Using the Kafka Connect Connectors for IBM MQ

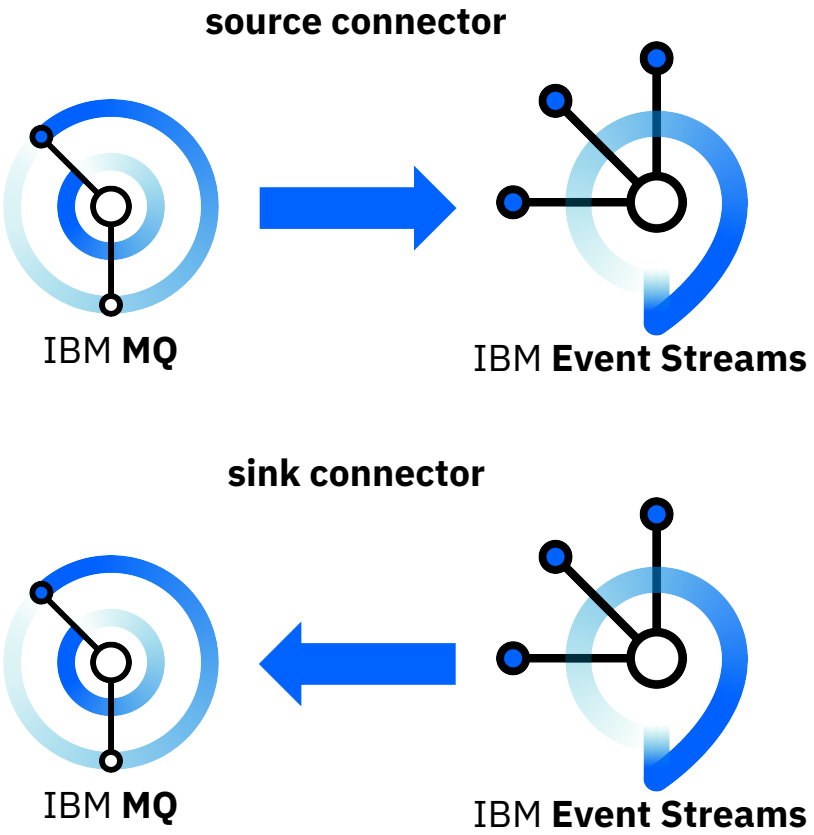
- Connect IBM Event Streams and IBM MQ by using Kafka Connect connectors for IBM MQ
- Describe what each connector does
- Configure and test each connector

What does it do?

Kafka Connect is a tool for scalably and reliably streaming data between Apache Kafka and other systems

The ***source connector for IBM MQ*** copies data from IBM MQ into Kafka (IBM Event Streams)

The ***sink connector for IBM MQ*** copies data from Kafka (IBM Event Streams) into IBM MQ



Use cases

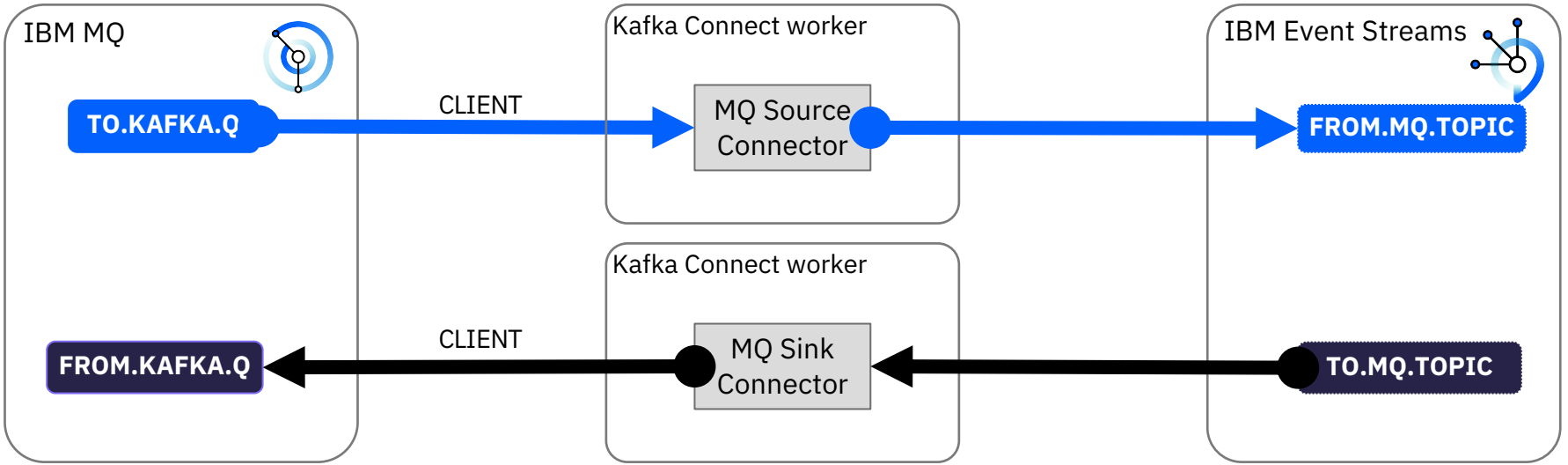
IBM MQ can be integrated with systems of record

Apache Kafka is commonly used for streaming events from web applications

The ability to connect the two systems together enables scenarios in which these two environments intersect

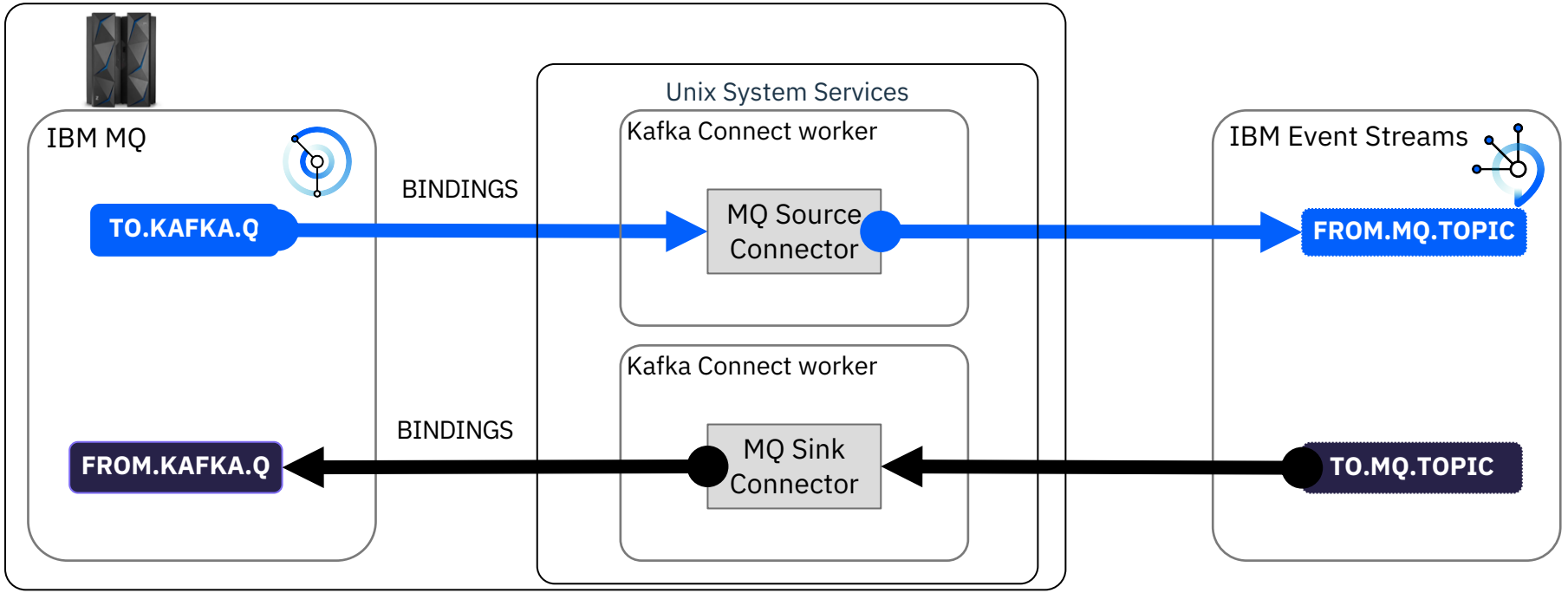
Running the Connectors for IBM MQ

The connectors are deployed into a component of Apache Kafka called a Kafka Connect ***worker.*** This runs between IBM MQ and IBM Event Streams (or open-source Apache Kafka).



Running the Connectors for IBM MQ on z/OS

The Kafka Connect workers can be deployed onto z/OS Unix System Services. Then, the connection to MQ can be a bindings connection.



Running Kafka Connect

Kafka Connect currently supports two modes of execution: ***standalone*** (single process) and ***distributed.***

In standalone mode, all work is performed in a single process:

- Simpler to setup and get started
- Useful where one worker makes sense (for example, collecting log files)
- Does not benefit from fault tolerance
- You can pass configuration parameters on the command line

Distributed mode is more suitable for production systems.

In distributed mode, the connector configurations are not passed on the command line. Instead, use the REST API to create, modify, and destroy connectors.

Publish Events from Anywhere with the REST Producer API

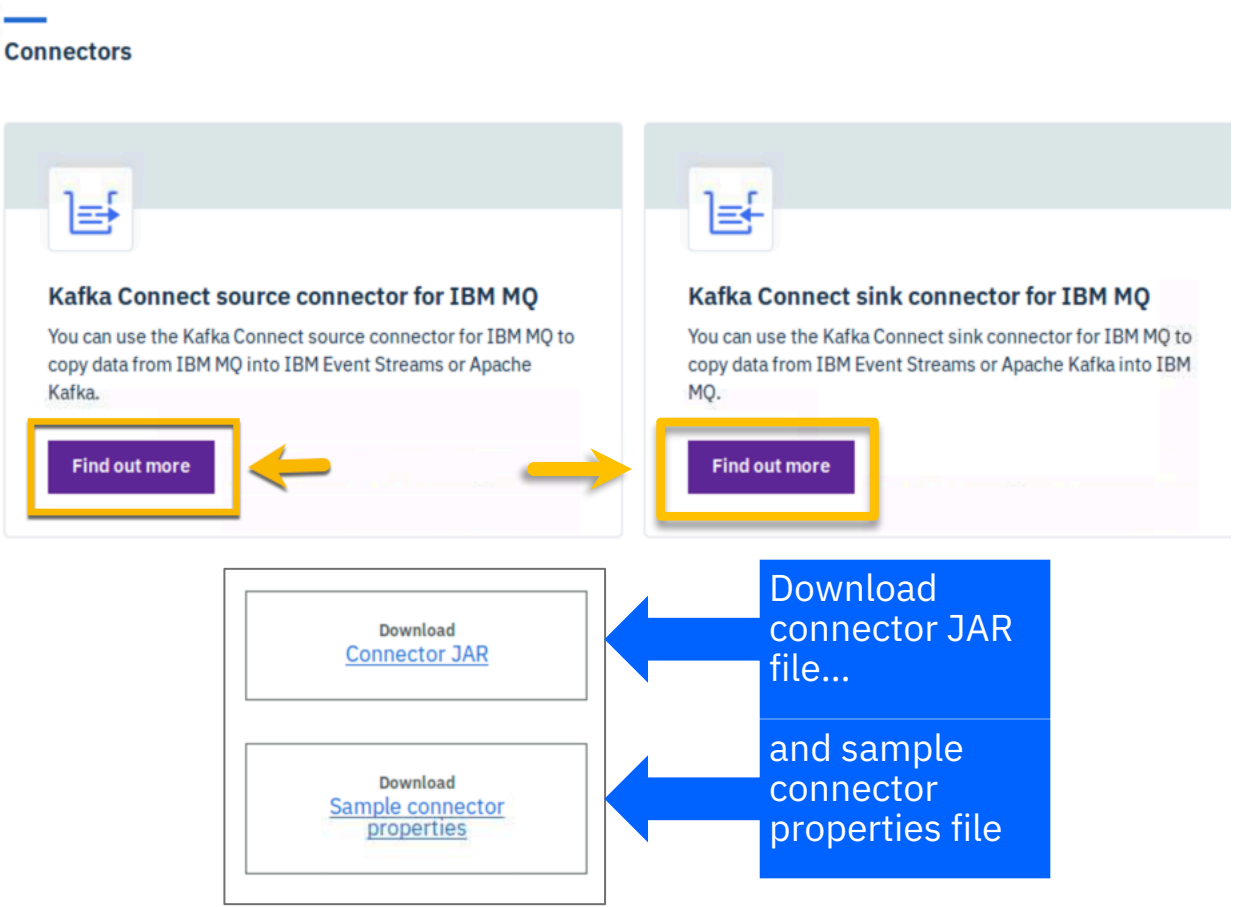
```
POST /topics/{topic_name}/records
Content-Type: text/plain
Authorization: Bearer {bearer_token}
Hello Event Streams
```

- Use it where ever it is difficult to use a real Kafka client (for example, DataPower, z/OS)
- Straightforward design makes it easy to use from the command line and developer tools
- Supports partitioning keys and headers
- You can use it from the command line with cURL

Obtaining the connectors

You can get the source code from a GitHub repository, and build the connector JAR file, or you can download a connector JAR file from the Event Streams console: **Toolbox > Connectors**

You also need a connector properties file to configure the connector



Source connector

Configuring the source connector

You need two configuration files:

- One for the information that applies to all of the connectors, (for example, Kafka bootstrap servers). The Kafka distribution includes a file called `connect-standalone.properties` that you can use as a starting point.
- One for information that is specific to the MQ source connector, such as the connection information for your queue manager. This is the file (`mq-source.properties`) that you download from GitHub or the Event Streams console.

Connector properties file

For a client connection, you must provide:

- Queue manager
- Connection name
- Channel name
- Queue name
- User name and password, if required
- Target topic

```
# The name of the MQ queue manager - required
mq.queue.manager=QM1

# The connection mode to connect to MQ - client (default) or bindings - optional
# mq.connection.mode=client
# mq.connection.mode=bindings

# A list of one or more host(port) entries for connecting to the queue manager. Entries
separated with a comma - required (unless using bindings or CCDT)
mq.connection.name.list=10.0.0.1(30753)

# The name of the server-connection channel - required (unless using bindings or CCDT)
mq.channel.name=DEV.APP.SVRCONN

# The name of the source MQ queue - required
mq.queue=DEV.QUEUE.1

# The name of the target Kafka topic - required
topic=eslab
```

Kafka Connect standalone properties file

For Event Streams, you must provide:

- The broker URL for your cluster as the bootstrap server address
- Security credentials such as SSL truststore information, and API keys

```
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# These are defaults. This file just demonstrates how to override some settings.
bootstrap.servers=10.0.0.5:32307

security.protocol=SASL_SSL
ssl.protocol=TLSv1.2
ssl.endpoint.identification.algorithm=
ssl.truststore.location=/home/student/Downloads/es-cert.jks
ssl.truststore.password=password
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token"
password="g45[redacted]8VJ";

producer.security.protocol=SASL_SSL
producer.ssl.protocol=TLSv1.2
producer.ssl.endpoint.identification.algorithm=
producer.ssl.truststore.location=/home/student/Downloads/es-cert.jks
producer.ssl.truststore.password=password
producer.sasl.mechanism=PLAIN
producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required
username="token" password="g45[redacted]8VJ";

# The converters specify the format of data in Kafka and how to translate it into Connect data. Every
Connect user will
```

Producer API key

Running the connector

You can run the source connector by connecting to a local installation of Apache Kafka, or by connecting to IBM Event Streams

Requirements:

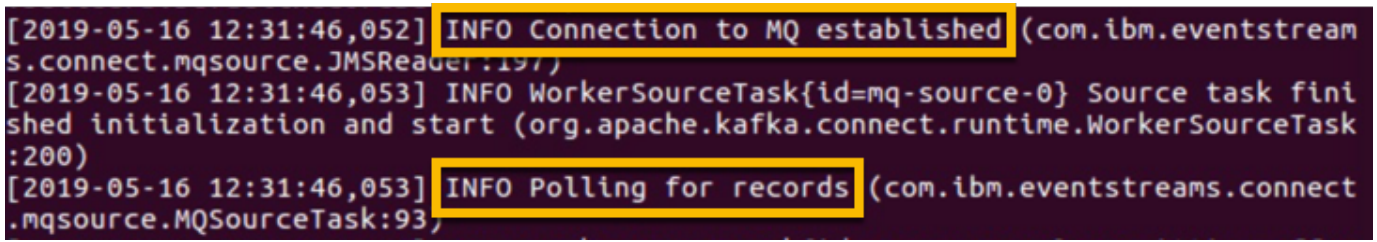
- The connector JAR file
- The connector properties file
- Apache Kafka 2.0.0 or later, either standalone or included as part of an offering such as IBM Event Streams
- IBM MQ v8 or later, or the IBM MQ on Cloud service

The connector can be run in a Kafka Connect worker in either standalone (single process) or distributed mode

Standalone command example

Run in the Kafka root directory:

```
CLASSPATH=/<location of connector jar file>/kafka-connect-
mq-source-1.0.1-jar-with-dependencies.jar bin/connect-
standalone.sh config/connect-standalone-es.properties
/<location of properties file>/mq-source.properties
```

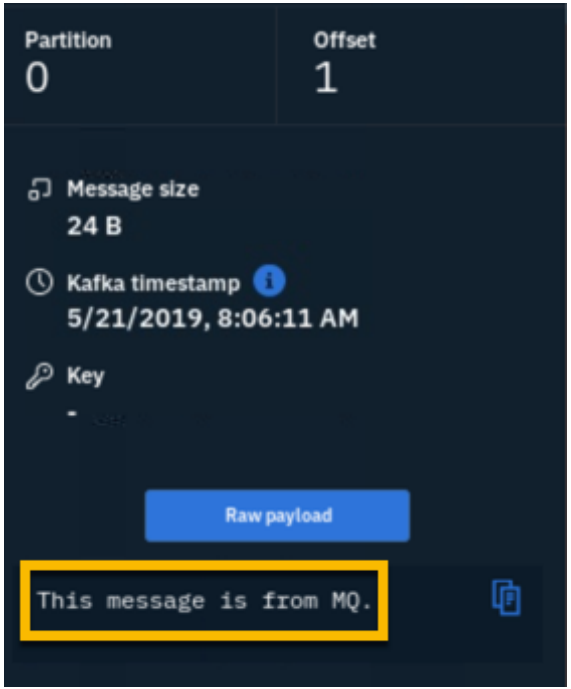


Testing the source connector

You can test the connector by using the Event Streams starter application (available from the Event Streams console) to consume messages

Put a message in the MQ queue, and then check the application

You can also test the connector by using the Kafka console consumer



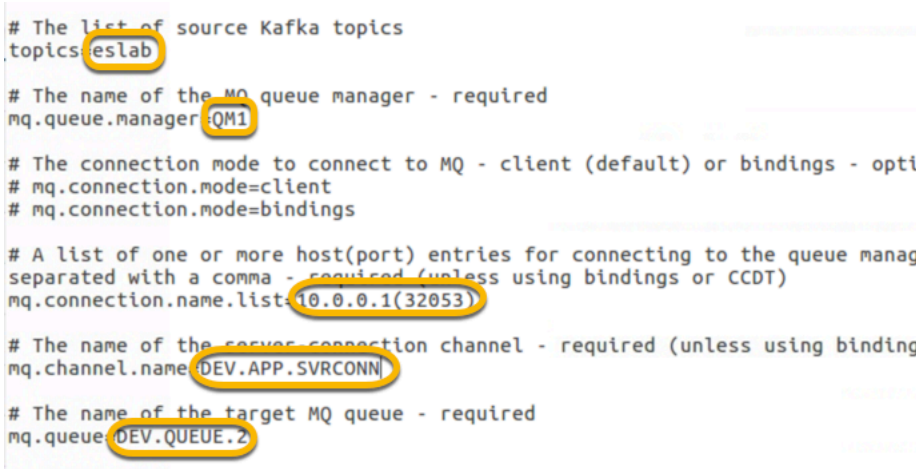
Sink connector

Configuring the sink connector

Similar to the source connector

You must update `mq-sink.properties` file with the appropriate information:

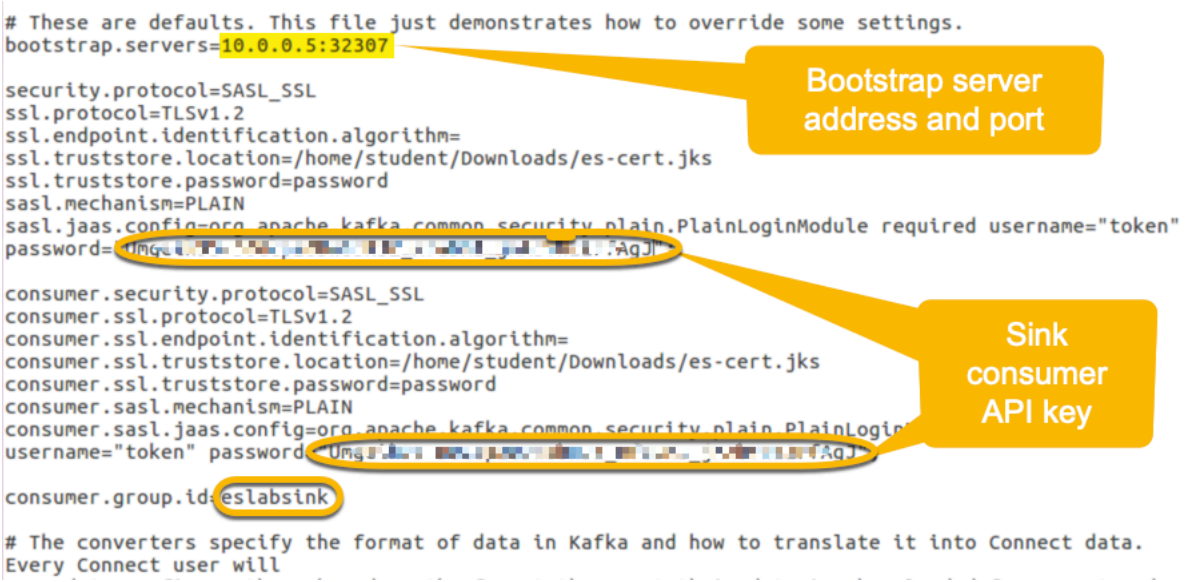
- Comma-separated list of Kafka topics to pull events from
- Queue manager
- Connection name
- Channel name
- Queue name
- User name and password, if required



Kafka Connect standalone properties

For Event Streams, you must provide:

- The broker URL for your cluster as the bootstrap server address
- Security credentials such as SSL truststore information, and API keys
- A consumer group ID




```
# These are defaults. This file just demonstrates how to override some settings.
bootstrap.servers=10.0.0.5:32307

security.protocol=SASL_SSL
ssl.protocol=TLSv1.2
ssl.endpoint.identification.algorithm=
ssl.truststore.location=/home/student/Downloads/es-cert.jks
ssl.truststore.password=password
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="token"
password=

consumer.security.protocol=SASL_SSL
consumer.ssl.protocol=TLSv1.2
consumer.ssl.endpoint.identification.algorithm=
consumer.ssl.truststore.location=/home/student/Downloads/es-cert.jks
consumer.ssl.truststore.password=password
consumer.sasl.mechanism=PLAIN
consumer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLogin
username="token" password=

consumer.group.id=eslabsink

# The converters specify the format of data in Kafka and how to translate it into Connect data.
Every Connect user will
```

Standalone command example

Run in the Kafka root directory:

```
CLASSPATH=/<location of connector jar file>/kafka-connect-mq-sink-1.0.1-jar-with-dependencies.jar bin/connect-standalone.sh config/connect-standalone-sink.properties /<location of properties file>/mq-sink.properties
```

Testing the sink connector

You can test the connector by using the Event Streams starter application (available from the Event Streams console) to produce messages

Click the arrow to start producing messages