ACIT 3855 – Service Oriented Architectures – Assignment 1

Instructor	Mike Mulder (mmulder10@bcit.ca) – Sets A and C Tim Guicherd (tguicherd@bcit.ca) – Set B
Total Marks	20
Due Dates	End of Last Class: • April 8th for Set C • April 11th for Sets A and B **It can be demoed concurrently with the last lab if necessary.**

<u>This assignment is to be completed and submitted individually.</u>

This assignment requires that you modify your existing services and create a new service. Please treat this as practice for the final exam as you will also have to code a new service that interacts with your existing services.

You will be creating an Event Logging service that receives events logs from other services and writes them to its data store and log file. This will be used by system administrators to monitor specific events in the system.

Update to Kafka

Create a new Kafka topic called ***event_log*** with one partition and one replica. This can be done in the kafka service in your docker-compose.yml file.

Updates to Existing Backend Services

The following services should be setup as producers for the event_log topic:

> **Receiver** – It should publish a message to event_log after it successfully starts and connects to Kafka indicating that it is ready to receive messages on its RESTful API. The code for this message is 0001, which should be included in the message text.
>
> **Storage** - It should publish a message to event_log after it successfully starts and connects to Kafka indicating that it is ready to consume messages from the events topic on Kafka. The code for this message is 0002.
>
> **Processor** - It should publish a message to event_log in the following two cases:
> - After it successfully starts up. The code for this message is 0003.
> - On periodic processing if it receives more than a configurable number of messages. The default is 25 for this configurable value, but it may have to be tuned based on how fast your system is at processing events. The code for this message is 0004.

New Service

Create a new Event Logger service running on Port 8120.

It should be setup as a consumer for the event_log topic. Similar to the Storage service and the events topic, it should consume new messages and keep track of its offset.

When it consumes one of the event log messages from the topic, it should write it to 1) it log file and 2) a database.

The database can be SQLite or a JSON data file. For each event log message in the database, it must include:

- A unique ID of the record
- The message itself
- The message code
- The date/time the message was written to the database

The Event Logger service will have a single API endpoint: GET /events_stats

The endpoint will return the number of events of each code:

GET /event_stats

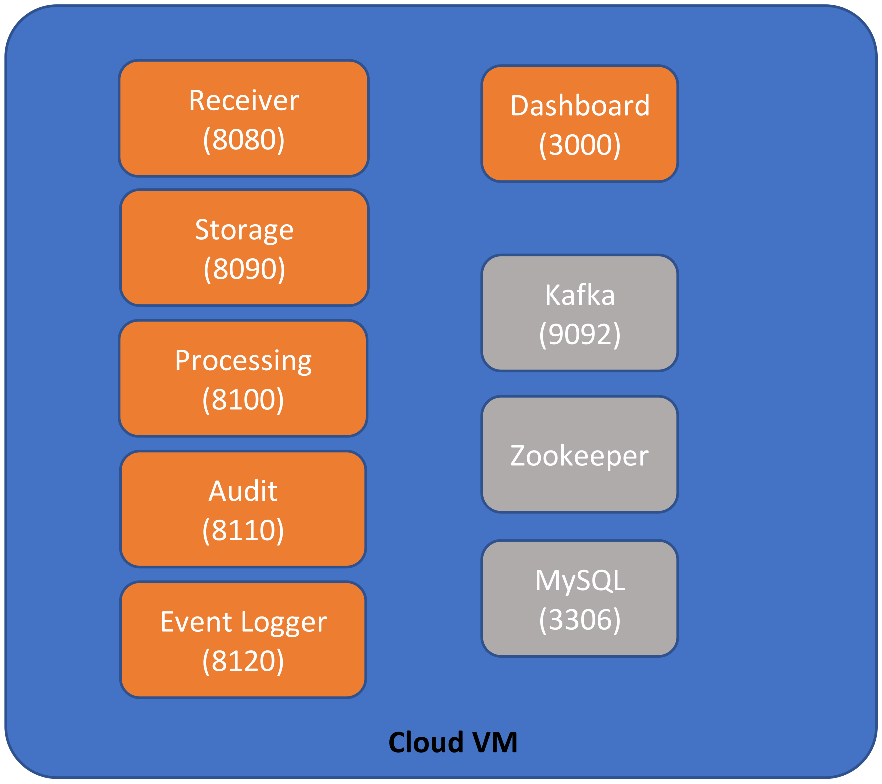
```
{
    “0001”: 10,
    “0002”: 22,
    “0003”: 12,
    “0004”: 98
}
```

Make sure to include the following in your implementation of the service:

- Configurations
- Logging
- Database creation scripts if using SQLite
- Follows good Python coding conventions (i.e., PEP-8)

Make sure all data files, API endpoints, etc. are maintained in a configuration file.

Deploy your Event Logger service as part of your system. This means it must have a dockerfile, requirements.txt and be integrated in your docker-compose.yml file.



Updates to Existing Frontend Dashboard

The Dashboard should show the current counts of the event log messages of each type. Like the processor statistics, the statuses should be updated periodically.

It should reflect the data in the API JSON response, but should be formatted nicely, such as:

0001 Events Logged: 10
0002 Events Logged: 22
0003 Events Logged: 12
0004 Events Logged: 98

Submission

Upload a zip file containing the code for your Event Logger service, including:

- app.py
- application and log configuration files
- requirements.txt and dockerfile
- OpenAPI specification
- Database creation script (if applicable)
- Updated docker-compose.yaml file

Evaluation

The grades for each item will be based on the quality of the demo and submission.

Demonstration • Added Kafka topic to Kafka and existing services. Events of the 4 codes are published by the existing services (2 marks) • New Event Logger service is running as part of your system as a Docker container that can be started and stopped with Docker Compose (2 marks) • Event Logger counts for each code are correct and available through the API and Dashboard (2 marks) • Event log messages consumed from Kafka are recorded to the app.log and database of the Event Logger service (2 marks) • Database persists to a docker volume (2 marks)	10
Event Logger Service Code • Code is correct in terms of the RESTful API and data persistence (2 marks) • Uses OpenAPI, Kafka, JSON OR Sqlite for DB, Docker for deployment (2 marks) • Code uses logging and external configurations (2 marks) • Code Follows Coding Standards (Naming, DocString, Etc) (2 marks) • Code is logical and well-organized (2 marks)	10
Total	**20 marks**