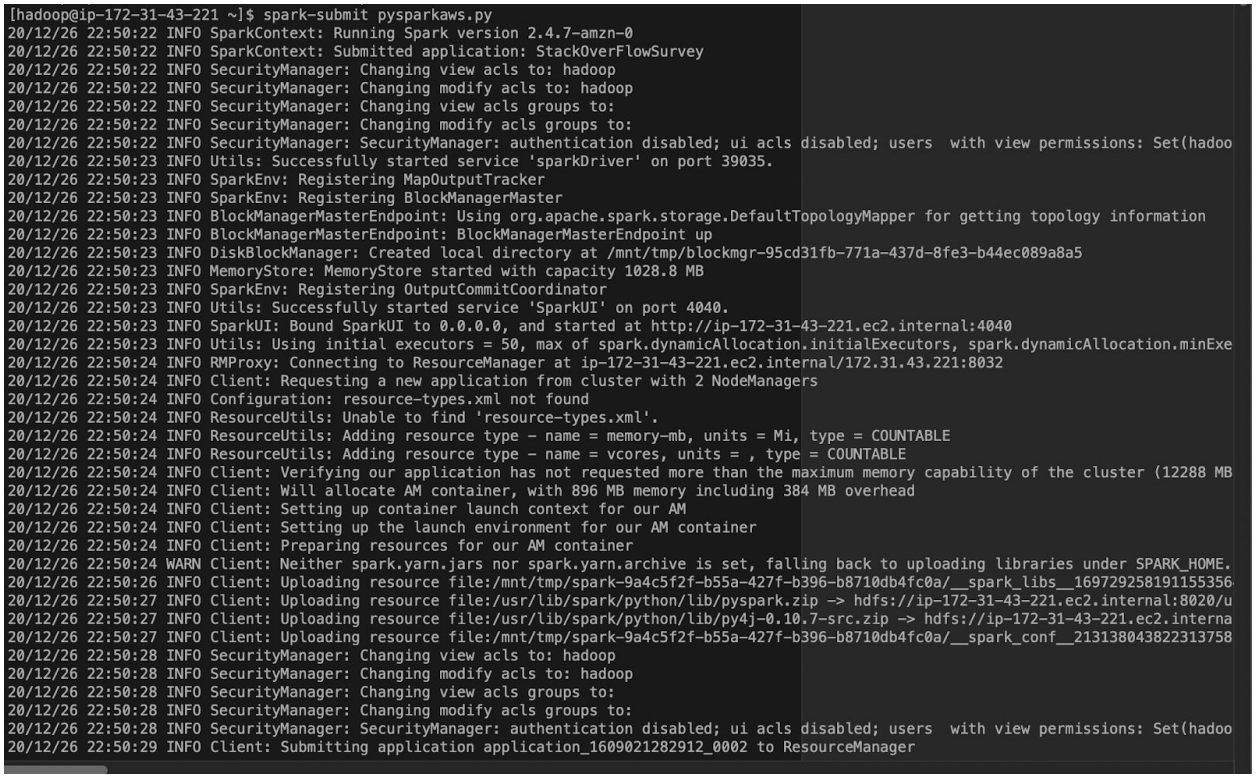
I used ssh using the keypair in the following manner:(essentially a emr cluster is an ec2 instance underneath)



Then I downloaded the pyspark program from s3 to make a spark-submit to the emr-cluster.



Now I made a spark-submit in the following manner:


[hadoop@ip-172-31-43-221 ~]$ spark-submit pysparkaws.py
20/12/26 22:50:22 INFO SparkContext: Running Spark version 2.4.7-amzn-0
20/12/26 22:50:22 INFO SparkContext: Submitted application: StackOverFlowSurvey
20/12/26 22:50:22 INFO SecurityManager: Changing view acls to: hadoop
20/12/26 22:50:22 INFO SecurityManager: Changing modify acls to: hadoop
20/12/26 22:50:22 INFO SecurityManager: Changing view acls groups to:
20/12/26 22:50:22 INFO SecurityManager: Changing modify acls groups to:
20/12/26 22:50:22 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(hadoo
20/12/26 22:50:23 INFO Utils: Successfully started service 'sparkDriver' on port 39035.
20/12/26 22:50:23 INFO SparkEnv: Registering MapOutputTracker
20/12/26 22:50:23 INFO SparkEnv: Registering BlockManagerMaster
20/12/26 22:50:23 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
20/12/26 22:50:23 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
20/12/26 22:50:23 INFO DiskBlockManager: Created local directory at /mnt/tmp/blockmgr-95cd31fb-771a-437d-8fe3-b44ec089a8a5
20/12/26 22:50:23 INFO MemoryStore: MemoryStore started with capacity 1028.8 MB
20/12/26 22:50:23 INFO SparkEnv: Registering OutputCommitCoordinator
20/12/26 22:50:23 INFO Utils: Successfully started service 'SparkUI' on port 4040.
20/12/26 22:50:23 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://ip-172-31-43-221.ec2.internal:4040
20/12/26 22:50:23 INFO Utils: Using initial executors = 50, max of spark.dynamicAllocation.initialExecutors, spark.dynamicAllocation.minExe
20/12/26 22:50:24 INFO RMProxy: Connecting to ResourceManager at ip-172-31-43-221.ec2.internal/172.31.43.221:8032
20/12/26 22:50:24 INFO Client: Requesting a new application from cluster with 2 NodeManagers
20/12/26 22:50:24 INFO Configuration: resource-types.xml not found
20/12/26 22:50:24 INFO ResourceUtils: Unable to find 'resource-types.xml'.
20/12/26 22:50:24 INFO ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
20/12/26 22:50:24 INFO ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
20/12/26 22:50:24 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (12288 MB
20/12/26 22:50:24 INFO Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
20/12/26 22:50:24 INFO Client: Setting up container launch context for our AM
20/12/26 22:50:24 INFO Client: Setting up the launch environment for our AM container
20/12/26 22:50:24 INFO Client: Preparing resources for our AM container
20/12/26 22:50:24 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
20/12/26 22:50:26 INFO Client: Uploading resource file:/mnt/tmp/spark-9a4c5f2f-b55a-427f-b396-b8710db4fc0a/__spark_libs__169729258191155356
20/12/26 22:50:27 INFO Client: Uploading resource file:/usr/lib/spark/python/lib/pyspark.zip -> hdfs://ip-172-31-43-221.ec2.internal:8020/u
20/12/26 22:50:27 INFO Client: Uploading resource file:/usr/lib/spark/python/lib/py4j-0.10.7-src.zip -> hdfs://ip-172-31-43-221.ec2.interna
20/12/26 22:50:27 INFO Client: Uploading resource file:/mnt/tmp/spark-9a4c5f2f-b55a-427f-b396-b8710db4fc0a/__spark_conf__213138043822313758
20/12/26 22:50:28 INFO SecurityManager: Changing view acls to: hadoop
20/12/26 22:50:28 INFO SecurityManager: Changing modify acls to: hadoop
20/12/26 22:50:28 INFO SecurityManager: Changing view acls groups to:
20/12/26 22:50:28 INFO SecurityManager: Changing modify acls groups to:
20/12/26 22:50:28 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(hadoo
20/12/26 22:50:29 INFO Client: Submitting application application_1609021282912_0002 to ResourceManager

Output:

```
=== Print the selected columns of the table ===
+-----------+--------------------+------------+---------------+
|    country|          occupation|age_midpoint|salary_midpoint|
+-----------+--------------------+------------+---------------+
|Afghanistan|                null|        22.0|           null|
|Afghanistan|Mobile developer ...|        32.0|        45000.0|
|Afghanistan|                null|        null|           null|
|Afghanistan|               DevOps|        null|         5000.0|
|Afghanistan|                null|        65.0|           null|
|Afghanistan|                null|        22.0|           null|
|Afghanistan|       Growth hacker|        null|       210000.0|
|Afghanistan|Back-end web deve...|        27.0|         5000.0|
|    Albania|                null|        27.0|           null|
|    Albania|Back-end web deve...|        22.0|         5000.0|
|    Albania|Full-stack web de...|        27.0|         5000.0|
|    Albania|Full-stack web de...|        22.0|        15000.0|
|    Albania|Full-stack web de...|        27.0|         5000.0|
|    Albania|Back-end web deve...|        27.0|         5000.0|
|    Albania|Back-end web deve...|        22.0|        15000.0|
|    Algeria|                null|        44.5|           null|
|    Algeria|   Desktop developer|        27.0|           null|
|    Algeria|             Student|        16.0|           null|
|    Algeria|                null|        22.0|           null|
|    Algeria|   Desktop developer|        27.0|        15000.0|
+-----------+--------------------+------------+---------------+
only showing top 20 rows

=== Print records where the response is from Afghanistan ===
+-----------+--------------------+------------+---------------+
|    country|          occupation|age_midpoint|salary_midpoint|
+-----------+--------------------+------------+---------------+
|Afghanistan|                null|        22.0|           null|
|Afghanistan|Mobile developer ...|        32.0|        45000.0|
|Afghanistan|                null|        null|           null|
|Afghanistan|               DevOps|        null|         5000.0|
|Afghanistan|                null|        65.0|           null|
|Afghanistan|                null|        22.0|           null|
|Afghanistan|       Growth hacker|        null|       210000.0|
|Afghanistan|Back-end web deve...|        27.0|         5000.0|
|Afghanistan|Database administ...|        27.0|         5000.0|
|Afghanistan|                null|        22.0|           null|
|Afghanistan|                null|        null|           null|
|Afghanistan|                null|        16.0|           null|
|Afghanistan|                null|        65.0|           null|
|Afghanistan|                null|        27.0|           null|
|Afghanistan|Full-stack web de...|        37.0|        75000.0|
```

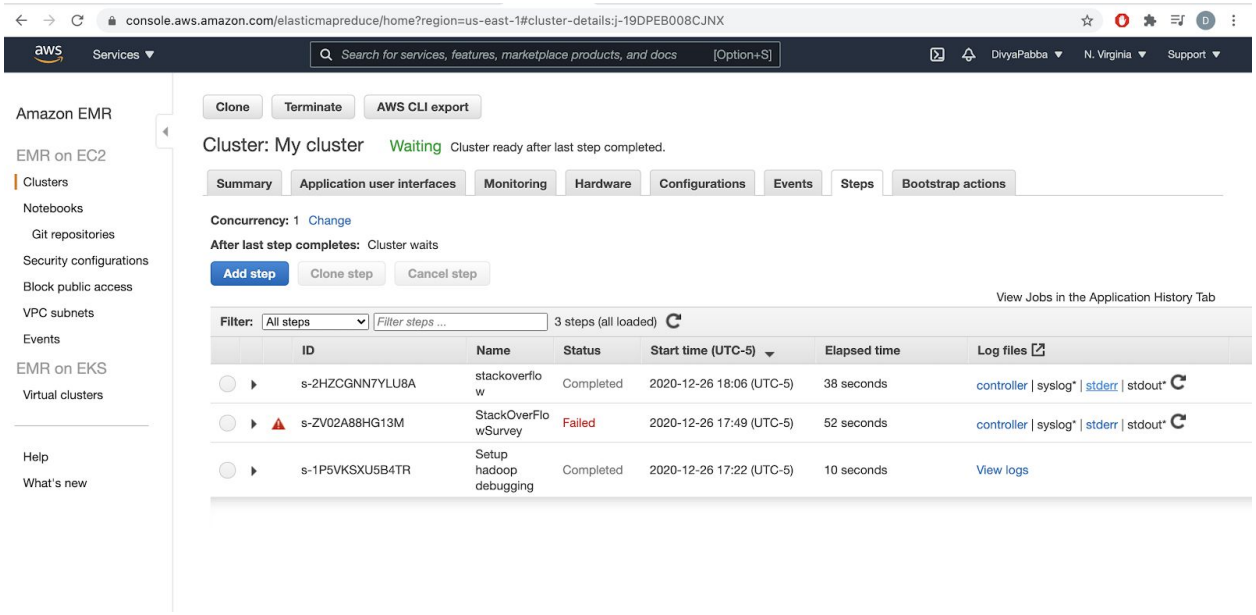
```
=== Print the count of occupations ===
+--------------------+-----+
|          occupation|count|
+--------------------+-----+
|System administrator|  745|
|                null| 6511|
|            Designer|  333|
|       Growth hacker|  120|
|Front-end web dev...| 2873|
|              DevOps| 1074|
|Business intellig...|  392|
|Developer with a ...|  960|
|      Data scientist|  800|
|    Mobile developer| 1373|
| Graphics programmer|  293|
|             Analyst|  570|
|Enterprise level ...| 1471|
|Executive (VP of ...|  888|
|     Product manager|  333|
|Mobile developer ...|   59|
|             Student| 5619|
| Engineering manager|  697|
|               other| 2585|
|   Desktop developer| 3390|
+--------------------+-----+
only showing top 20 rows

=== Print records with average mid age less than 20 ===
+---------+--------------------+------------+--------------+
|  country|          occupation|age_midpoint|salary_midpoint|
+---------+--------------------+------------+--------------+
|  Algeria|             Student|        16.0|          null|
|  Algeria|Back-end web deve...|        16.0|          null|
|Argentina|             Student|        16.0|        5000.0|
|Argentina|Back-end web deve...|        16.0|        5000.0|
|  Armenia|Back-end web deve...|        16.0|        5000.0|
|  Armenia|                null|        16.0|          null|
|  Armenia|Mobile developer ...|        16.0|        5000.0|
|  Armenia|Mobile developer ...|        16.0|        5000.0|
|  Austria|Mobile developer ...|        16.0|          null|
|  Austria|Full-stack web de...|        16.0|          null|
|  Austria|Full-stack web de...|        16.0|       15000.0|
|  Austria|                null|        16.0|          null|
|  Austria|             Student|        16.0|          null|
|  Austria|                null|        16.0|          null|
|  Austria|             Student|        16.0|          null|
|  Austria|             Student|        16.0|          null|
```

```
=== Print the result by salary middle point in descending order ===
+------------------+--------------------+------------+--------------+
|           country|          occupation|age_midpoint|salary_midpoint|
+------------------+--------------------+------------+--------------+
|     United States|               other|        44.5|      210000.0|
|     United States|               other|        54.5|      210000.0|
|     United States|Executive (VP of ...|        22.0|      210000.0|
|              null|               other|        44.5|      210000.0|
|     United States| Graphics programmer|        27.0|      210000.0|
|              null|Developer with a ...|        null|      210000.0|
|     United States|Back-end web deve...|        32.0|      210000.0|
|           Belgium|Executive (VP of ...|        44.5|      210000.0|
|     United States|Executive (VP of ...|        44.5|      210000.0|
|           Belgium|Executive (VP of ...|        32.0|      210000.0|
|     United States|Full-stack web de...|        32.0|      210000.0|
|           Denmark|Full-stack web de...|        16.0|      210000.0|
|     United States|    Mobile developer|        44.5|      210000.0|
|           Iceland| Graphics programmer|        44.5|      210000.0|
|     United States|Executive (VP of ...|        44.5|      210000.0|
|Ireland {Republic}|Back-end web deve...|        54.5|      210000.0|
|     United States| Engineering manager|        37.0|      210000.0|
|Ireland {Republic}|              DevOps|        16.0|      210000.0|
|     United States|             Analyst|        32.0|      210000.0|
|          Slovakia|Executive (VP of ...|        27.0|      210000.0|
+------------------+--------------------+------------+--------------+
only showing top 20 rows

=== Group by country and aggregate by average salary middle point ===
+----------------+--------------------+
|         country|avg(salary_midpoint)|
+----------------+--------------------+
|          Guyana|   73333.33333333333|
|          Turkey|   23993.90243902439|
|Myanmar, {Burma}|   8333.333333333334|
|            null|  51666.666666666664|
|       Argentina|  27062.937062937064|
|          Angola|             22500.0|
|      East Timor|             65000.0|
|         Albania|             22000.0|
|            Peru|  21923.076923076922|
|           China|   36621.62162162162|
|         Croatia|  21732.673267326732|
|         Andorra|             62500.0|
```

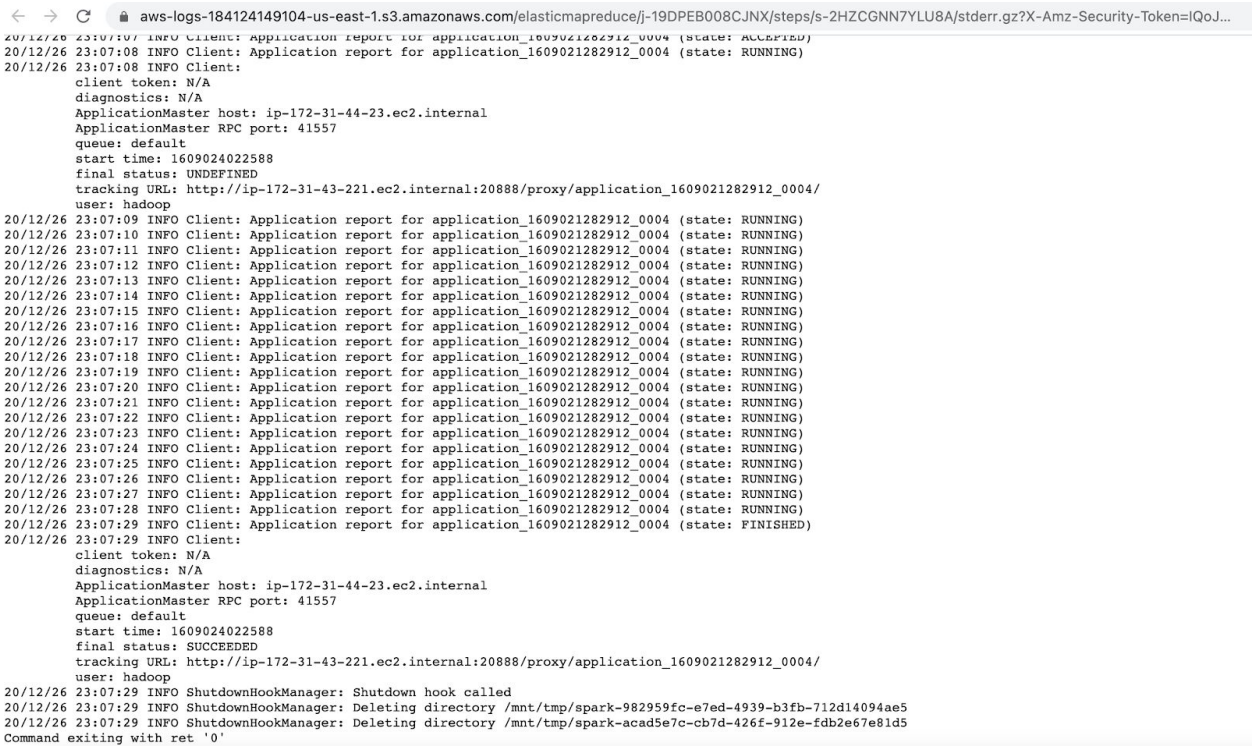
Goal:
To submit a spark-job to process the dataset for the StackOverFlowSurvey results dataset to the AWS EMR cluster from AWS console (through the step option):

Repeat steps 1 and 2 in the same manner(cluster creation and upload files to s3).

Now in the step3, we can simply navigate to the step option on the aws console and submit a new spark application by providing the s3 link of the pyspark program.


console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-details:j-19DPEB008CJNX
aws
Services
Search for services, features, marketplace products, and docs [Option+S]
DivyaPabba
N. Virginia
Support
Amazon EMR
EMR on EC2
Clusters
Notebooks
Git repositories
Security configurations
Block public access
VPC subnets
Events
EMR on EKS
Virtual clusters
Help
What's new
Clone
Terminate
AWS CLI export
Cluster: My cluster
Waiting
Cluster ready after last step completed.
Summary
Application user interfaces
Monitoring
Hardware
Configurations
Events
Steps
Bootstrap actions
Concurrency: 1 Change
After last step completes: Cluster waits
Add step
Clone step
Cancel step
View Jobs in the Application History Tab
Filter: All steps
Filter steps ...
3 steps (all loaded)
ID
Name
Status
Start time (UTC-5)
Elapsed time
Log files
s-2HZCGNN7YLU8A
stackoverflow
Completed
2020-12-26 18:06 (UTC-5)
38 seconds
controller | syslog* | stderr | stdout*
s-ZV02A88HG13M
StackOverFlowSurvey
Failed
2020-12-26 17:49 (UTC-5)
52 seconds
controller | syslog* | stderr | stdout*
s-1P5VKSXU5B4TR
Setup hadoop debugging
Completed
2020-12-26 17:22 (UTC-5)
10 seconds
View logs

Output:


aws-logs-184124149104-us-east-1.s3.amazonaws.com/elasticmapreduce/j-19DPEB008CJNX/steps/s-2HZCGNN7YLU8A/stderr.gz?X-Amz-Security-Token=IQoJ...
20/12/26 23:07:08 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:08 INFO Client:
client token: N/A
diagnostics: N/A
ApplicationMaster host: ip-172-31-44-23.ec2.internal
ApplicationMaster RPC port: 41557
queue: default
start time: 1609024022588
final status: UNDEFINED
tracking URL: http://ip-172-31-43-221.ec2.internal:20888/proxy/application_1609021282912_0004/
user: hadoop
20/12/26 23:07:09 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:10 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:11 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:12 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:13 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:14 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:15 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:16 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:17 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:18 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:19 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:20 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:21 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:22 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:23 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:24 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:25 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:26 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:27 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:28 INFO Client: Application report for application_1609021282912_0004 (state: RUNNING)
20/12/26 23:07:29 INFO Client: Application report for application_1609021282912_0004 (state: FINISHED)
20/12/26 23:07:29 INFO Client:
client token: N/A
diagnostics: N/A
ApplicationMaster host: ip-172-31-44-23.ec2.internal
ApplicationMaster RPC port: 41557
queue: default
start time: 1609024022588
final status: SUCCEEDED
tracking URL: http://ip-172-31-43-221.ec2.internal:20888/proxy/application_1609021282912_0004/
user: hadoop
20/12/26 23:07:29 INFO ShutdownHookManager: Shutdown hook called
20/12/26 23:07:29 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-982959fc-e7ed-4939-b3fb-712d14094ae5
20/12/26 23:07:29 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-acad5e7c-cb7d-426f-912e-fdb2e67e81d5
Command exiting with ret '0'

Reference:
https://medium.com/@roshinijohri/spark-with-jupyter-notebook-on-macos-2-0-0-and-higher-c61b971b5007
https://github.com/jleetutorial/python-spark-tutorial/blob/master/sparkSql/StackOverFlowSurvey.py

Task 3: Final Data Pipeline: Credit Card fraud detection and billing

Goal: You need to build the data pipeline to process credit card transactions in real time to classify if fraudulent or authentic (based on the user's buying pattern) and approve or deny the transaction. For the approved transaction, you batch the transaction details and create a billing statement at monthly time scale.

Step 1: You need two types of events - transaction approval request and actual transaction with details once approved. You need to create two topics - Topic 1: one for approval request and Topic 2: one for transaction details. The Topic 1 should be ingested by kafka --> spark streams --> Listener 1. Listener 1 should trigger a Lambda function to determine if it is fraud or not based on user buying profile. You could have a very simple function that just checks the user profile (buying place, amount of purchase) and if any anomaly, it sends a reject message to another topic in kafka. If approved, then approved message to the same topic.

To accomplish step 1, I created two Topics.

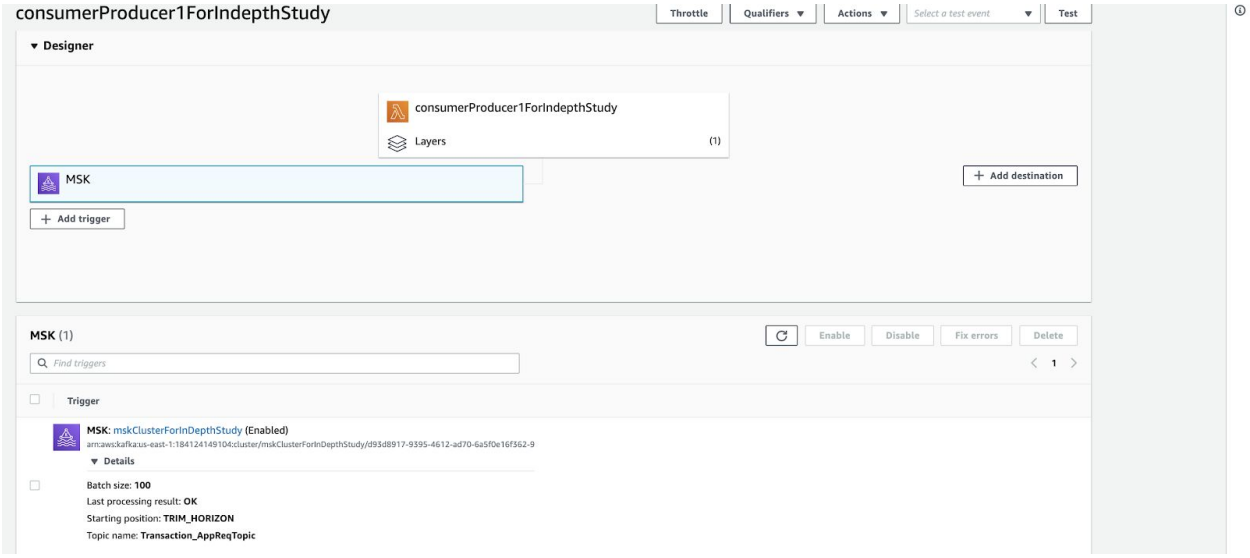
Topic 1:

Transaction_AppReqTopic

Topic 2:

Transaction_DetailsTopic

Once a transaction is triggered to the first topic(Transaction_AppReqTopic), a lambda function(consumerProducer1ForIndepthStudy in zip-folder) which is continuously listening to the first topic through a MSK cluster trigger, processes the event to determine if the transaction is fraudulent or not.



It makes a decision by obtaining the user-profile data from dynamodb table “UserProfiles”.If the transaction-amount and placeoftransaction are valid/allowed values based on the “UserProfiles” table for a given user, then the transaction’s status is updated to “approved” else “rejected” and then sent to the “Transactions_DetailsTopic” by instantiating a kafka producer object.

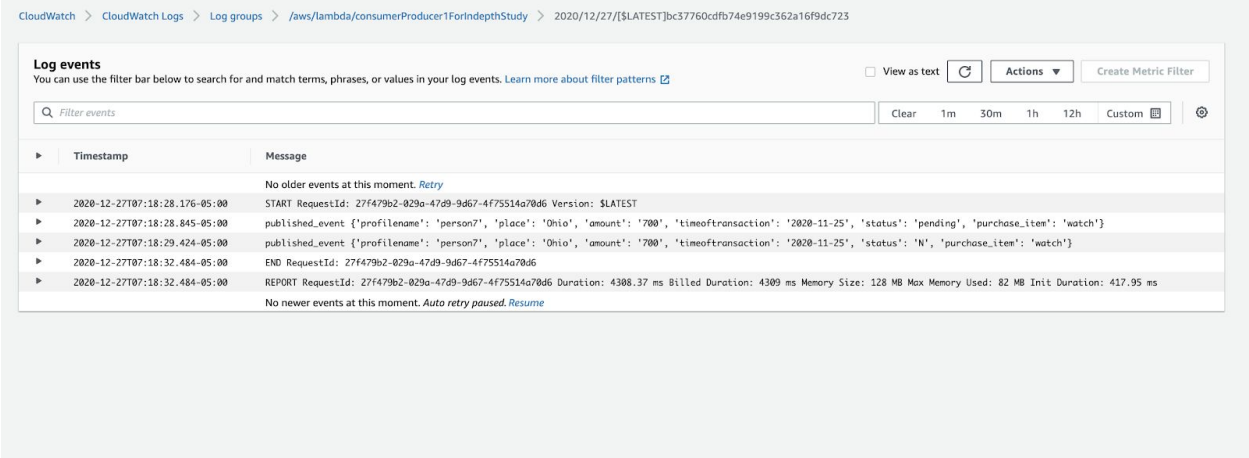
Code is present in the ‘consumerProducer1ForIndepthStudy’ lambda function in the zip folder.

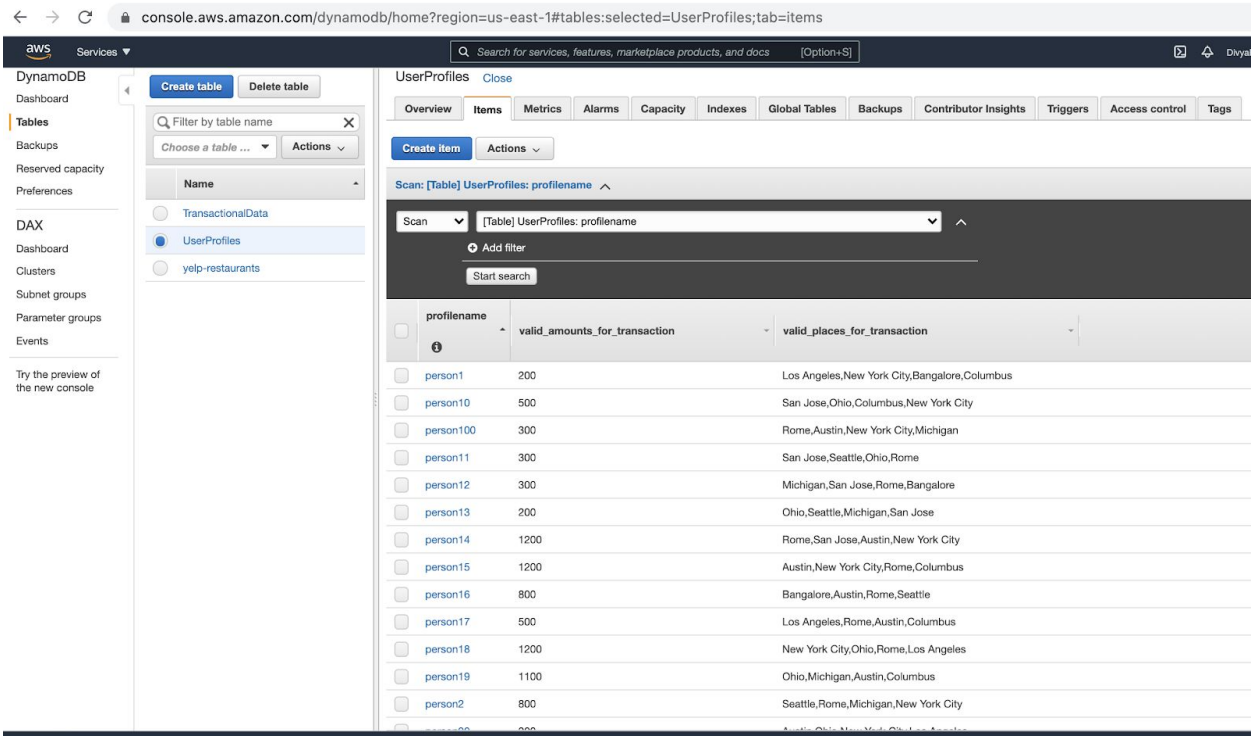
Example of a published event to the Topic-Transaction_AppReqTopic consumed by lambda through MSK trigger:[I’m assuming input transaction contains entire information about the transaction (jus like sent by a POS machine) such as item,amount,time,place etc)]

{"profilename":"person1","**place**":"Ohio","**amount**":"700","**timeoftransaction**":"2020-11-25","**status**":"pending","purchase_item":"watch"}

Example of a published event to the Topic-Transaction_DetailsTopic by the lambda’s producer object:

{"profilename":"person1","**place**":"Ohio","**amount**":"700","**timeoftransaction**":"2020-11-25","**status**":"rejected","purchase_item":"watch"}





Step 2: Topic 2: for every approved transaction, the Point of sale (where the purchase being made) sends the detailed purchase (item, amount, time etc) to topic 2. This is ingested by spark that does ETL and saves the message in a database like dynamodb or RDS for processing later on to generate monthly billing statements.

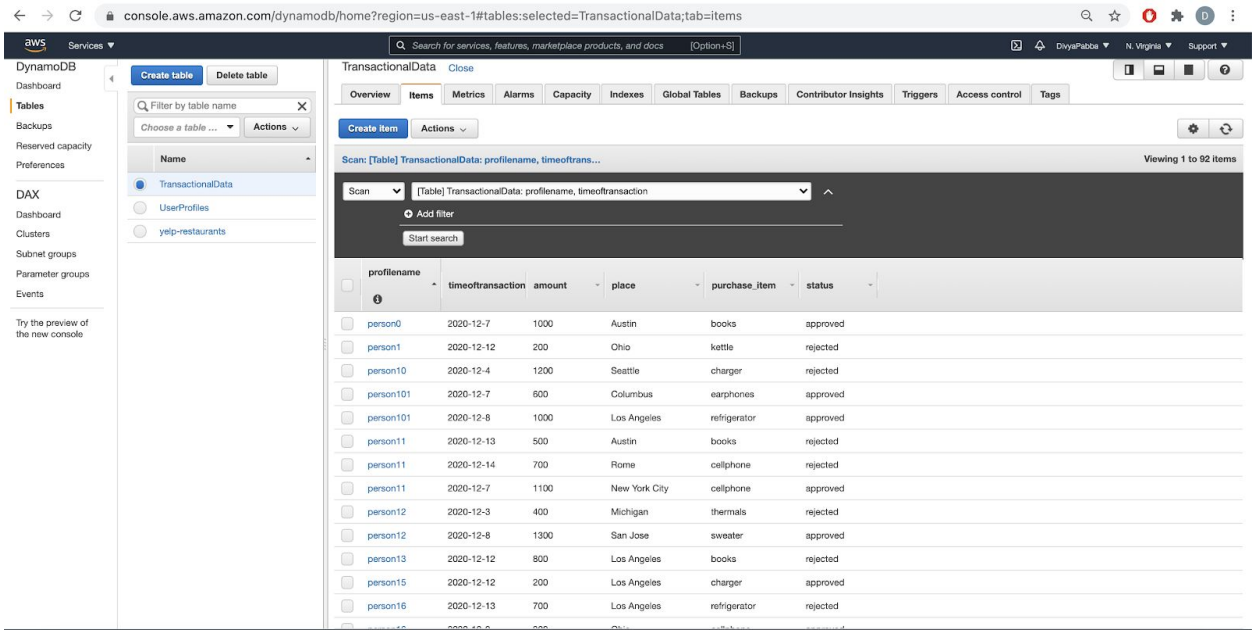
Now as a part of Step1, I have successfully accomplished publishing messages to the second Kafka topic which contains the details of each transaction such as item,amount,time etc along with the approved/reject status.So once the second kafka topic(**Transaction_DetailsTopic**) receives the events, I wrote another lambda function(**ConsumerProducer2ForIndepth**) that listens to this topic through a MSK cluster trigger and processes the events and stores the transactions in dynamodb Table,**'TransactionalData'**.

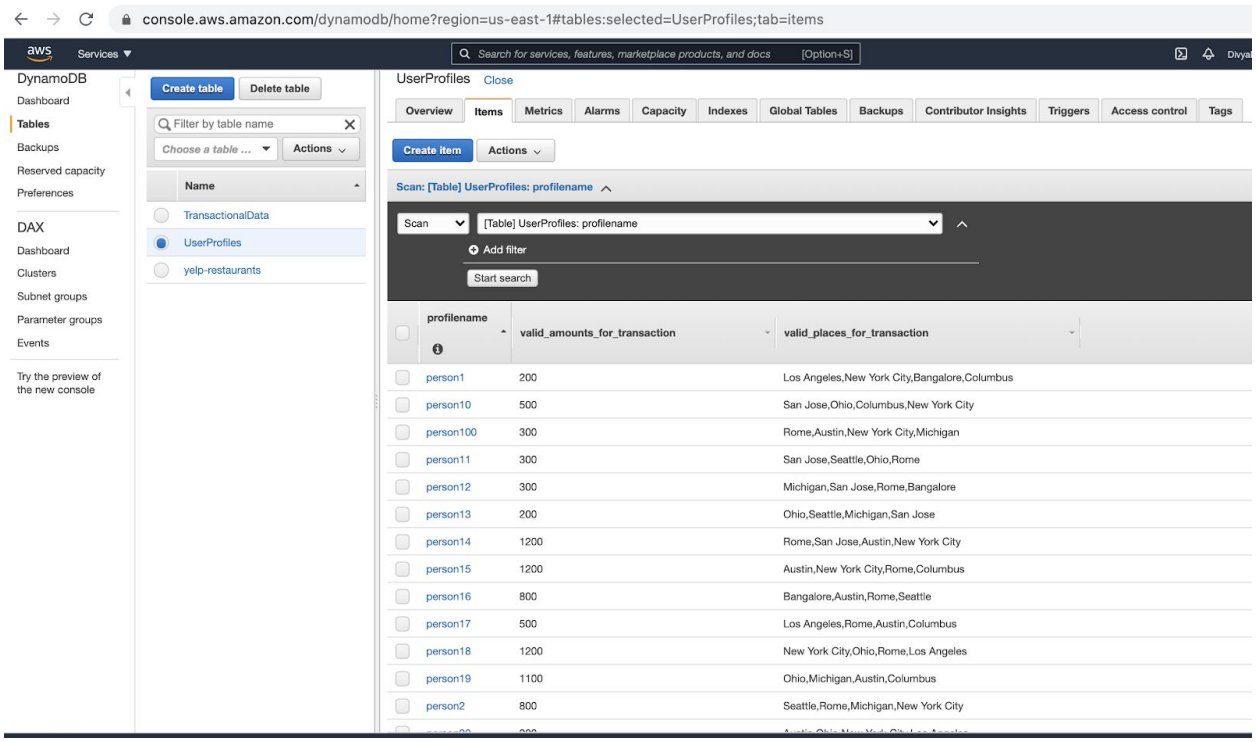
Code is present in the **ConsumerProducer2ForIndepth** lambda function in the zip folder.

Before actually performing the above steps 1 and 2, I set up the dynamodb tables using the code present in **DynamoUtilityForInDepthTask3.**

Step 3: Create a simulator to simulate at least 100 users and create a series of transaction requests to demonstrate this. You could be a bit creative to show the demo - you could build a small dashboard to plot the events/topics to show which one rejected and which one approved and billing statement.

In the **DynamoUtilityForInDepthTask3,** I wrote a method to simulate 100 users and created UserProfiles for each user with some random valid/permitted amounts and places in the 'UserProfiles' Table and also randomly generated around 100 sample transactions.





I performed obtaining a billing statement through a trigger to the lambda function namely, billing through the API gateway by making a POST call.

In detailed:

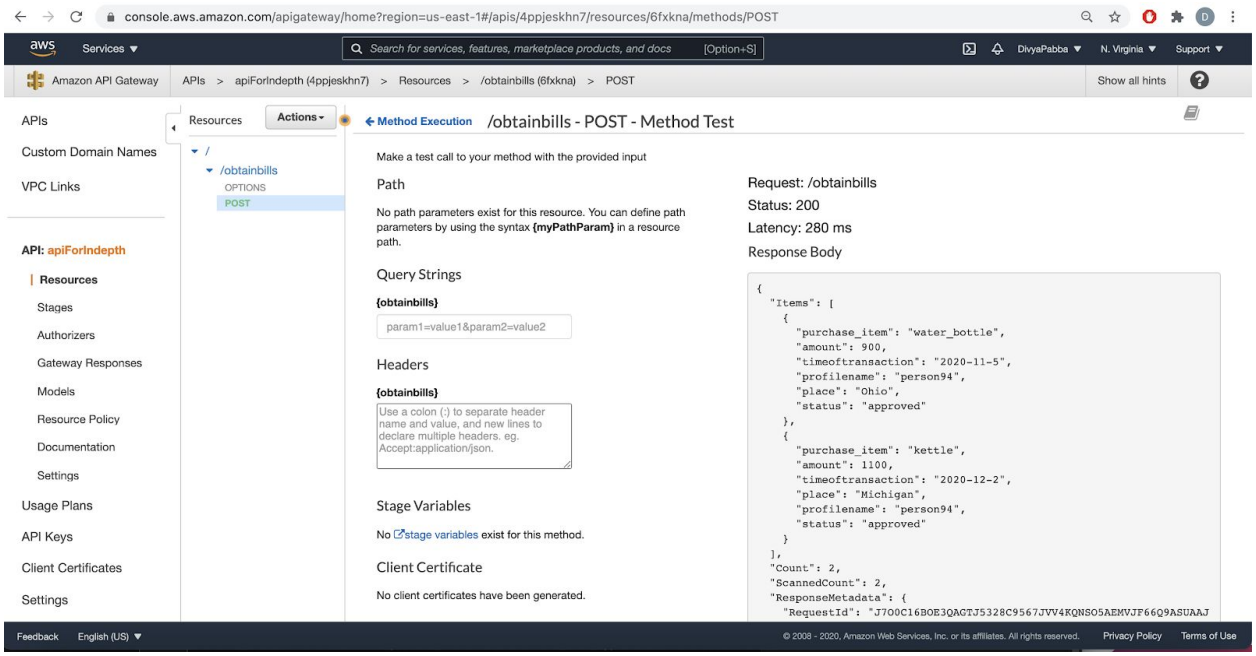
I make a POST call through API gateway with the following parameters:

{

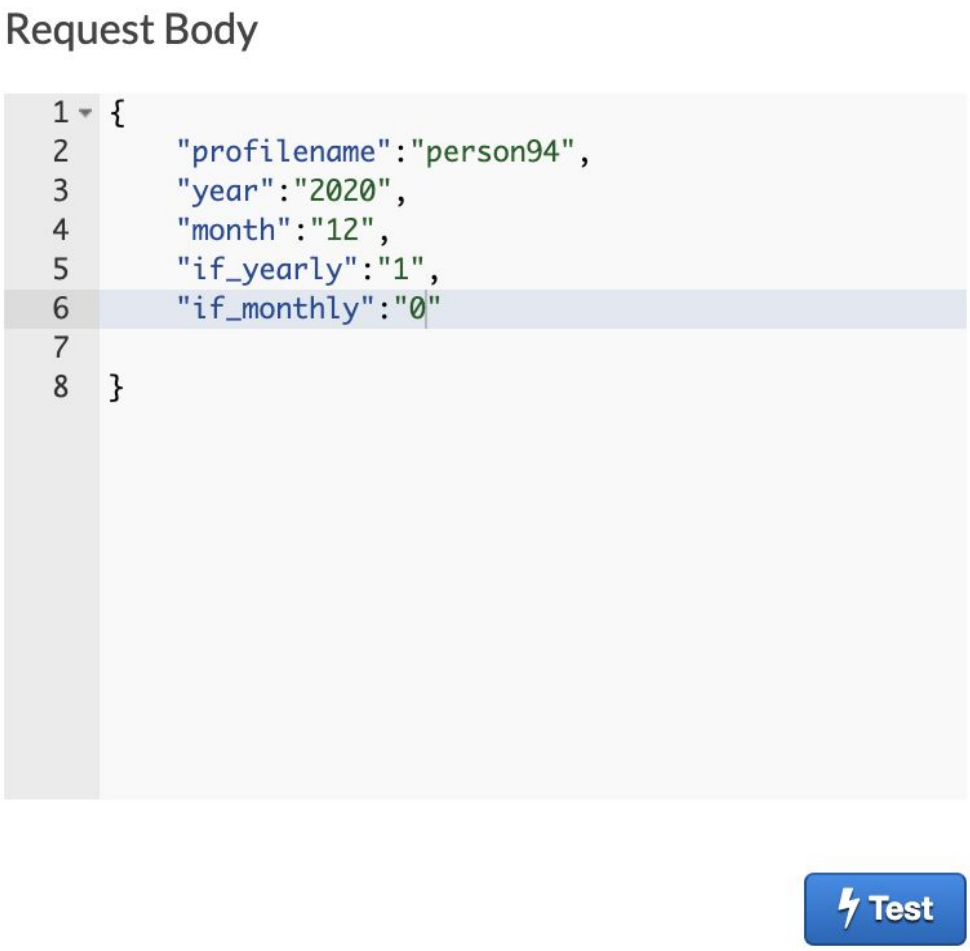
"profilename":"person94", #username
"year":"2020", #year for the billing statement to be generated
"month":"12", #month of the billing statement to be generated
"if_yearly":"1", #if statement to be generated is of yearly type
"if_monthly":"0" #if statement to be generated is of monthly type

}

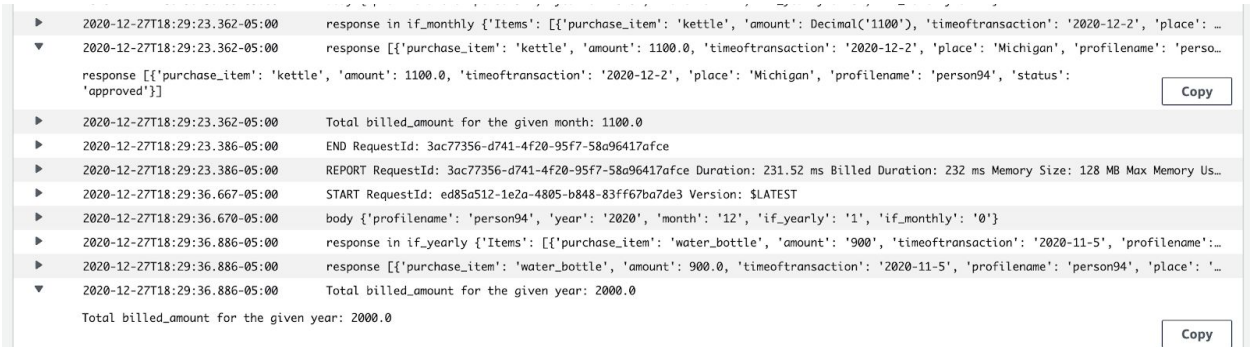
POST method:



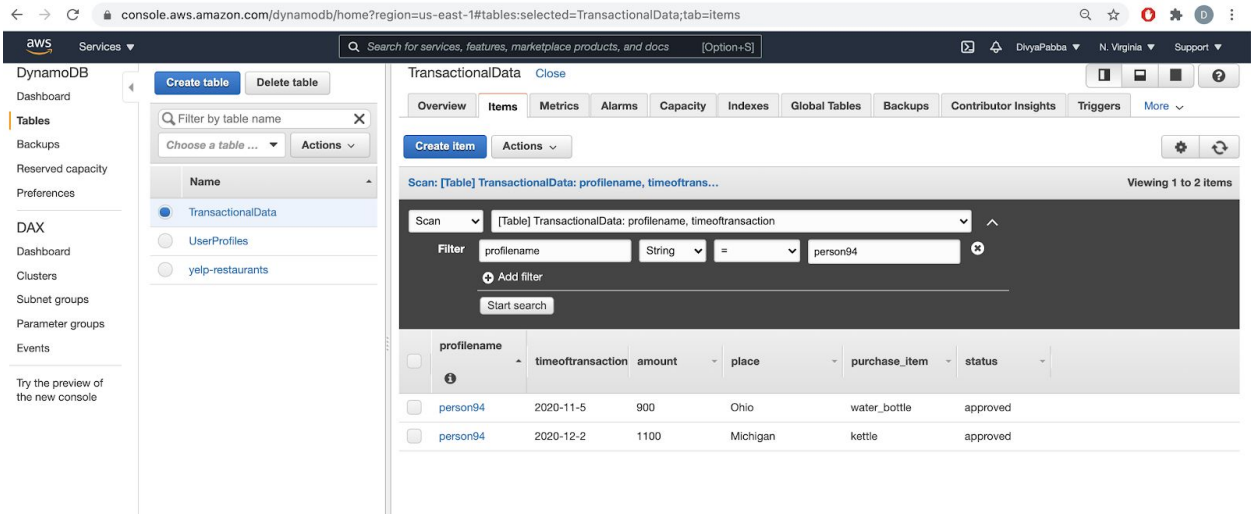
API call Test:



Output from Cloudwatch logs:(This says what are the transactions that are approved and billed.It also generates the total billed amount if it is monthly or yearly)



Validation from DynamoDb data:



In this way we can utilize API Gateway POST call Test, to obtain billing information for any user monthly or yearly.