Lab 4 Report

50.046 CCIoT

SUTD 2022 ISTD

Group 3

Han Xing Yi (1004330)

Ang Song Gee (1004589)

James Raphael Tiovalen (1004555)

Velusamy Sathiakumar Ragul Balaji (1004101)

Table of Contents

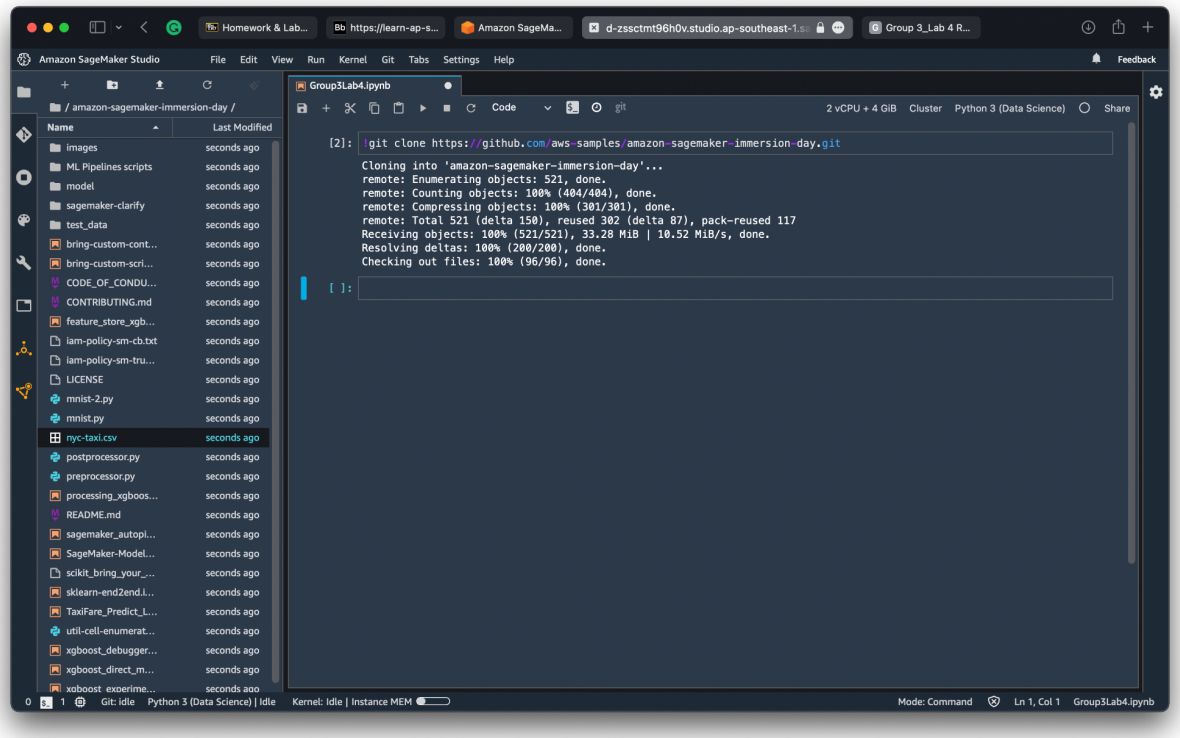
Prerequisite: Setup SageMaker Studio

✅ Create a New IAM Role

✅ Provisioned SageMaker Domain

✅ Created Jupyter Notebook

✅ Downloaded the `nyc-taxi.csv` dataset

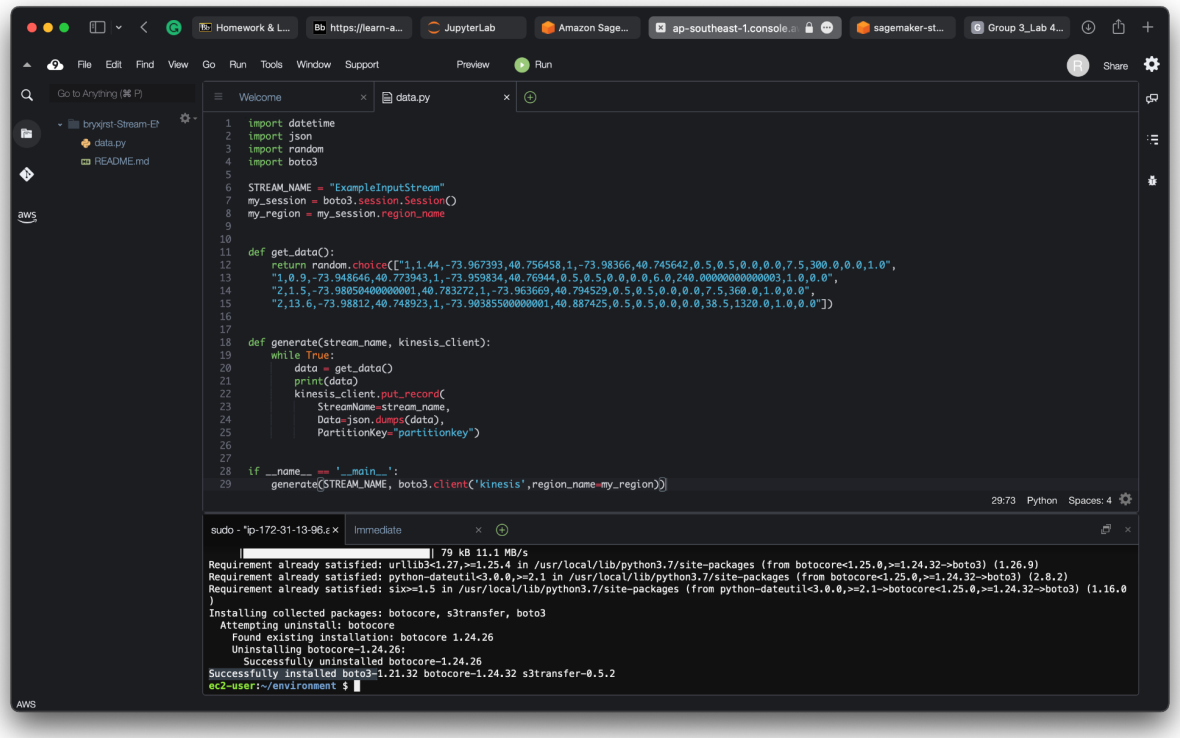


The tasks below are executed in the following order as specified in the [workshop's instructions]:

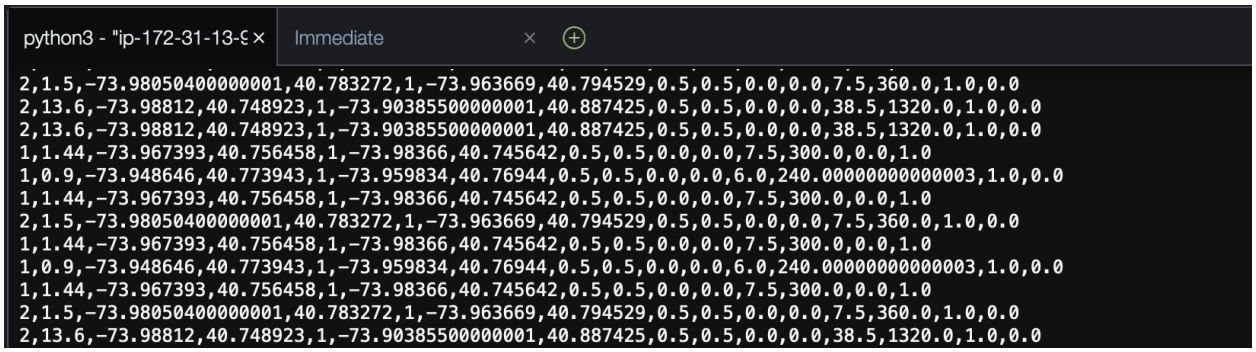
1. Task 3: ML Model Training
2. Task 2: Streaming Data Processing
3. Task 1: Streaming Data Generation
4. Task 4: Storing the Prediction

Task 1: Streaming Data Generation

✅ Create Cloud9 Instance and Configure `data.py` Script

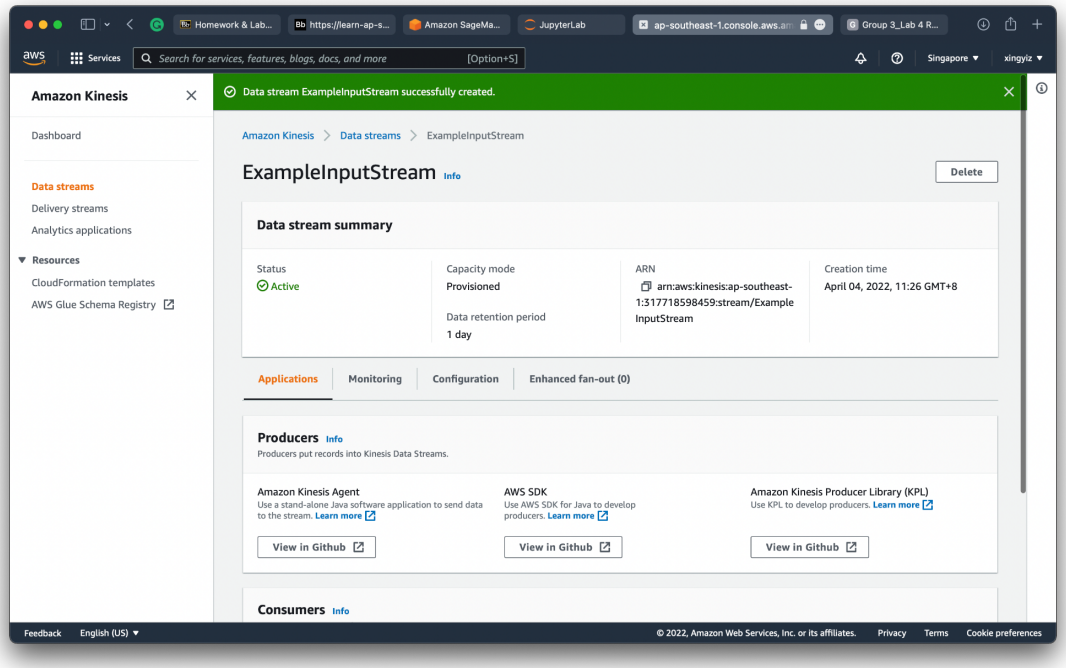


✅ Run `data.py` Script for Streaming Data Injection

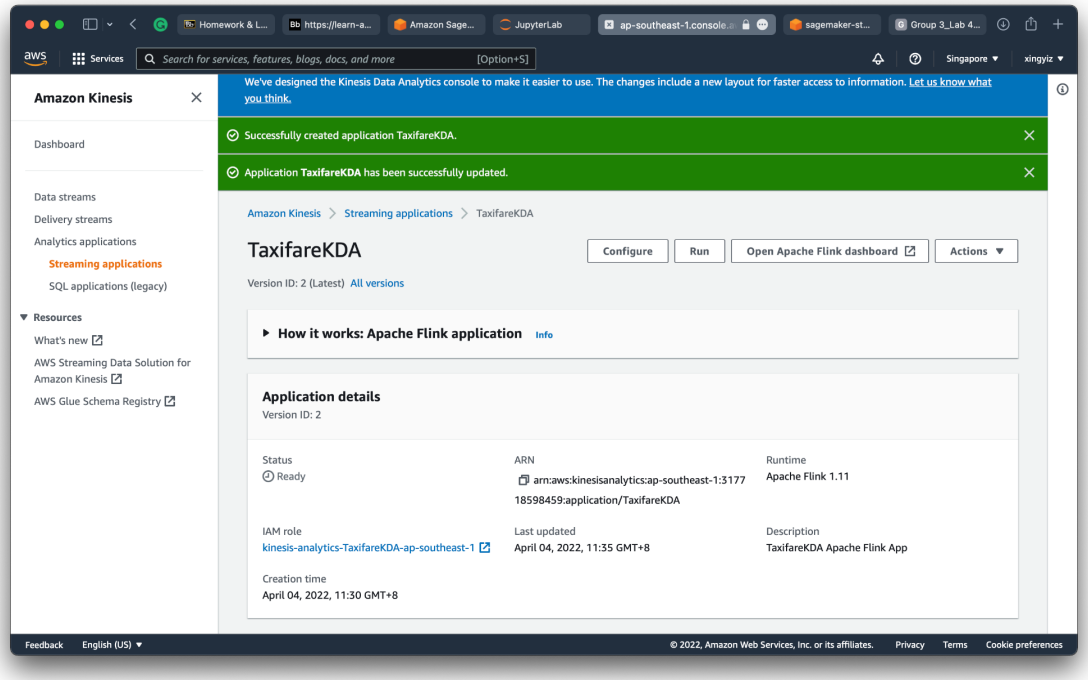


Task 2: Streaming Data Processing

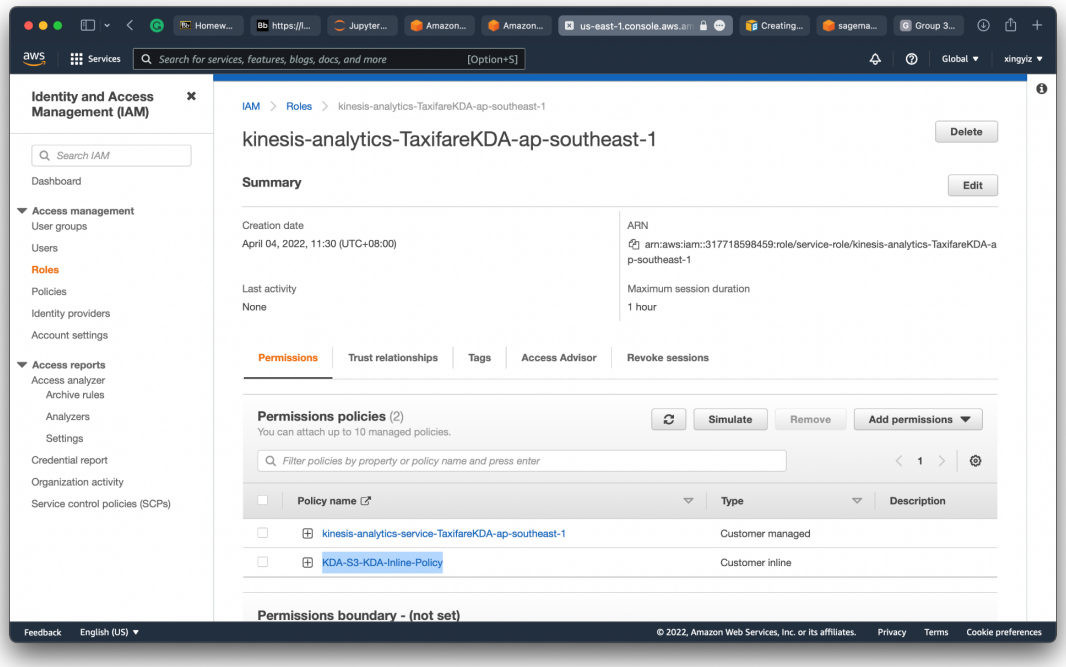
✅ Create Kinesis Data Stream



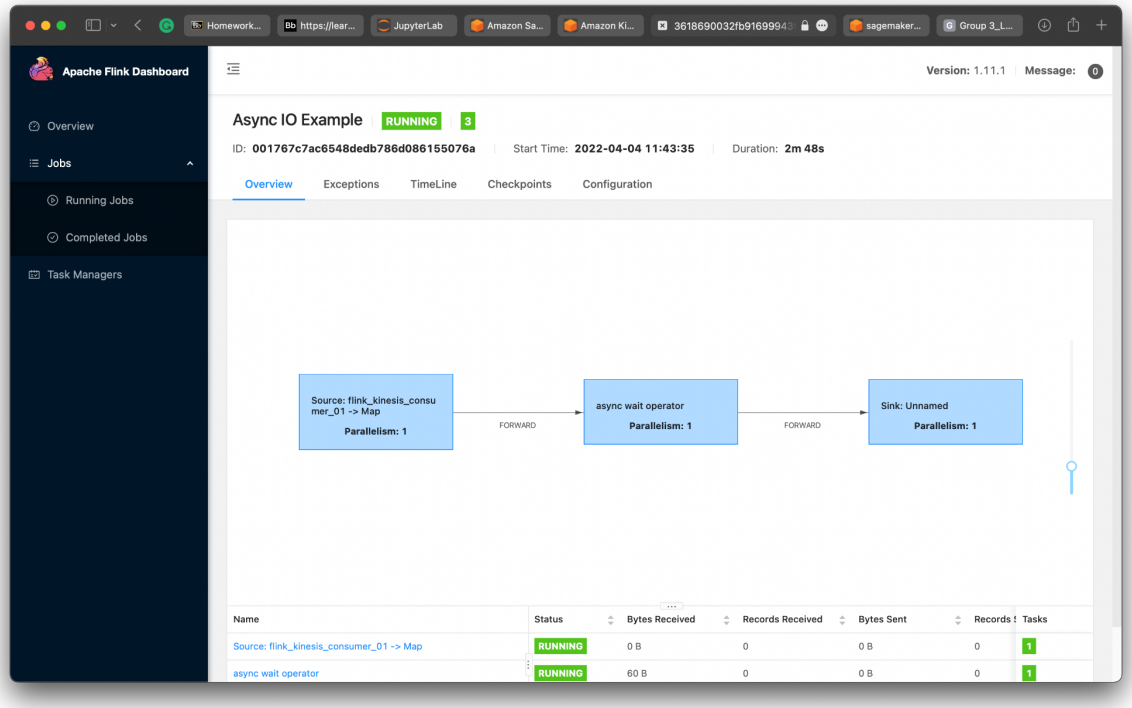
✅ Create and Configure Kinesis Data Analytics App



✅ Add IAM Policies to the KDA Role

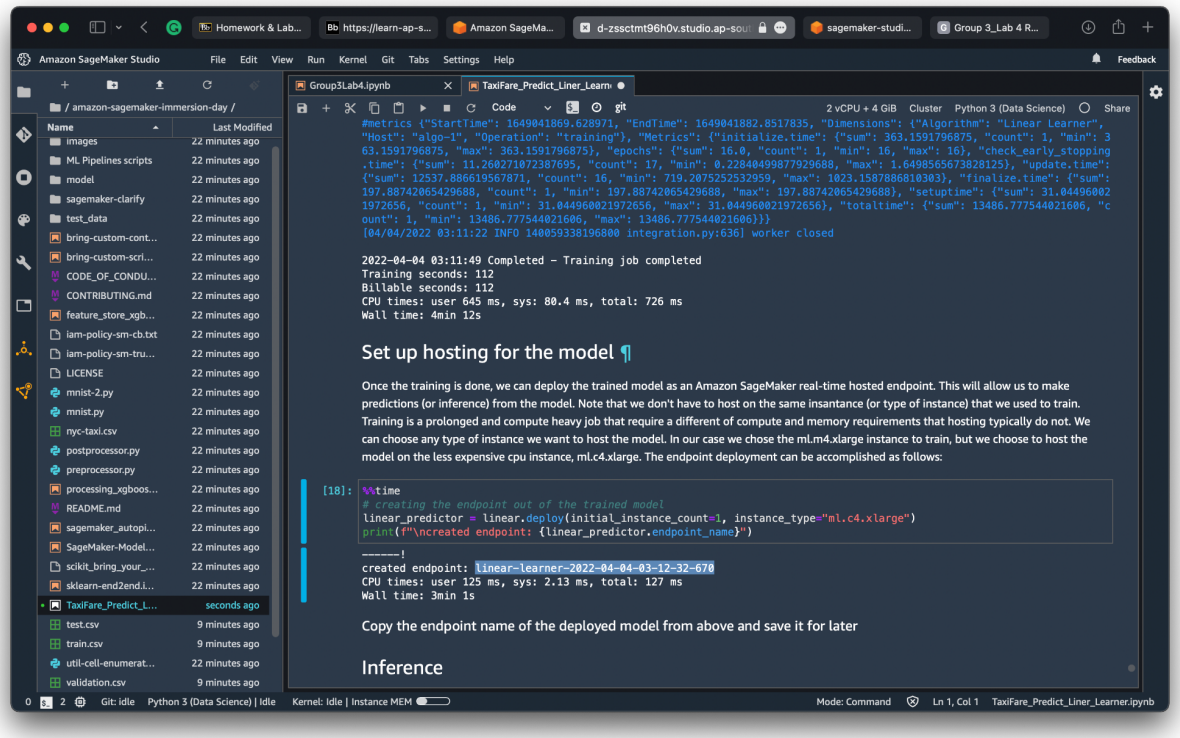


✅ Run the KDA App and View Apache Flink Dashboard



Task 3: ML Model Training

✅ Train and Deploy the ML Model

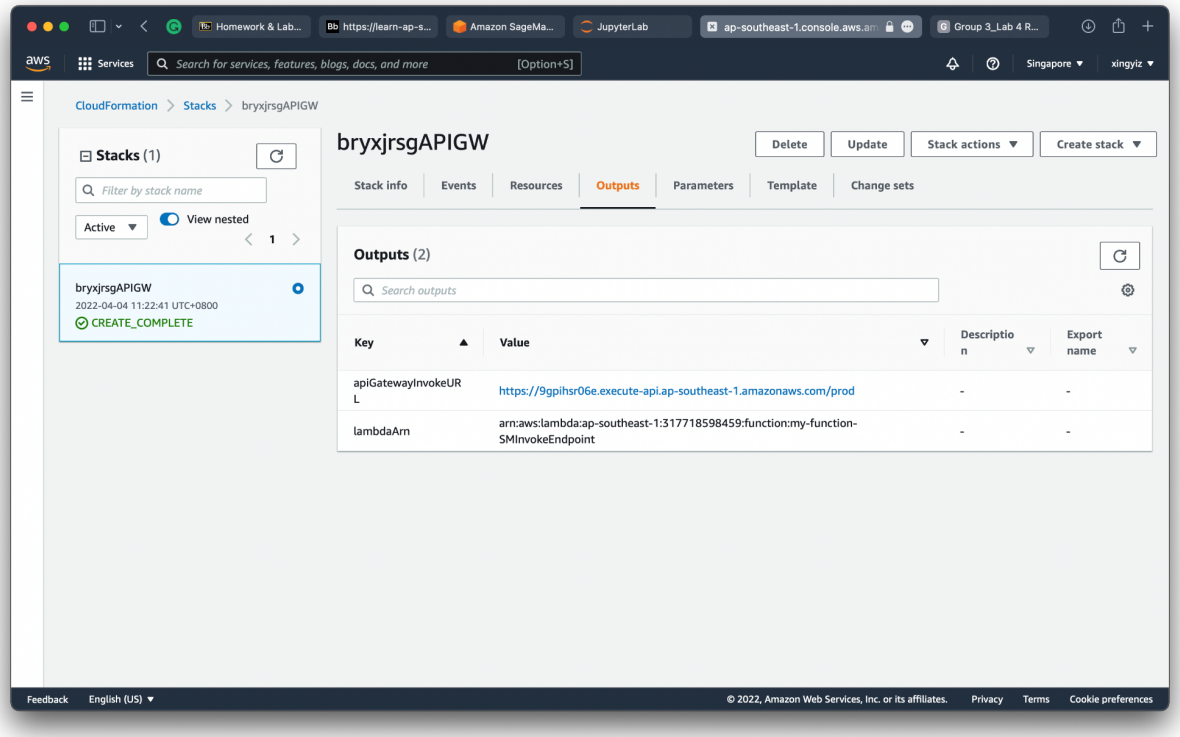


Created endpoint: `linear-learner-2022-04-04-03-12-32-670`

✅ Run a Simple Inference on Our Endpoint

```
payload:  1,1.6,-73.973587,40.754896,1,-73.998192,40.760822,0.5,0.5,0.0,0.0,8.5,420.0,1.0,0.0
 <class 'str'>
Result: {'predictions': [{'score': 7.277685165405273}]}
Actual fare: 7.5
Prediction: 7.28
Accuracy: 97.07
CPU times: user 28.5 ms, sys: 3.38 ms, total: 31.9 ms
Wall time: 142 ms
```

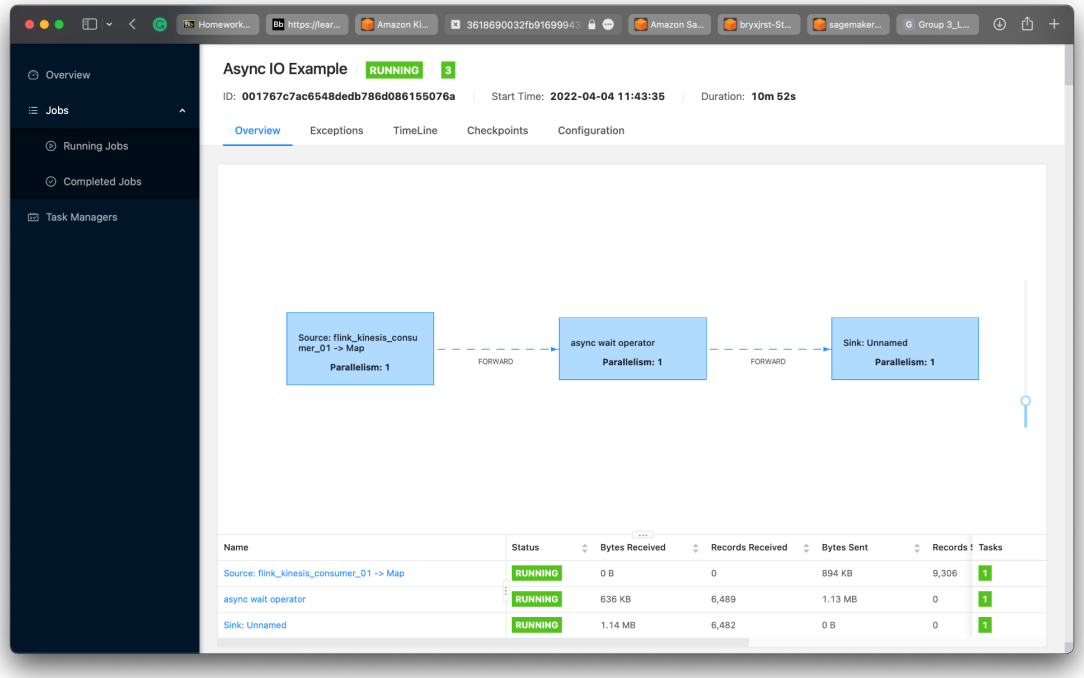
✅ Create Lambda Function and API Gateway



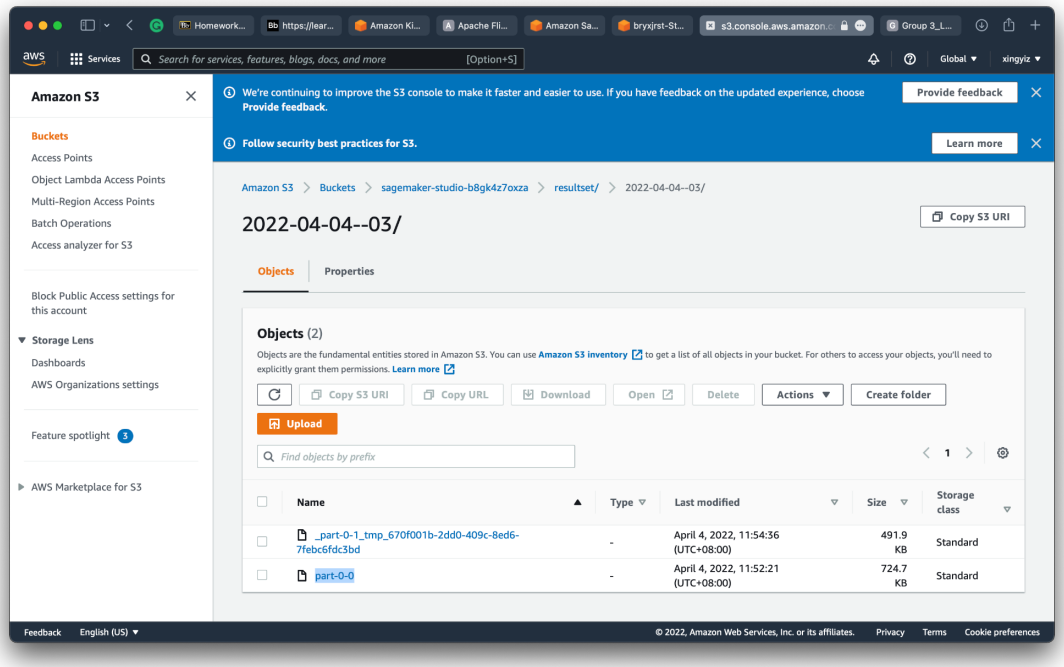
Invoke URL:
https://9gpihsr06e.execute-api.ap-southeast-1.amazonaws.com/prod

Task 4: Storing the Prediction

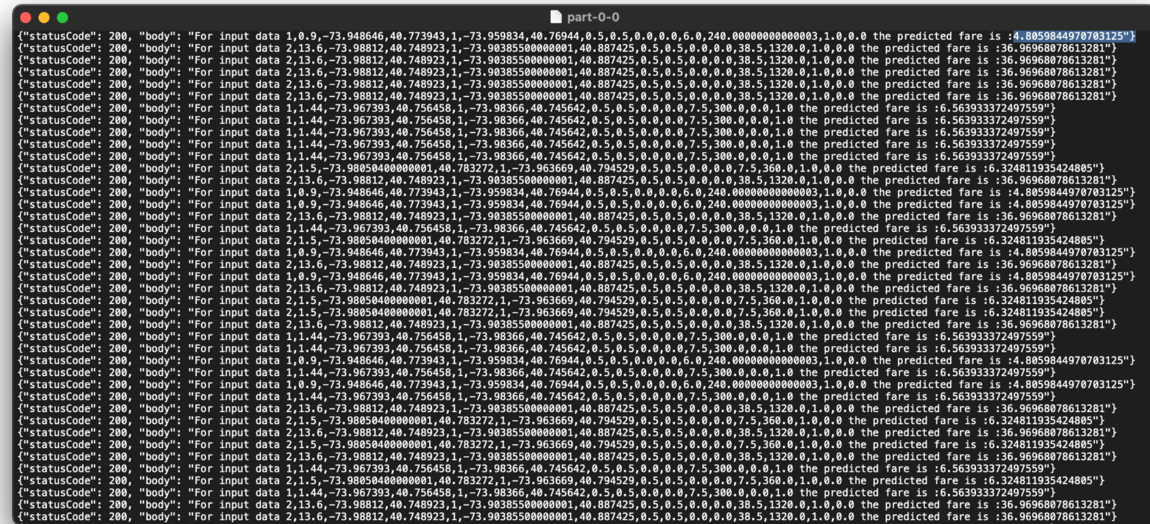
✅ 3 Jobs Running on Apache Flink Dashboard



✅ Prediction Data Generated Stored in S3 Bucket



✅ Download Results File and View Prediction Results



{"statusCode": 200, "body": "For input data 1,0.9,-73.948646,40.773943,1,-73.959834,40.76944,0.5,0.5,0.0,0.0,6.0,240.00000000000003,1.0,0.0 the predicted fare is :4.8059844970703125"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 2,1.5,-73.98050400000001,40.783272,1,-73.963669,40.794529,0.5,0.5,0.0,0.0,7.5,360.0,1.0,0.0 the predicted fare is :6.324811935424805"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 1,0.9,-73.948646,40.773943,1,-73.959834,40.76944,0.5,0.5,0.0,0.0,6.0,240.00000000000003,1.0,0.0 the predicted fare is :4.8059844970703125"}
{"statusCode": 200, "body": "For input data 1,0.9,-73.948646,40.773943,1,-73.959834,40.76944,0.5,0.5,0.0,0.0,6.0,240.00000000000003,1.0,0.0 the predicted fare is :4.8059844970703125"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 2,1.5,-73.98050400000001,40.783272,1,-73.963669,40.794529,0.5,0.5,0.0,0.0,7.5,360.0,1.0,0.0 the predicted fare is :6.324811935424805"}
{"statusCode": 200, "body": "For input data 1,0.9,-73.948646,40.773943,1,-73.959834,40.76944,0.5,0.5,0.0,0.0,6.0,240.00000000000003,1.0,0.0 the predicted fare is :4.8059844970703125"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 1,0.9,-73.948646,40.773943,1,-73.959834,40.76944,0.5,0.5,0.0,0.0,6.0,240.00000000000003,1.0,0.0 the predicted fare is :4.8059844970703125"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,1.5,-73.98050400000001,40.783272,1,-73.963669,40.794529,0.5,0.5,0.0,0.0,7.5,360.0,1.0,0.0 the predicted fare is :6.324811935424805"}
{"statusCode": 200, "body": "For input data 2,1.5,-73.98050400000001,40.783272,1,-73.963669,40.794529,0.5,0.5,0.0,0.0,7.5,360.0,1.0,0.0 the predicted fare is :6.324811935424805"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 1,0.9,-73.948646,40.773943,1,-73.959834,40.76944,0.5,0.5,0.0,0.0,6.0,240.00000000000003,1.0,0.0 the predicted fare is :4.8059844970703125"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 1,0.9,-73.948646,40.773943,1,-73.959834,40.76944,0.5,0.5,0.0,0.0,6.0,240.00000000000003,1.0,0.0 the predicted fare is :4.8059844970703125"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,1.5,-73.98050400000001,40.783272,1,-73.963669,40.794529,0.5,0.5,0.0,0.0,7.5,360.0,1.0,0.0 the predicted fare is :6.324811935424805"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,1.5,-73.98050400000001,40.783272,1,-73.963669,40.794529,0.5,0.5,0.0,0.0,7.5,360.0,1.0,0.0 the predicted fare is :6.324811935424805"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 2,1.5,-73.98050400000001,40.783272,1,-73.963669,40.794529,0.5,0.5,0.0,0.0,7.5,360.0,1.0,0.0 the predicted fare is :6.324811935424805"}
{"statusCode": 200, "body": "For input data 1,1.44,-73.967393,40.756458,1,-73.98366,40.745642,0.5,0.5,0.0,0.0,7.5,300.0,0.0,1.0 the predicted fare is :6.563933372497559"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}
{"statusCode": 200, "body": "For input data 2,13.6,-73.98812,40.748923,1,-73.90385500000001,40.887425,0.5,0.5,0.0,0.0,38.5,1320.0,1.0,0.0 the predicted fare is :36.96968078613281"}

Answers to Reflection Questions

1. What's the job of AWS Kinesis Data Stream in this pipeline? Why can't we link the Python data generator script directly to the AWS Kinesis Data Analytics (KDA) engine?

   The role of AWS Kinesis Data Stream (KDS) is to collect and process all incoming streams of data arriving from the python data generator script `data.py` running in the Cloud9 IDE. The python script does this by placing records into the created KDS stream.

   AWS KDS is required in this pipeline because the AWS KDA engine does not collect input data. It reads data processed by KDS, before transforming and analysing the data in real-time using Apache Flink.

   In other words, the KDA engine relies on the KDS as a data source to continuously intake and process data records from external input sources. Then, the processed data records in KDS can be retrieved and consumed by the KDA engine for data analytics purposes.

2. What's the relation between AWS Kinesis Data Analytics (KDA) and Apache Flink? What are the roles played by the AWS KDA and SageMaker respectively in this pipeline?

   Apache Flink is a framework to build streaming applications. The relationship between KDA and Apache Flink is that KDA is a fully managed and serverless AWS service for Apache Flink.

   KDA provides the underlying infrastructure and handles core capabilities such as provisioning compute resources, parallel computation, automatic scaling, and application backups (from the docs: What Is Amazon Kinesis Data Analytics for Apache Flink?).

   KDA's role is to read data from KDS and asynchronously invoke the SageMaker endpoint for all incoming streaming data using the API Gateway. Upon receiving the inference results, KDA will then store the results in an S3 bucket. In the background, KDA will also provide real-time data

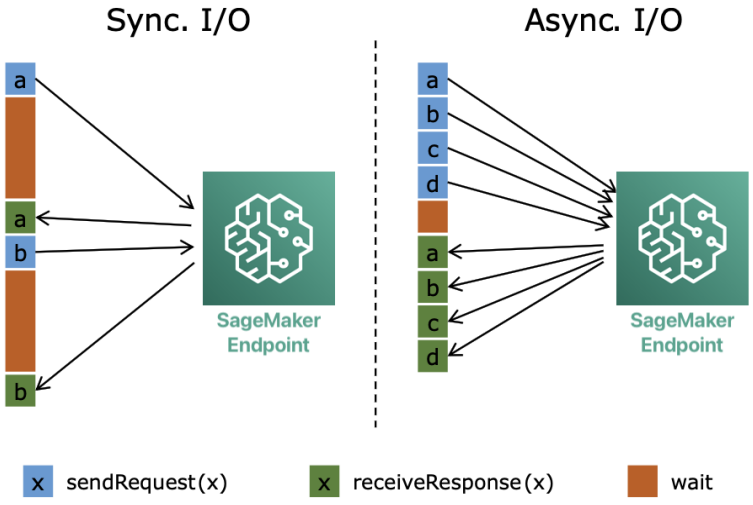
analytics on the data volume and number of records received and sent at each task.

SageMaker's role is to provide a cloud environment for training a machine learning model to predict taxi fares. After the training, SageMaker helps to deploy the model on a SageMaker real-time hosted endpoint. We can then make real-time inferences about predicted taxi fares by querying this SageMaker endpoint that can be invoked using a Lambda function.

3. Why is an *Asynchronous* I/O operator used between the AWS Kinesis Data Analytics and API Gateway in this pipeline?

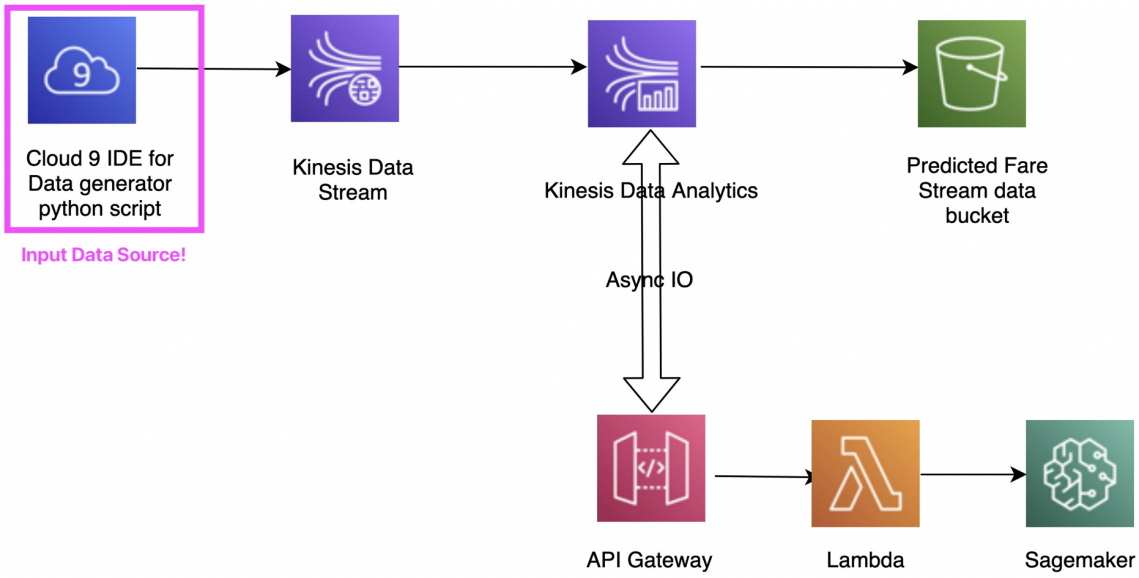
   The Async I/O operator is used because it is non-blocking and allows for interactivity with high scalability. By using Async I/O, a single parallel function instance can handle many requests concurrently and receive the responses concurrently (from Async I/O | Apache Flink). This leads to a much higher streaming throughput for our pipeline. (See image below adapted from Async I/O | Apache Flink).

   Interacting with the SageMaker endpoint to obtain the predicted taxi fares takes time. The Async I/O operator helps to ensure that the communication delay with the SageMaker endpoint does not cause the application to be blocked. This prevents the waste of precious compute resources, and allows incoming streaming data to be handled with low latency.

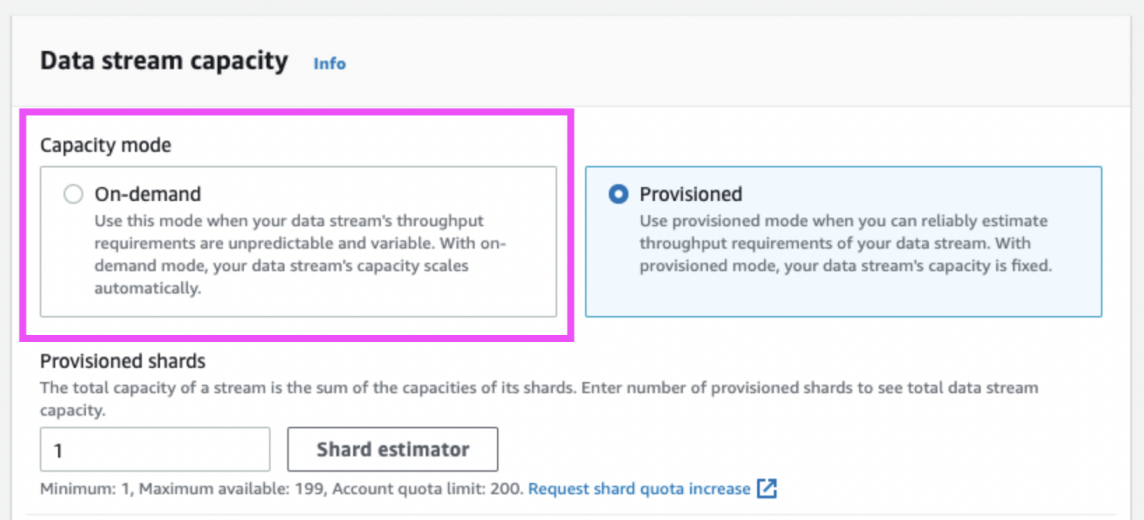


4. This demo is using simulated data created by a Python script. What changes in the pipeline does it need to ingest real-world data in a production environment? (You don't need to implement that.)

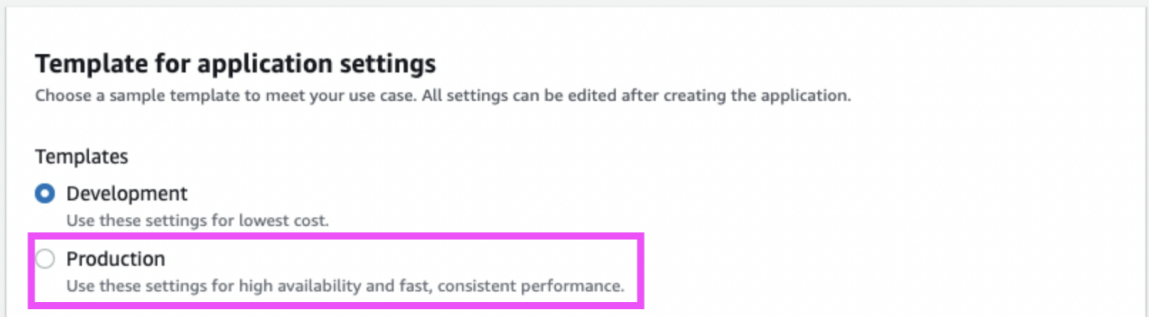
   To ingest real-world data in a production environment, we have to change the input data source to Kinesis Data Stream. To do so, the customer application can either use the AWS SDK for Python (Boto3) to put data records into the Kinesis Data Stream, or other AWS SDKs available for other programming languages. We need to ensure that the KDS can handle multiple incoming streams of data from different sources.



   Next, since we might not be able to reliably predict the throughput required, we would need to configure KDS's capacity mode to be "On-demand". This ensures that the data stream scales automatically with the amount of data received in a production environment.



Finally, we would also need to ensure that the KDA engine can handle the data load in a production environment. To do so, we would need to change the application settings from Development → Production mode. This would help to ensure that the KDA engine is optimised for high availability and fast performance.



Note how we don't have to change much of the pipeline in the cloud, but only the input data source. This is the advantage of cloud computing - the same pipeline can be used across multiple applications that require the same functionalities (e.g., data ingestion and machine learning inferences for this lab).