Real time stream processing with Kafka

Background
StopHacking is a start-up would like to develop cloud service to detect and stop computer hackers. Although they have some rule-based service to identify certain hacks, they would like to add machine learning models which can integrate with their Spark cluster to process large amounts of data and detect any potential hacks. They hired us as the Analytics Engineer to investigate the open data from the Cyber Range Labs of UNSW Canberra and build models based on the data to identify abnormal system behaviour. In addition, they want us to help them integrate the machine learning models into the streaming platform using Apache Kafka and Apache Spark Streaming to detect any real-time threats, in order to stop the hacking.

Instructions
In this project, there are two main tasks - producing streaming data and processing the streaming data.

1. In part 1 for producing the streaming data, you can use csv module or Pandas library or other libraries to read and publish the data to the Kafka stream.
2. In part 2 for consuming the streaming using Kafka consumer, you can use csv module or Pandas library or other libraries to process the ingested data from Kafka.
3. In part 3 for streaming data application, you need to use Spark Structured Streaming together with Spark ML / SQL to process the data streams. For part 3, Pandas can only be used for plotting steps, but excessive usage of Pandas for data processing is discouraged.

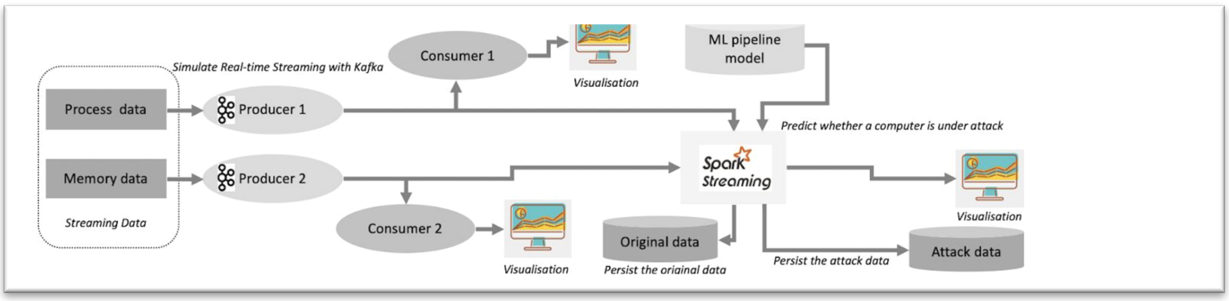
Architecture


Fig 1: Overall architecture of project.

1. Producing the data
In this task, we will implement two Apache Kafka producers (one for process and one for memory) to simulate the real-time streaming of the data.

1.1 Process Event Producer
Write a python program that loads all the data from *"Streaming_Linux_process.csv*". Save the file as *Real time stream processing with Kafka - Part 1.1 - Process Event Producer.ipynb.*

Your program should send X number of records from each machine following the sequence to the Kafka stream every 5 seconds.

- The number X should be a random number between 10-50 (inclusive), which is regenerated for each machine in each cycle.

- You will need to append event time in unix-timestamp format (as mentioned above).
- If the data is exhausted, restart from the first sequence again

1.2 Memory Event Producer
Write a python program that loads all the data from "Streaming_Linux_memory.csv". Save the file as Real time stream processing with Kafka - Part 1.2 - Memory Event Producer.

Your program should send X number of records from each machine following the sequence to the Kafka stream every 10 seconds. Meanwhile, also generate Y number of records with the same timestamp. These Y number of records would be sent after 10 seconds (or the next cycle)2 . A figure demonstrating the timeline is shown below.

- The number X should be a random number between 20~80 (inclusive), which is regenerated for each machine in each cycle.
- The number Y should be a random number between 0~5 (inclusive), which is regenerated for each machine in each cycle.
- You will need to append event time in unix-timestamp format (as mentioned above).
- If the data is exhausted, restart from the first sequence again

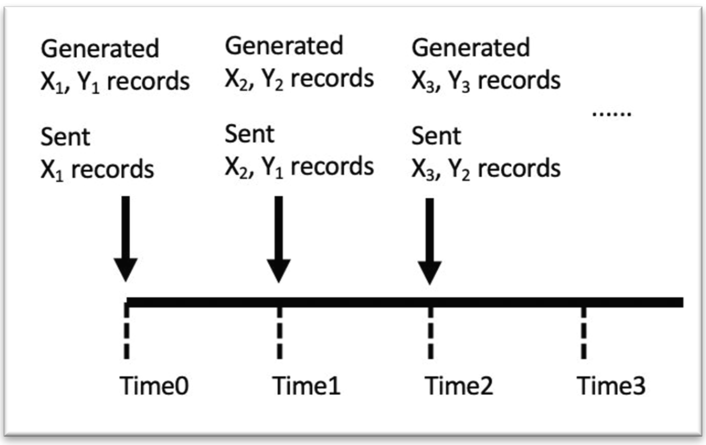


Fig 2: Timeline on the data generation and publication onto the stream

2. Consuming data using Kafka
In this task, we will implement multiple Apache Kafka consumers to consume the data from part 1.

Note:

- In this task, use Kafka consumer to consume the data from part 1.
- Do not use Spark in this task

2.1 Process Event Consumer
Write a python program that consumes the process events using kafka consumer, visualise the record counts in real time. Save the file as "Real time stream processing with Kafka - Part 2.1 - *Process Event Consumer.ipynb*." and use line charts to visualise.

2.2 Memory Event Consumer
Write a python program that consumes the memory events using kafka consumer, visualise the record counts in real time. Save the file as "*Real time stream processing with Kafka - Part 2.2 - Memory Event Consumer.ipynb*."

Your program should get the count of records arriving in the last 2 minutes (use processing time) for each machine, and use line charts to visualise.

3. Streaming application using Spark Structured Streaming

In this part, we will implement Spark Structured Streaming to consume the data from part 1 and perform predictive analytics.

Note:

- In this part, use Spark Structured Streaming together with Spark SQL and ML
- You are also provided with a set of pre-trained pipeline models, one for predicting attack in process data, another for predicting attack in memory data

Write a python program that achieves the following requirements. Save the file as "*Real time stream processing with Kafka - Part 3.*ipynb."