**Kafka - Event Driven microservice architecture**

Apache Kafka is an open-source distributed event streaming platform.

Continuous sending messages to the kafkaesque server and reading and processing of these messages is called real time event streaming.

It is a publish subscribe model

Publishers will publish the data

Subscribers will consume the data

There can be multiple publishers and multiple subscribers.

Publisher —> Messsage Broker —> Consumer

**Kafka Core API’s**

Producer API’s

Consumer API’s

Streams API’s

Connector API’s

Zookeeper will provide the base to deploy kafka . Use below command to start the application

/opt/homebrew/bin/zookeeper-server-start /opt/homebrew/etc/kafka/zookeeper.properties

**To verify installations:**

kafka-topics --version

zookeeper-server --version

**To list kafkaesque topics:**

kafka-topics --list --bootstrap-server localhost:9092

**Start zookeeper with full path:**

zookeeper-server-start /opt/homebrew/etc/kafka/zookeeper.properties

Latest command : $KAFKA\_HOME/bin/zookeeper-server-start.sh /opt/homebrew/etc/kafka/zookeeper.properties

Default port: 2181

**Start kafka with full path:**

kafka-server-start /opt/homebrew/etc/kafka/server.properties

Latest command: $KAFKA\_HOME/bin/kafka-server-start.sh /opt/homebrew/etc/kafka/server.properties

Default port : 9092

**Working with kafka through command line**

**To create a topic from command line:**

$KAFKA\_HOME/bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic manualcreatedtopic --partitions 3 --replication-factor 1

Note : There are three ways to create a topic

1. Create from above command.

2. Just give topic name in kafkatemplate.

3. Using kafka config file.

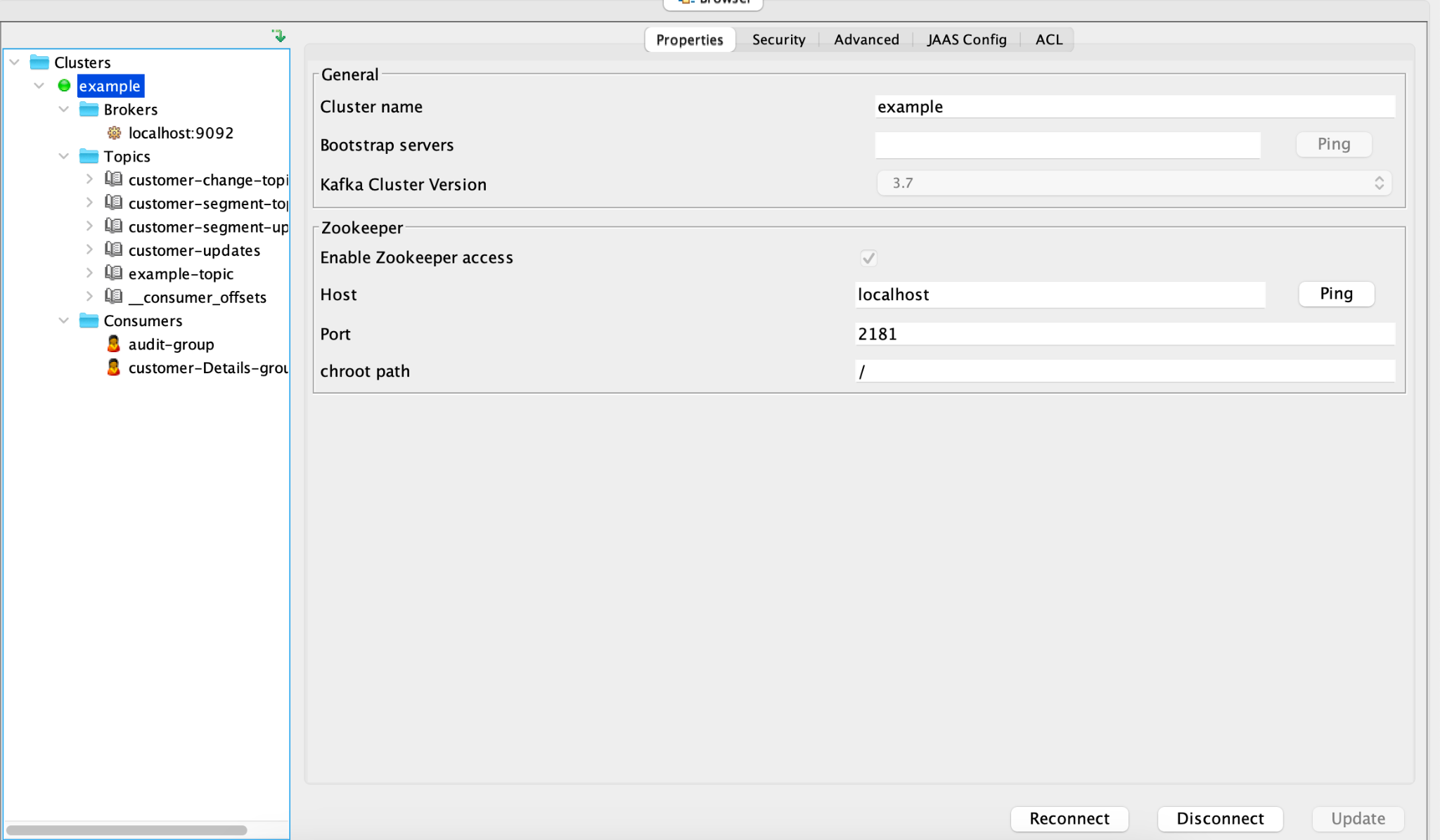
**To list Topics**

$KAFKA\_HOME/bin/kafka-topics.sh --bootstrap-server localhost:9092 —list

**To describe a particular topic**

$KAFKA\_HOME/bin/kafka-topics.sh --bootstrap-server localhost:9092 --describe --topic example-topic

**Open the offset explorer and create a connection to monitor topics and data in them:**

****

**Now start your producer app from console**

$KAFKA\_HOME/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic example-topic

**And start your consumer app from console**

$KAFKA\_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic example-topic --from-beginning

**To reset offset**

**Step1:** cd /opt/homebrew/Cellar/kafka/3.8.0/bin

**Step2:** ./kafka-consumer-groups --bootstrap-server localhost:9092 --group kafka-consumer-3 --topic customer-demo --reset-offsets --to-latest --execute

**Prompt in chatgpt**

write a simple springboot applications where one application stores the customer details like id,name and segment (where segment can be gold or silver) and other application will store the customername along with segment whenever there is a change made in first application for a particular customer same should be changed in second application using kafka publish and subscribe model microservices concept and also include one more topic in same example and some other details is changed

**Kafka Components:**

1. **Producer**
2. **Consumer**
3. **Message broker-** intermediate entity helps between producer and consumer
4. **Cluster -** It is a group of computers or servers working for a common purpose. Depending on the numbers of messages from producer.
5. **Topic -** It is inside a broker. It specifies the category of the message or the classification of the message . Listeners can then just respond to the messages that belong to the topics they are listening on.
6. **Partitions:** To handle the number of messages from producer to a particular topic.
7. **Offset:** Once a message is received by a topic and a particular partition then a number is assigned to the message to know which message is consumed . And this sequence number is called offset.
8. **Consumer groups:** Consumer instances can be made to consume different partitions to improve the performance of the consumer application and these instances can be grouped to form a consumer group.
9. **Zookeeper:** Zookeeper will act as a manager for kafkaesque cluster.Kafka uses zookeeper for coordination and to tract the status of kafkaesque cluster nodes.It also keeps track of kafkaesque topics, partitions, offsets etc.

**We can also send and receive messages to and from a particular partition**

By using below changes in producer and consumer applications:

In producer app:

kafkaTemplate.send("demo5", “2”,null,message);

Here 2 is the partition number

In consumer app:

@KafkaListener(topics = "customer-demo",groupId = "kafka-consumer-3”,

topicPartitions = {@TopicPartition(topic =“customer-demo”,partitions={"2"})}))

**Kafka Retry mechanism:**

Use two annotations on consumer end:

1. @RetryableTopic(attempts="4",backoff=@Backoff(delay=3000,multiplier=3,maxDelay=15000),exclude={NullPointerException.class,RuntimeException.class}) here 4 mean it will retry n-1 times i.e 3 times it will try to reprocess messages in a topic in case of failure. Use this annotation where you are using @Kafkalistener. If you want to do retry at some intervals instead of again and again we can use backoff in which delay , multipler,maxdelay are the options.If you want to exclude some kind of exception from retry we can use exclude option.

2. @DltHandler: DLT stands for Dead letter topic . This is used when even after kafka broker tried re processing messages for the number of times specified in @RetryableTopic annotation then it will move to DLT.

**Documentations:**

**# documents**

**## Open Source Kafka Startup in local ##**

**1. Start Zookeeper Server**

**```sh bin/zookeeper-server-start.sh config/zookeeper.properties```**

**2. Start Kafka Server / Broker**

**```sh bin/kafka-server-start.sh config/server.properties```**

**3. Create topic**

**```sh bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic NewTopic --partitions 3 --replication-factor 1```**

**4. list out all topic names**

**``` sh bin/kafka-topics.sh --bootstrap-server localhost:9092 --list ```**

**5. Describe topics**

**``` sh bin/kafka-topics.sh --bootstrap-server localhost:9092 --describe --topic NewTopic ```**

**6. Produce message**

**```sh bin/kafka-console-producer.sh --broker-list localhost:9092 --topic NewTopic```**

**7. consume message**

**``` sh bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic NewTopic --from-beginning ```**

**## Confluent Kafka Community Edition in local ##**

**1. Start Zookeeper Server**

**```bin/zookeeper-server-start etc/kafka/zookeeper.properties```**

**2. Start Kafka Server / Broker**

**```bin/kafka-server-start etc/kafka/server.properties```**

**3. Create topic**

**```bin/kafka-topics --bootstrap-server localhost:9092 --create --topic NewTopic1 --partitions 3 --replication-factor 1```**

**4. list out all topic names**

**``` bin/kafka-topics --bootstrap-server localhost:9092 --list ```**

**5. Describe topics**

**``` bin/kafka-topics --bootstrap-server localhost:9092 --describe --topic NewTopic1 ```**

**6. Produce message**

**```bin/kafka-console-producer --broker-list localhost:9092 --topic NewTopic1```**

**7. consume message**

**```bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic NewTopic1 --from-beginning ```**

**8. Send CSV File data to kafka**

**```bin/kafka-console-producer --broker-list localhost:9092 --topic NewTopic1 <bin/customers.csv```**

**https://github.com/basanta-spring-boot/documents/blob/main/README.md**