**Commerce Microservices Bootcamp**

**Demo Assignment Questions**

## **Objective:**

The objective of the assignment is to provide exercises that cover the key technical concepts required to perform BY Commerce microservice implementation. Videos/tutorials shared along with the assignment will help participants to bridge their knowledge gap.

Participants need to complete three assignments. Each assignment builds on top of the other and hence, it is advised to implement them in sequence. (Assignment 1,2,3).

## **Software pre-requisites:**

The following list of software required for the assignment. Please download and install them locally.

* Intellij
* Apache Cassandra
* Kafka and Zookeeper
* Springboot project with all required dependencies. For example,
  + Spring webflux
* Maven to be used as the build tool.

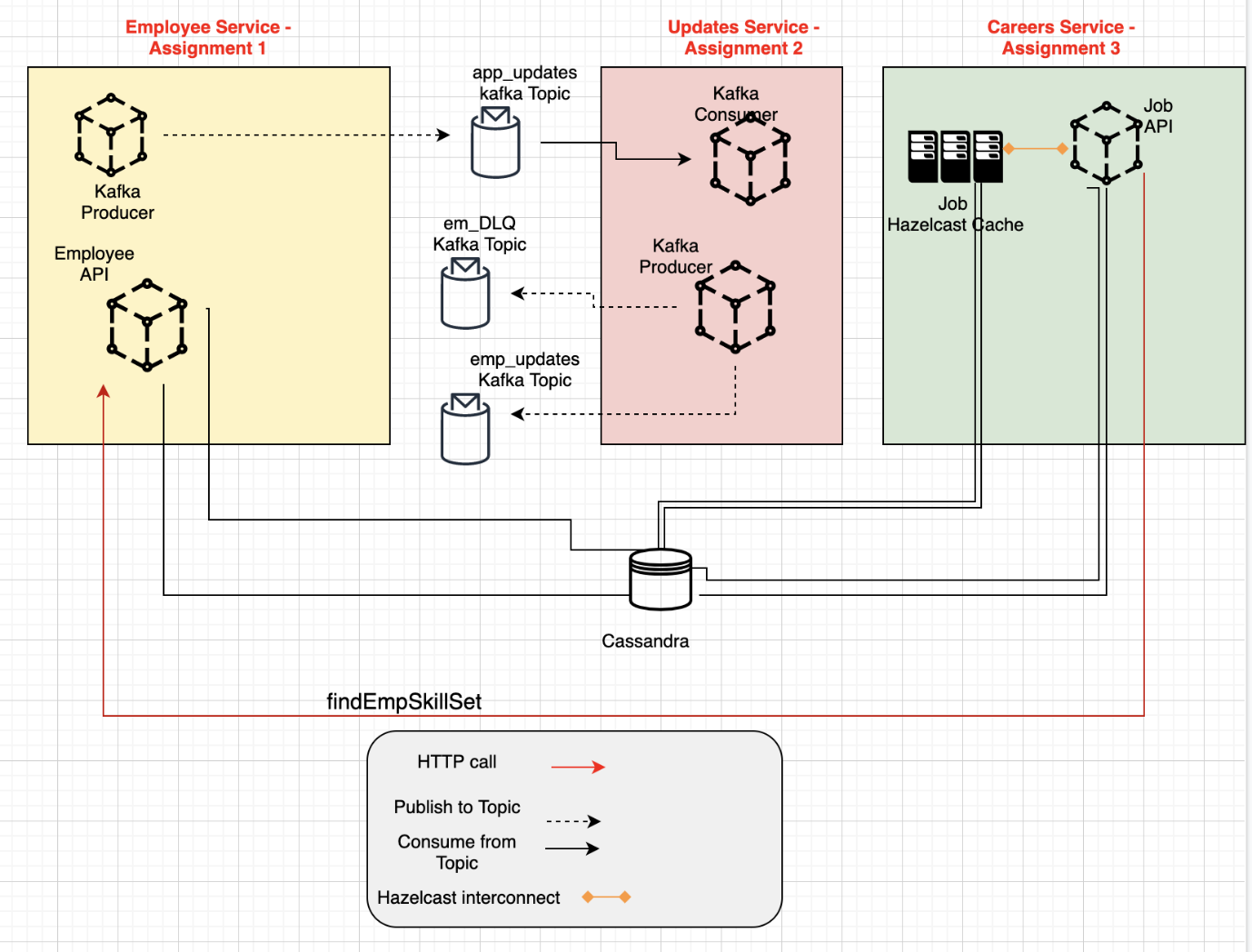
## **Assignment Implementation Guidelines**

All Springboot applications created as part of the assignment should follow the below guidelines. Further details can be found in the respective assignment sections.

* Reactive non-blocking implementation of Springboot to be used (Spring Webflux - Flux and Mono)
* Cassandra to be used as the persistence layer. Reactive Cassandra interface to be implemented.
* Kafka Consumer /Producer code to be written in the reactive non-blocking style.
* Invocations, across microservices to follow reactive non-blocking style.
* Java Coding standards to be followed.

## **Assignment System Architecture Diagram**

Three microservices would be built as part of the assignment. Below is a pictorial visualization of the same.



## **Assignment Overview:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assignment** | **SpringBootApplication** | **Exposed REST Endpoints** | **REST endpoints invoked** | **Kafka Producer/Consumer** |
| Assignment -1 | EmployeeService | createEmployee | NA | Kafka Producer  Topic – app\_updates |
| findEmpSkillSet | NA |  |
| Assignment - 2 | UpdatesService |  |  | Kafka Consumer  Read from: app\_updates  Kafka Producer  Write to:  emp\_updates  emp\_DLQ |
| Assignment - 3 | CareersService | findEmpForJobID | findEmpSkillSet (Assignment 1) |  |
| createJobProfile | NA |  |
| getJobProfileFromCache |  |  |

# **Assignment 1**

## **Prerequisite Applications**

* Integrated development environment – IntelliJ or Eclipse
* Apache Cassandra
* Spring Web Flux
* Kafka

## **Business Use Case:**

* **Scenario –1:** Implement a Springboot REST endpoint - createEmployee - that will persist the employee information into Cassandra database.
* **Scenario –2:** Implement a Kafka producer that posts the employee information received from the createEmployee REST endpoint into a Kafka topic.
* **Scenario –3:** Implement a Springboot REST endpoint - findEmpSkillSet - that fetches a list if employees from Cassandra matching the input.

## **High level Steps**

* Create a Cassandra Schema
  + CREATE KEYSPACE bootcamp WITH replication={'class':'SimpleStrategy','replication\_factor':1}
* Create employee and skillset table.
  + CREATE TABLE emp( emp\_id int, emp\_name text, emp\_city text, emp\_phone text, PRIMARY KEY (emp\_id)).
  + CREATE TABLE emp\_skill( emp\_id int, java\_exp double,spring\_exp double, PRIMARY KEY ((emp\_id,java\_exp, spring\_exp)));
    - Please use ALLOW FILTERING for quesries as needed

## **Scenario-1**

* Implement a Springboot REST endpoint createEmployee that will perform the following.
  + Persist the employee information into emp table if all the values are non-null values.
  + Persist skillset information into emp\_skill table, if all the values are non-null values.
  + REST endpoint: <http://localhost:8080/createEmployee>
  + Sample input and output

|  |  |  |
| --- | --- | --- |
| Scenario | Sample Input | Sample Output |
| Create new employee. | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0  } | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0,  "status":"Created"  } |
| Employee already exists. | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0  } | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0,  "status":"Already Exists".  } |

## **Scenario-2**

* Implement a Kafka producer to post the message received from the above REST endpoint (createEmployee endpoint) into a Kafka topic.
  + Post the message in topic named “app\_updates”
  + Sample input and output

|  |  |  |
| --- | --- | --- |
| Scenario | Message | Kafka Topic |
| Post the new employee information to Kafka topic. | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0  } | app\_updates |

## **Scenario-3**

* Implement a Springboot REST endpoint findEmpSkillset that will perform the following.
  + Looks for matching skillsets from the Cassandra DB – emp and emp\_skill table.
  + Provide a list of employees that have a matching skillset.
  + REST endpoint: http://localhost:8080/findEmpSkillset

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Sample Input | Sample Output |  |
| Get list of employees, having minimum years of experience in the technology mentioned. Java\_exp >= 3 | {"java\_exp":3  } | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0  } |  |

# **Assignment 2**

This assignment builds on Assignment 1. Please complete Assignment 1 before attempting this assignment.

## **Prerequisite Applications/Dependencies**

* Integrated development environment – IntelliJ or Eclipse
* Spring Web Flux
* Embedded Kafka

## **Business Use Case:**

**Scenario -** Process the employee information consumed from "app\_updates” Kafka topic (created by producer in Assignment 1. Validate all required fields (None of the fields should be null). Once validated post the message to an employee\_updates topic. If it is an invalid message, post to employee\_DLQ topic.

## **High level Steps**

* Create a java spring boot application.

## **Scenario**

* Create the Kafka consumer to consume messages from app\_updates topic.
* Validate if the message is valid or not.
* Once validated post the message to employee\_updates Kafka topic
* Failures to be reported to DLQ (Dead Letter Queue) topic

|  |  |  |
| --- | --- | --- |
| Scenario | Message | Kafka Topic |
| Post the new employee information to Kafka topic. | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0  } | Read from: app\_updates  Post to employee\_updates topic/employee\_DLQ topic |

# **Assignment 3**

This assignment builds on Assignment 1. Please make sure Assignment 1 is completed.

## **Prerequisite Applications**

* Integrated development environment – IntelliJ or Eclipse
* Cassandra
* Hazelcast Embedded (Add dependency in pom.xml)

## **Business Use Case:**

* **Scenario – 1** 
  + **I**mplement a Springboot REST endpoint createJobProfile that will perform the following.
    - Persist job profile information, provided as input, into Cassandra table - “job”.
* **Scenario – 2** 
  + Implement a Springboot REST endpoint findEmpForJobID that will perform the following.
  + Find the years of experience required for the provided JobId, from Cassandra (job table)
  + Makes a HTTP non-blocking call to EmployeeService microservice using the endpoint - findEmpSkillSet
  + Returns all employees that match the information.
* **Scenario – 3**
  + Implement a Springboot REST Endpoint that performs operations on the Hazelcast cache for Job table.
  + Queries Hazelcast cache for Job information (by jobID)

## **High level Steps**

* Create a table in Cassandra as follows.
  + CREATE TABLE job( job\_id text, job  
    \_name text, java\_exp double,spring\_exp double, PRIMARY KEY (job\_id,java\_exp, spring\_exp)));
* Create a java spring boot application.

## **Scenario -1**

* Create an endpoint - createJobProfile that will perform the following.
  + Persist the job information into job table.
  + REST endpoint: [http://localhost:8080/](http://localhost:8080/createEmployee) createJobProfile (POST)
  + Sample input and output

|  |  |  |
| --- | --- | --- |
| Scenario | Sample Input | Sample Output |
| Create new job profile. | {  "job\_id":1,  "job\_name":"Developer",  "java\_exp": 3.0,  "spring\_exp":0  } | {  "job\_id":1,  "job\_name":"Developer",  "java\_exp":3.0,  "spring\_exp":0,  "status":"Created"  } |
| Job profile already exists. | {  "job\_id":1,  "job\_name":"Developer",  "java\_exp" : 3.0,  "spring\_exp":0  } | {  "job\_id":1,  "job\_name":"Developer",  "java\_exp":3.0,  "spring\_exp":0  "status":"Already Exists".  } |

## **Scenario -2**

* + Implement a Springboot REST endpoint findEmpForJobID that will perform the following.
    - Find the years of experience required for the provided JobId, from Cassandra (job table)
    - Makes a HTTP non-blocking call to EmployeeService microservice using the endpoint – findEmpSkillSet
      * Call the REST endpoint: <http://localhost:8080/findEmpSkillset> from EmployeeServices microservice, created in Assignment 1, in a non-blocking manner (Please refer assignment 1)

|  |  |  |
| --- | --- | --- |
| Scenario | Sample Input | Sample Output |
| Get list of employees matching the exp - findEmpForJobID | {"job\_id":1  } | {  "emp\_id":1,  "emp\_name":"Bob",  "emp\_city":"Bangalore",  "emp\_phone":"988785614",  "java\_exp":3.5,  "spring\_exp":0  } |

## **Scenario -3**

* + Create a Springboot application endpoint – getJobProfileFromCache that will query Hazelcast Job cache to get the information.
    - Create an instance of Hazelcast Client (Add hazelcast dependency in pom.xml). Please use embedded Hazelcast mode.
    - Query Hazelcast Cache to see if the value exists.
    - If it does not exist, query Cassandra to get the necessary values and load it into the cache.
    - On invoking the endpoint, a second time around, the data should be fetched from Hazelcast Cache.

|  |  |  |
| --- | --- | --- |
| Scenario | Sample Input | Sample Output |
| getJobProfileFromCache | {"job\_id":1  } | {  "job\_id":1,  "job\_name":"Developer",  "java\_exp":3.5,  "spring\_exp":0  } |