Homework-4: Apache Kafka

Deadline: March 12th, 11:59PM ET/8:59pm PT.

In this homework, you are expected to practice Apache Spark. It's expected you will apply the practices and understandings provided in the lecture materials in addition to self-learning to complete the homework questions. Apply best practices in solving every question and design your application for scalability.

Use the following GitHub classroom to access the assignment and create your assignment repository: https://classroom.github.com/a/kHxQQ_lw

Cite any external sources you use. External sources shouldn't exceed more than 30% of the final solution.

You should submit the URL for your GitHub repository on Canvas. Grading penalty of 5 points will be applied if otherwise. Students are responsible for making sure they have uploaded all materials to GitHub on-time. For 3 days after the due date, a late penalty will be applied according to the guidelines in the Syllabus.

Q1. YouTube Video Analysis Based on Comment Popularity (40 points):

- Enhance the YouTube demonstration from the lecture by developing a feature that identifies the most popular video out of up to five videos. This will be determined by calculating which video has the highest total number of likes on its comments.
- Design your application to prompt the user to input up to five YouTube video IDs.
- Ensure your application aggregates the total number of likes from the comments on each video and displays the title of the video that has accumulated the most likes on its comments.
    - You may assume that calling commentThreads().list() function once is sufficient for retrieving all the comments that you need to consider for your analysis.
    - Ensure that all communications with the YouTube API are confined to the Producer script to avoid redundant API calls from other areas of the code.
    - You can print the output on a terminal shell or a webpage.
- Your application must use Apache Kafka (or Confluent Kafka).
- Hint: You may find the YouTube API reference useful: https://developers.google.com/youtube/v3/docs/

For Q1., submit the following:

1. All the code files you developed/used.

2. Screenshot of the “Messages” tab of your Kafka topic showing sample messages published to your Kafka topic.
    - Hint: Open “Messages” tab before you start the Producer to view the messages properly.
3. Screenshot of the output of the consumer.
    - If your output is not printed in a Jupyter notebook, take a screenshot of the output and make sure your name is written in a Text Editor in the screenshot (Don’t photoshop your name).

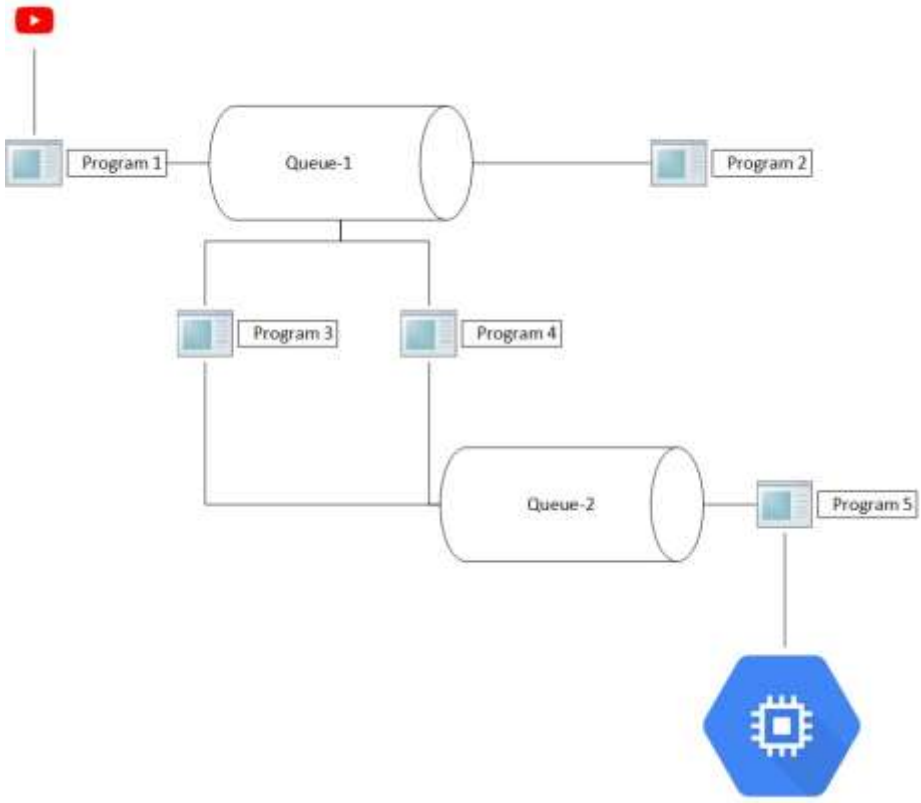
Q2. Apache Kafka in DevOps Pipeline (30 points):

- **Task Overview:** You are to build and implement the pipeline as depicted in the provided diagram. Your primary objectives are to design and implement the information flow between various programs to both meet the requirements specified and optimize the application's performance.
- Program-1 Requirements:
    - Program-1 should accept up to five different YouTube video IDs from the user.
    - It will retrieve comments, the number of likes on each comment, and any other relevant information for the requested YouTube videos.
    - Ensure that all communications with the YouTube API are confined to Program-1 to avoid redundant API calls from other programs.
- Program-2 Operations:
    - Program-2 will process messages from Queue-1.
    - It must output the titles of the YouTube videos along with the aggregated number of likes from the comments of each video.
    - Program-2 may output on a console shell or a Web application page.
- Program-3 and Program-4:
    - Both programs will manage load balancing for messages in Queue-1, including those consumed by Program-2.
    - Program-3 and Program-4 are each responsible for processing all comments pertaining to a given video. In other words, Program-3 must process **ALL** the comments for a given video and not let Program-4 process any comments for that specific video (and vice versa).
    - Optimize your message queue setup to facilitate efficient distribution of tasks between these two programs.
    - Message Handling in Program-3 and Program-4:
        - Publish only those comments that have a positive like count to Queue-2.
- Program-5 and VM Instance Creation:
    - Program-5 will consume all messages from Queue-2.
    - It will create a VM instance on Google Cloud. This VM should be named after the author of the comment with the highest like count.

        - You can remove any special characters from the author's name to avoid any issues/challenges with creating the VM instance.
        - You can use the author's YouTube handle (or username) if it's easier to fetch than the author's name.
- Your application must use Apache Kafka (or Confluent Kafka) for message queues.
  Hint: This tutorial lists the steps of creating VM (You need to add authentication): https://cloud.google.com/compute/docs/instances/create-start-instance

For Q2., submit the following:

1. All the code files you developed/used.
4. Screenshots of the "Messages" tab of your Kafka Queue-1 and Queue-2 topics showing sample messages published to your Kafka topics.
2. A screenshot of the output of Program-2.
    - If your output is not printed in a Jupyter notebook, take a screenshot of the output and make sure your name is written in a Text Editor in the screenshot (Don't photoshop your name).
3. A screenshot of the VM instance created by Program-5.
    - Note that if you created the VM manually and not via the code (and you didn't conclude the name of the VM instance programmatically, you will receive no points).



Q3. Update Microservice Orchestration App to use Message Queues (30 points):

- Improve the reliability of your HW-3 microservice orchestration application by implementing message queues.
- Position the message queue specifically between the WebApp microservice and the Logic microservice for this assignment.
- The microservice orchestration application and the message queue(s) should be deployed to Google Kubernetes Engine.
- There is no requirement to establish additional message queues between other microservices in the application.
- If you don't have a functional microservice orchestration application that you would like to use for this exercise, you may use the following repository as a base for your solution:
  https://github.com/mohamedfarag/14-848-supplementary-task
  Note: URL will be available starting Oct. 7th.
- You may find the following tutorial useful to add Kafka support to Java:
  https://developer.confluent.io/get-started/java/#create-project
    - Feel free to search and use other tutorials as needed.

For Q3., submit the following:

1. A copy of all the code files you developed/used.
2. A video demonstration with the following requirements:
    - Provide a comprehensive video demonstration that showcases your application and its message queue functionality deployed on Google Kubernetes Engine (GKE).
    - Execute the application by initiating a sample message through the user interface and demonstrate waiting for and receiving the response from the backend.

- Ensure the video clearly verifies your authorship of the project. You can achieve this by either showing your Google Cloud Platform (GCP) account during the video or by displaying your name in a text editor while recording the demonstration. This step is crucial to confirm that you personally created and demonstrated the work.

3. Screenshot of the “Messages” tab of your Kafka topic showing sample messages published to your Kafka topic. These messages should represent the communication between the WebApp microservice and the Logic microservice.

Don’t forget to disable billing after you finish using GCP