**Web Application with Kafka on Kubernetes**

**Project Description:**

* We created a Student Survey Data Broker that allows to publish and consume student survey records at scale. This entails setting up a Kafka cluster, creating a Kafka topic, developing producer/consumer applications to publish and read messages to/from the topic, and integrating this solution with the Angular application deployed on a kubernetes cluster running on a Amazon EKS.
* We have implemented student data broker using Strimzi Kafka operator to setup a Kafka cluster with three Kafka brokers and three ZooKeepers on a Kubernetes cluster. The data broker supports message ordering (first-in-first-out) on a per partition basis
* We deployed broker on the Kubernetes cluster and made it accessible using load balancer. The broke uses persistent storage to save student survey records. The persistent storage is implemented using persistent volume claim and storage class concepts of Kubernetes.

**Instruction Steps:**

**1. Installing Kafka Cluster and operator in kubernetes environment**

* Download kafka cluster operator using the Helm charts 3 as mentioned in Strimzi kafka official website.
* Unzip the downloaded tgz file and then install it using the kubectl commands.
* Wait until cluster operator installs and a pod for cluster operator starts running.
* Now kafka persistent yaml file present in ppt slides or from github links for kafka persistent, then change yaml file according to your usecase and then use kubectl apply.
* Wait for few minutes until kafka cluster and zookeeper starts running. Kafka broker is present in cluster and later entity operator also starts running.
* While deploying kafka cluster yaml file make sure you choose load balancer as an external service to connect from outside the server.

2. **Installing Kafka on your local machine**

* Download kafka for your machine from the official kafka website. Make sure you download the stable version.
* Now unzip the downloaded file and then install it.
* After successfully installing we need to test if it works by running it locally using command line.
* Inorder to test start the zookeeper and then kafka cluster using the commands present in official kafka website.
* Now create a topic to inorder to test your producer and consumer application.
* Start producer application and then type a message to topic.
* Start consumer and make sure to mention from beginning at end of start command to consume messages from topic.

3. **Developing Springboot Producer application**

* Start a new springboot application and select Spring web service and Spring kafka service as initial dependencies.
* Now write a class (bean object) which you can pass through Rest API with all necessary variables, constructors, getters and setter methods.
* Create a configuration class in where we write a producer configuration which writes a message to a kafka topic
* This method uses key and data as two parameters. Before sending it topic we need to mention the bootstrap server location and port number and then provide serializable classes for string and json
* Then create another class which sends data to topic. Mention topic name here.
* First test locally by producing a message to kafka topic present in locally by running the springboot application.
* Now connect to kafka topic present in kubernetes using the load balancer service and port created above in step 1.

4. **Developing springboot Consumer Applciation**

* Start a new springboot application and select JSONObjectify, Spring web service and Spring kafka service as initial dependencies.
* Now write a class (bean object) which you can pass through Rest API with all necessary variables, constructors, getters and setter methods.
* Create a configuration class in where we write a consumeer configuration which consumes a message from a kafka topic.
* This method uses key and data as two parameters. Before sending it topic we need to mention the bootstrap server location and port number and then provide deserializable classes for string and json
* Then create another class which consumes data from topic. Mention topic name and group id here.
* First test locally by consuming a message from kafka topic present in locally by running the springboot application.
* Now connect to kafka topic present in kubernetes using the load balancer service and port created above in step 1.

5. **Deploying producer and consumer applications in kubernetes**

* Firsly write a docker file for producer application and then run the application using maven install.
* This will then generate a jar file and using docker create a image and push the producer application to dockerhub.
* Now write a yaml file for deploying springboot application in kubernetes and make sure you use a loadbalancer so as to connected to this application from outside kubernetes.
* Repeat the same steps for consumer application and now make sure you can produce and consume messages from a service like PostMan.

6. **Setting up Producer and Consumer Applications with Angular.**

* In the angular application we need to change the services typescript file with previous links to current producer and consumer app links using loadbalancer service.
* We need to use POST Mapping in producer app to produce message from angular application and send data to kafka topic.
* We need to use GET Mapping in consumer app to consume messages and send it to angular application.
* We have used an Arraylist in consumer app to store messages from kafka topic and return the array list of type student using get mapping to the angular application.
* Now Deploy angular app in kubernetes as we did in homework 3 using aload balancer.
* Now using loadbalancer URL and Port of angular app we can fill the student survey form and send data to kafka topic and consume it again by showing entries in list.

**Links:**

* **URL for Strimzi Kafka:** https://strimzi.io/docs/operators/latest/quickstart.html
* **URL for Springboot:** https://medium.com/@contactsunny/simple-apache-kafka-producer-and-consumer-using-spring-boot-41be672f4e2b
* **URL for Kafka yaml files:** https://github.com/strimzi/strimzi-kafka-operator/tree/master/examples

**References:**

* <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>
* <https://strimzi.io/blog/2020/04/15/develop-apache-kafka-applications-with-strimzi-and-minikube/>
* <https://www.youtube.com/watch?v=GSh9aHvdZco>
* https://www.youtube.com/watch?v=KEPB7iG5Fgc
* https://strimzi.io/docs/operators/latest/deploying.html