Below steps should help to setup Kafka AQ demo.

**Step 1: Setup the database user.**

In file AQSetup.sql, we are connecting using a sys privilege with the database. In our testing environment, sys/knl\_test7 is the default user/password to connect as sysdba. This should be changed accordingly.

AQSetup.sql first creates a database user named aq/aq. It also grants privileges to aq so that further aq operations can be carried out.

Next it creates createTopic, purgeQueueand addSubscriber procedures which allows aq user to create a multi consumer queue which is a Topic in Kafka/JMS world. purgeQueue allows to truncate the messages of the queue. addSubscriber procedure allows to add a subscriber for the topic. This is similar to a ‘ConsumerGroup’ name of the Kafka.

AQSetup.sql then creates a Topic named Topic2 and adds a subscriber S1.

In our demo program we are going to publish and consume messages from Topic2. For our demo consumer application we are going to use S1 as ConsumerGroup name.

To run this setup script, user needs to connect using sys privilege and provide Topic Name and Subscriber Name as command line arguments.

For this demo we are restricting our Topic2 to have only 1 partition. This is because in the database version which we are going to use for demo does not have certain AQ functionality. Also partition 1 is sufficient for the test case explained by Sai in the meeting. Later on, we can backport the full AQ functionality to this database label and explore more possibilities.

**Step 2: Compiling Producer and Consumer.**

In the zip file, there is a folder named jars. This contains all sufficient jar files to run the program. They should be included in the classpath while compiling and running.   
*Note: ojdbc8.jar file is a JDBC Driver jar. In a larger setup where multiple applications are using the common JDBC Driver then, this ojdbc8.jar should not be included in the classpath.*

In the jars directory KafkaAQ.jar file is the one which contains the methods which allows Producer and Consumer to connect to the Oracle AQ.

**Step 3: Configure Producer and Consumer:**

Producer1.java and Consumer1.java are kafka applications which produces the message and consumes them respectively. To allow these Kafka application to connect with the Oracle database, we must provide certain configuration parameter. This parameters will tell KafkaAQ.jar file where to locate the Topics. Below are the list of parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Example Value |
| bootstrap.servers | Host and Port number where database listener or CMAN is running. This is the endpoint where applications are supposed to Connect with Oracle. This should be provided in either HOSTNAME:PORT or IP:PORT format. | den02tgo:1521, den02tgo.us.oracle.com:1521 10.23.168.19:1521 localhost:1521 |
| oracle.sid | Oracle SID. From oracle installation machine, check $ORACLE\_SID environment variable. |  |
| oracle.service.name | Database Service name which application is supposed to use. | pdb1.regres.rdbms.dev.us.oracle.com $ORACLE\_SID.rdbms.dev.us.oracle.com |
| oracle.instance.name | Instance name where application is supposed to connect. You can find it from ‘select instance\_name from v$instance’ | Usually same as Oracle SID |
| oracle.usr.name | User name under which, the Topic exists. | In our setup it is “aq” |
| oracle.password | Password for above | In our setup it is “aq”. |

**Step 4: Run Kafka Producer:**

Producer1.java is the Kafka Producer application. After making above changes, compile the producer file

*javac –classpath jars/KafkaAQ.jar … Producer1.java*

This program takes one command line argument which is how many messages to send.

For demo publish 10 messages: *java –classpath jars/KafkaAQ.jar … Producer1 10*

**Step 5: Run Kafka Consumer**

Consumer1.java is the Kafka Consumer application. After making configuration changes explained in step3, compile and run the consumer. No command line arguments are needed. The program will print the consumed messages.

**Step 6: Database Queries:**

To check the available messages in the Topic2, one can query:

connect aq/aq;

select count(\*), msg\_state from aq$topic2 group by msg\_state;

Get number of enqueued and dequeued messages.

Connect / as sysdba

select queue\_name, enqueued\_msgs, dequeued\_msgs from gv$persistent\_queues;