15-319 / 15-619
Cloud Computing

Overview 10

March 29th, 2022

Last Week’s Reflection

- **Conceptual content on OLI**
  - Module 15: Case Studies: Distributed File System
  - Module 16: Case Studies: NoSQL Databases
  - Module 17: Case Studies: Cloud Object Storage
- **Project 4: Iterative processing with spark**
- **Team Project Phase 1**

This Week

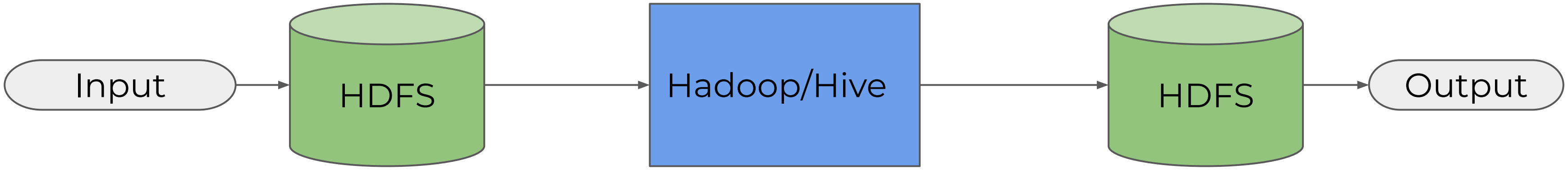
- **OLI, Unit 5: Distributed Programming and Analytics Engines for the Cloud**
    - Module 18: Introduction to Distributed Programming for the Cloud
    - Module 19: Distributed Analytics Engines for the Cloud: MapReduce
    - Module 20: Distributed Analytics Engines for the Cloud: Spark
- **Quiz 9 (OLI Module 18)**
    - Due **Friday**, April 1st, 2022, 11:59PM ET
- **Project 5 - Stream Processing with Kafka and Samza**
    - Due **Sunday**, April 10th, 2022, 11:59PM ET

Stream vs. Batch Processing

- Batch processing
  - Data parallel, graph parallel
  - Iterative, non-iterative
  - Runs once in few hours/days
  - Historical data analysis
  - Not well suited for real time events streams
- Stream processing
  - Process events as they come
  - Real time decision making
  - Sensor streams/web event data

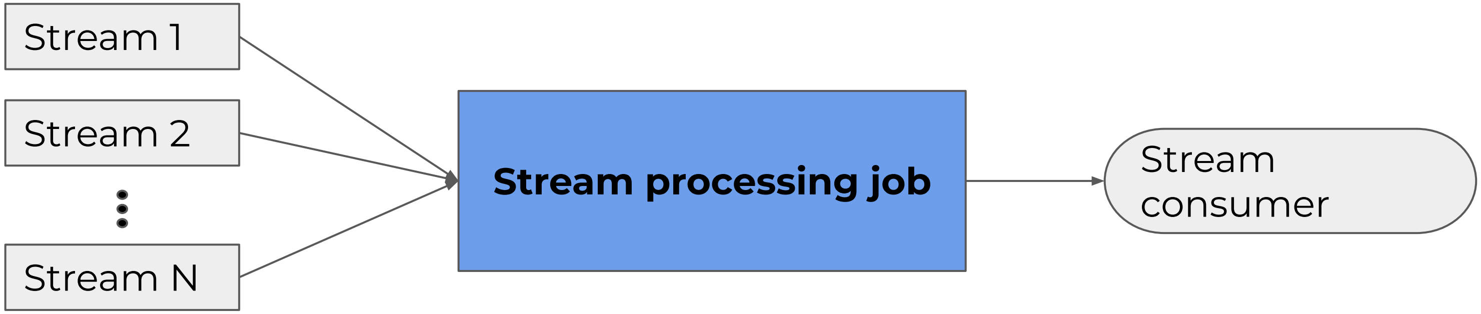
Typical Batch Processing Job

- Input is collected into static “batches” and processed holistically
    - Represents a single point in time
- Output is consumed sometime later
    - Data (analysis) retains “value” with time



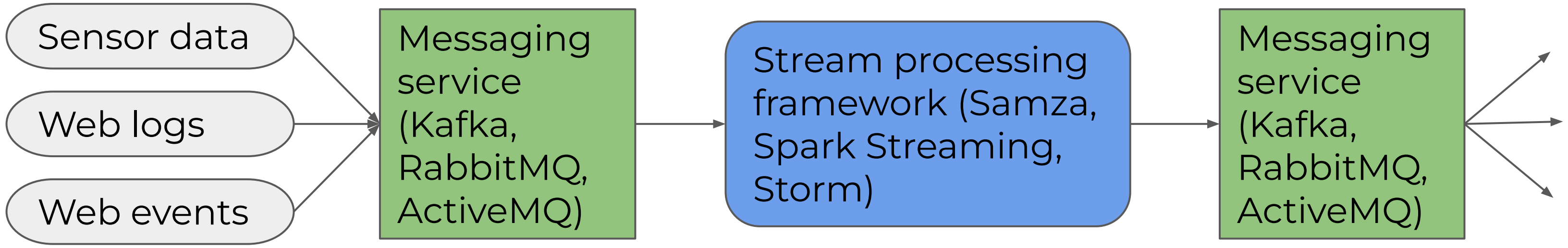
Typical Stream Processing Job

- Data is processed immediately*
  - *Upon queueing
- The processed data is available to downstream consumers for real time decision/analytics



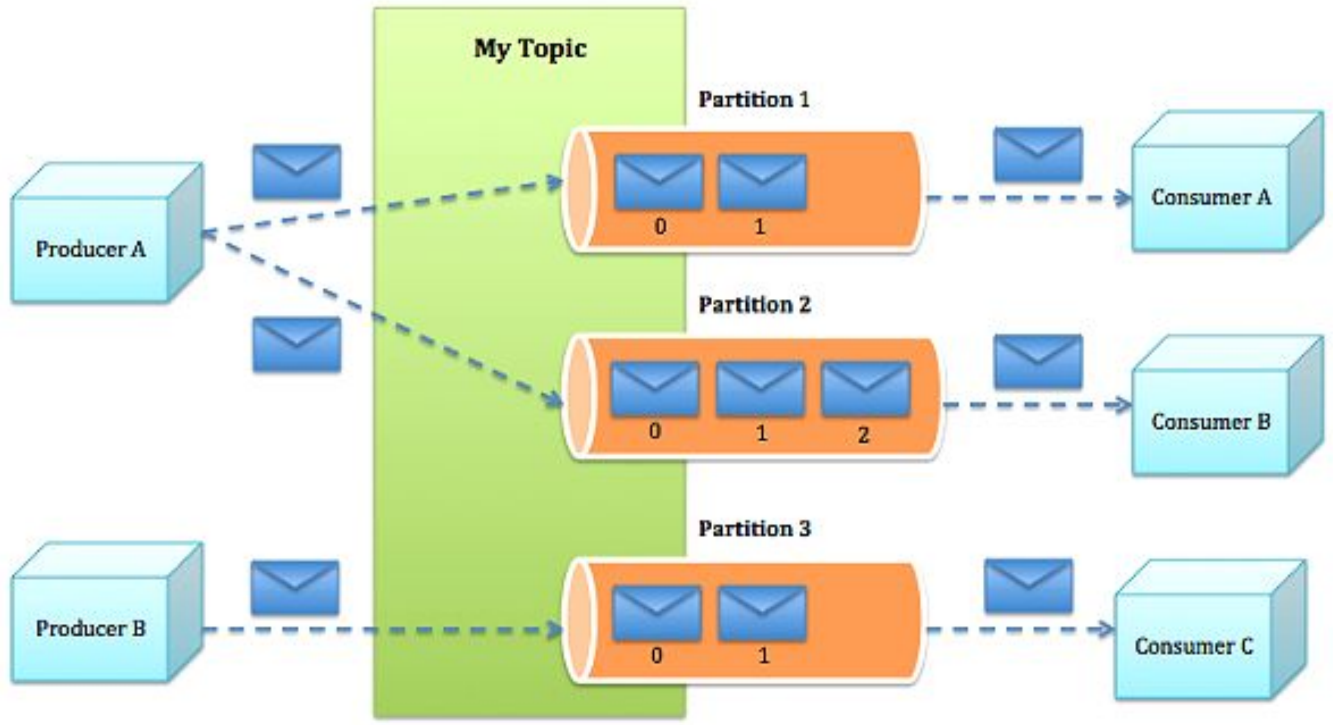
Components of a Stream Processing Job

- An event producer - Sensors, web logs, web events
- A messaging service - Kafka, RabbitMQ, ActiveMQ
- A stream processing framework - Samza, Storm, Spark Streaming


Sensor data
Web logs
Web events
Messaging service (Kafka, RabbitMQ, ActiveMQ)
Stream processing framework (Samza, Spark Streaming, Storm)
Messaging service (Kafka, RabbitMQ, ActiveMQ)

Apache Kafka

- A distributed messaging system developed at LinkedIn.

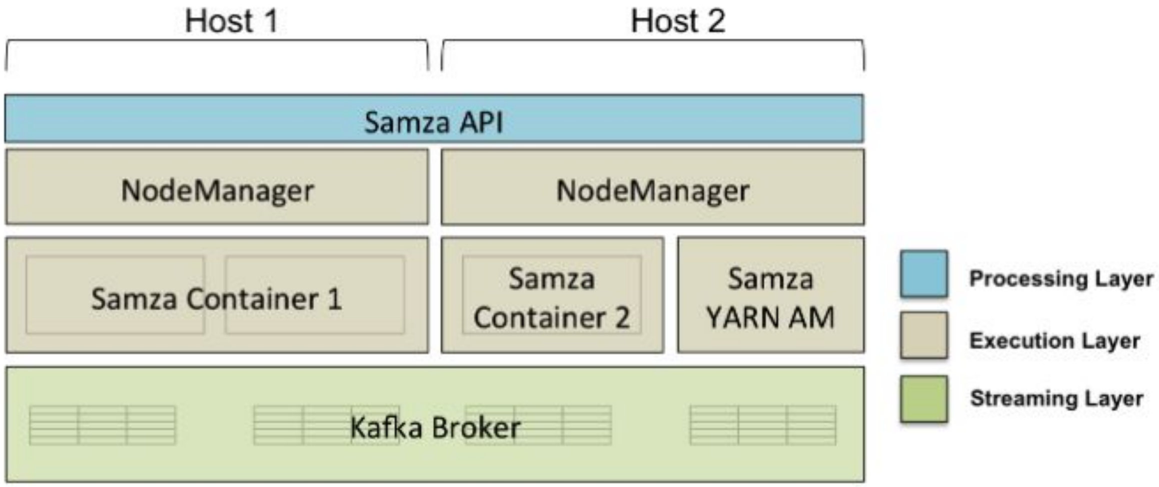


Semantic partitioning in Kafka

- Each topic (stream) is partitioned for scalability across all nodes in the Kafka cluster
- Default partitioning attempts message load balancing
- Streams can also be partitioned semantically by user - key of the message
- All messages with the same key arrive to the same partition
- Fault-tolerance: Replication
  - One leader and zero/more followers
  - Replication factor
  - ISR (in-sync replicas)

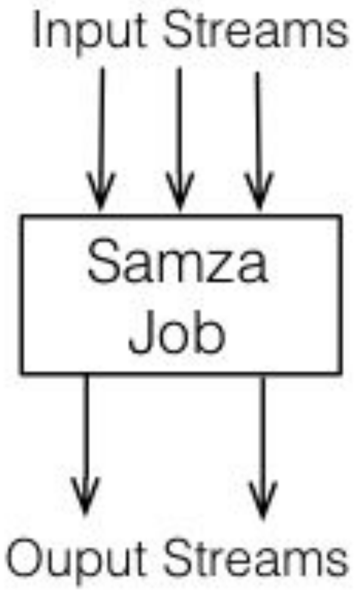
Apache Samza

- Stream processing framework developed at LinkedIn
- Consists of 3 layers:
  - Streaming layer
  - Execution layer
  - Processing (Samza) layer
- Most common use: Kafka for streaming, YARN for execution



Partitioning in Apache Samza

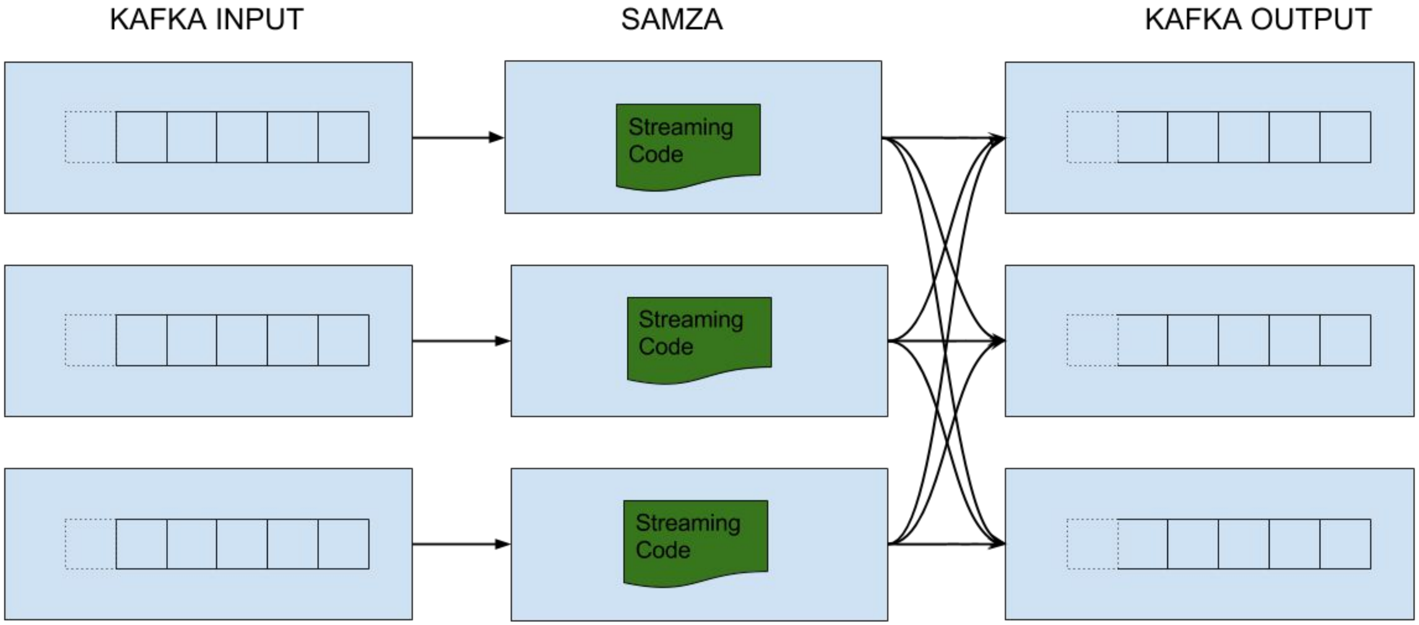
- Programmer uses the Samza API to perform stream processing
- Each partition in Kafka is assigned to a single Samza task instance



Stateful stream processing in Apache Samza

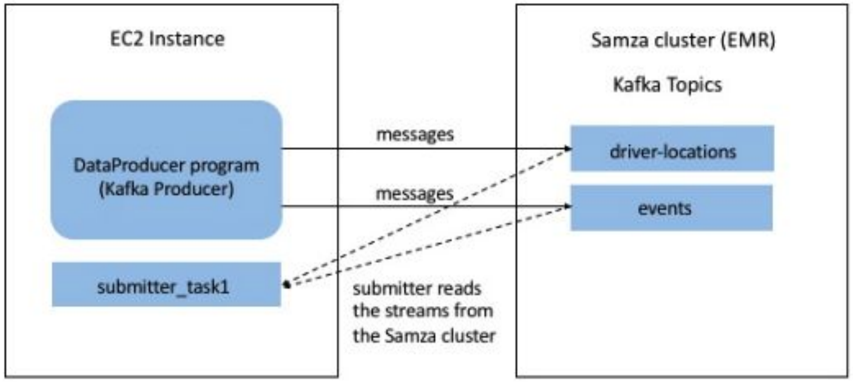
- Calculate sum, avg, count, etc.
- State in remote data store? - slow
- State in local memory? - machine might crash
- Solution - persistent KV store provided by Samza
  - Changes to KV store persisted to a different stream (usually Kafka) - replay on failure
  - RocksDB currently supported as a persistent KV store
  - **You MUST use a persistent KV store for P5!**

Kafka and Samza, Together



Project 5 - NYCabs (NYC based Taxi Service)

- Stream Processing with Kafka/Samza
  - Stream 1: Car GPS coordinates
  - Stream 2: Riders
- Task
  - Match riders with drivers to minimize travel time & other constraints
- Using AWS
  - **Proper tags and track budget**

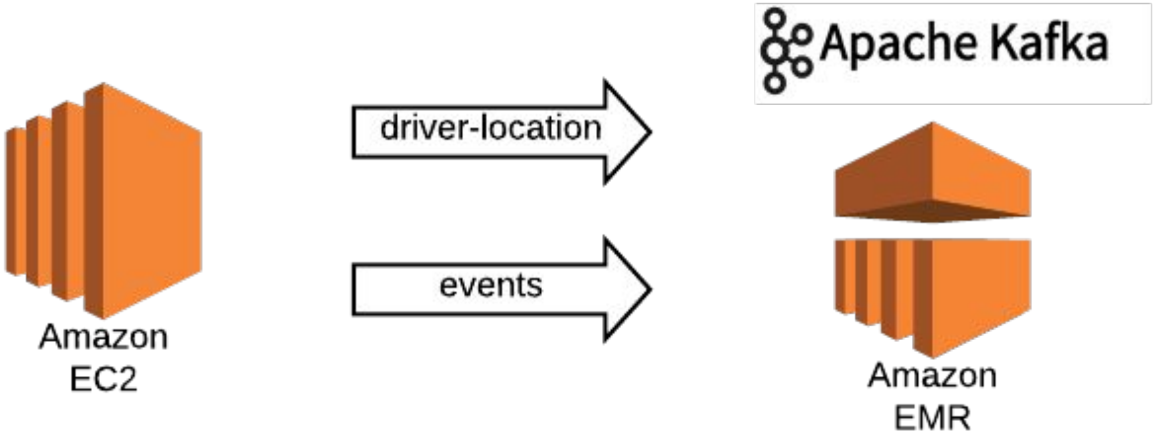


Task 1

- Simulate the scenario that the **drivers** update their locations on a regular basis as they move in the city and the **clients** request rides at some time.
- Data
  - Tracefile -> Two streams
  - DRIVER_LOCATION -> driver_locations stream
  - LEAVING_BLOCK, ENTERING_BLOCK, RIDE_REQUEST, RIDE_COMPLETE -> events stream

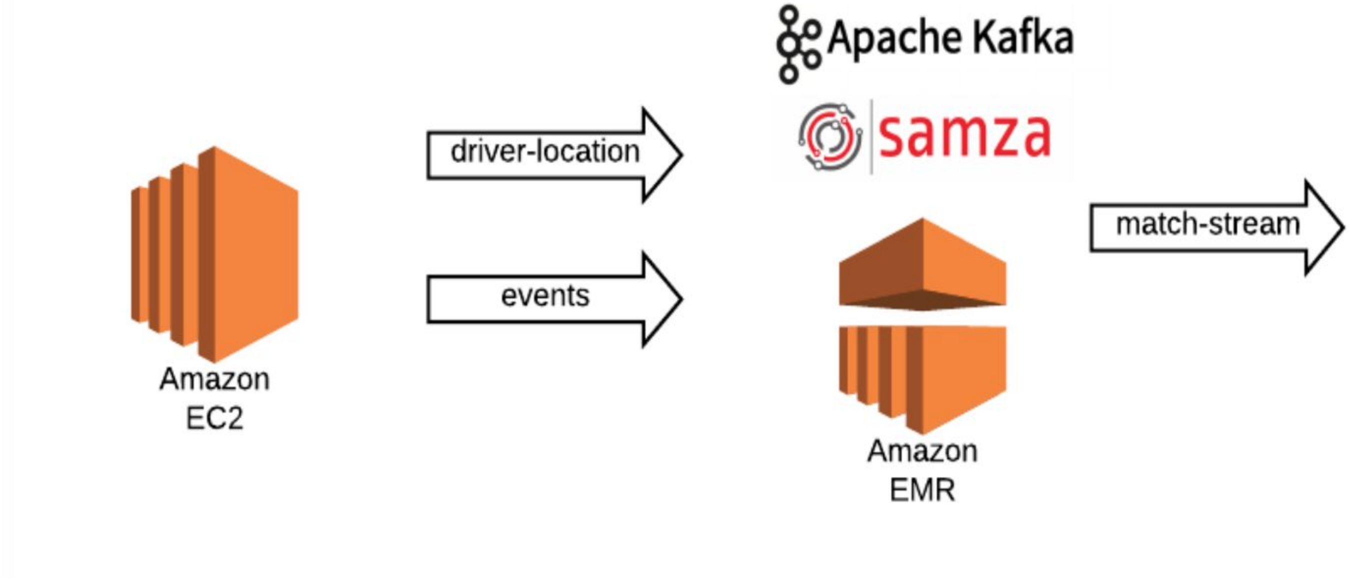
Task 1

- You will run your producer program on your student AMI instance
- The producer program will publish the data into Kafka brokers.
- The submitter for Task 1 is located on the student AMI instance.



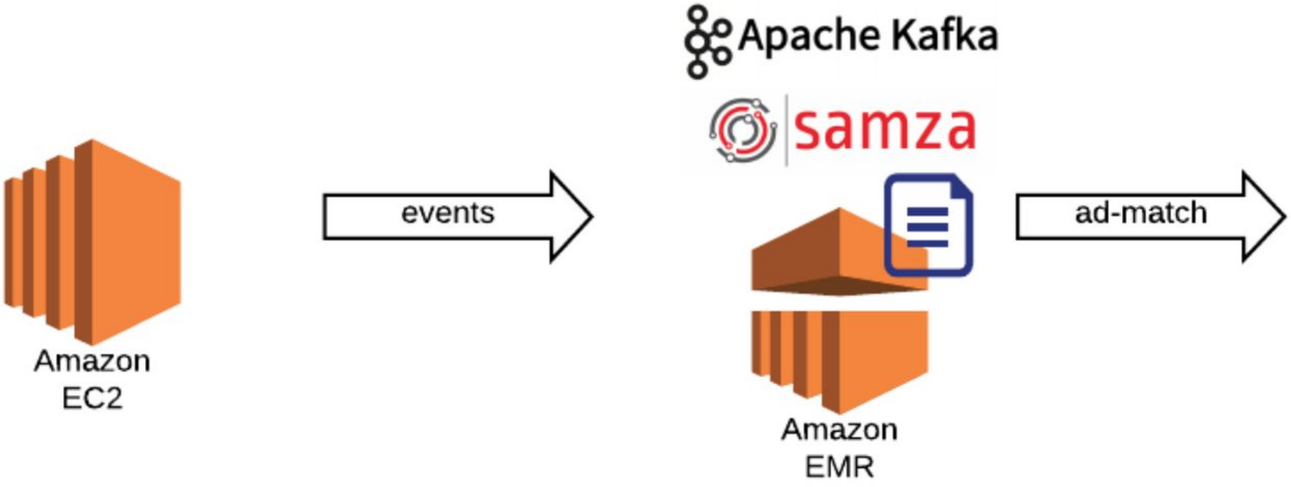
Task 2

- Use the same producer program used in Task 1.
- Find the best match for a ride request with a driver located in the **same block** as the rider based on published data



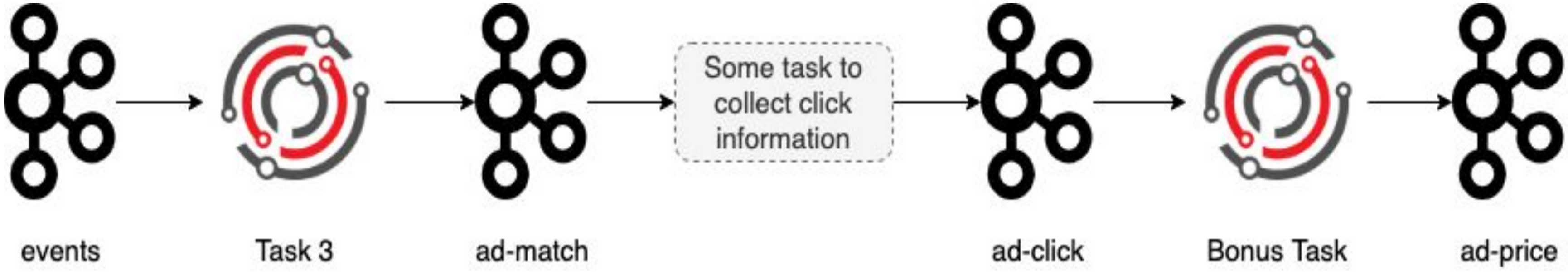
Task 3

- Find the best advertisement to place for a specific user.
- Utilize BOTH static data(user profile, health status and interests) and stream data to make this decision.



Bonus Task

- Find the advertisement price that a restaurant is paying to NYCabs and the advertising company.
- Write **at least 2** unit test test cases.



Debugging (Important!!)

- **Use the YARN UI without opening unsafe ports to public**
  - Ssh tunneling
  - Opening the port only to your IP address using AWS console
- Debugging with YARN logs
  - Yarn commands
    - >>> yarn application -list
  - From YARN UI
  - From container logs
- Output a kafka stream for debugging
- Refer Introduction to Kafka and Samza primer for detailed steps

Cloud Security Note

- **Never open unsafe ports of a Hadoop cluster to public, or it will get compromised and implicated in DDOS attacks, cryptomining, etc.**
- In general, failing to protect a cloud resource can get it compromised and exploited, such as becoming a source to launch DDoS attack
  - A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of server
  - DDoS attacks achieve effectiveness by utilizing as many compromised computer systems
  - DDoS attacks are often used as leverage for blackmail, therefore hacking groups are motivated to grow their DDOS capacity

Upcoming Deadlines

- **Quiz 9 (OLI Module 18)**
  - Due **Friday**, April 1st, 2022, 11:59PM ET
- **Project 5 - Stream Processing with Kafka and Samza**
  - Due **Sunday**, April 10th, 2022, 11:59PM ET
- **Team Project Phase 2 Live Test**
  - Due **Sunday**, April 3rd, 2022, 11:59PM ET
- **Team Project Phase 2**
  - Due **Tuesday**, April 5th, 2022, 11:59PM ET