

The verification assumes that the user is at the machine "**emsa-dev remote.westeurope.cloudapp.azure.com**".

Testing the deliverables of the challenge:

## Kafka Connect Deliverable

### Step 1:

Package the kafka connect application using the dependency management tool chosen.

### Step 2:

Paste the jar to the directory **/usr/share/java/kafka**.

### Step 3:

Issue the command:

```
service kafka-connect restart
```

The command above will update the java libraries that the connector is able to access.

### Step 4:

Install the connector using the properties provided by the challengee. (Since the machine is running kafka-connect the installation should be done using a curl).

**Note:** The machine already has the port 9999 open for tests and should be the one used in the curl of installation.

### Step 5:

Start a kafka-avro-console-consumer in order to assert that the messages are indeed sent to the kafka brokers. Change the <topic-name> in the command bellow and start the consumer.

```
kafka-avro-console-consumer --bootstrap-server $(hostname -f):9092 --consumer.config /etc/kafka/client-ssl.properties --topic <topic-name> --property schema.registry.url="http://$(hostname -f):8081"
```

**Note:** The command above assumes that the topic used to receive the messages processed by the connector is already created.

## Kafka Streams Deliverable

To test this piece of software keep the connector up the telnet connection on.

### Step 1:

Package the kafka streams application using the dependency management tool chosen.

### Step 2:

Paste the jar to the directory of your choice, the command used to start the application should also be used in this directory.

### Step 3:

Start a kafka-avro-console-consumer in order to assert that the messages are indeed processed by the stream and sent to the next kafka topic. Change the <topic-name> in the command below and start the consumer.

```
kafka-avro-console-consumer --bootstrap-server $(hostname -f):9092 --consumer.config /etc/kafka/client-ssl.properties --topic <topic-name> --property schema.registry.url="http://$(hostname -f):8081"
```

**Note:** The command above assumes that the topic used to receive the messages processed by the kafka streams is already created. You should use two kafka-consumers to assert that the messages are indeed going through the flow.

### Step 4:

Start the Kafka Stream with the following command.

```
java -jar <kafka-stream-jar-path> [<properties-file.properties>]
```

**Note:** The command above assumes that a properties file exists, if this is not the case you can omit it.

### Step 5:

Send messages through the telnet connection and assert that the functionality is up and running.