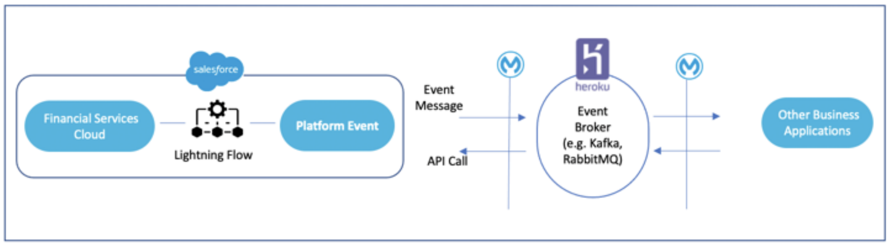
Integrating Salesforce (Platform Events/Lightning Flow) with Kafka and MySQL: Hands-on Workshop

Demo Scenario



- Account Address Change in Salesforce Financial Services Cloud triggers a Lightning Flow which generates a Platform Event
- A Mule Listener listens to the Platform Event and publishes it to Kafka on Heroku (after optionally transforming the Platform Event payload)
- Another Mule Listener consumes message from Kafka and routes it to a database connector which inserts the new address in MySQL database on Heroku.

Steps

PREREQUISITE

We assume that you have completed the FinsCabin Event-Driven Architecture workshop (Salesforce Event-Driven Architecture Hands-on Workshop: Student Workbook) and now have the demo environment and also now you are familiar with the various concepts (e.g. Mule Flows, Connectors & Transformations, Salesforce Platform Events, Lightning Flow & Heroku).

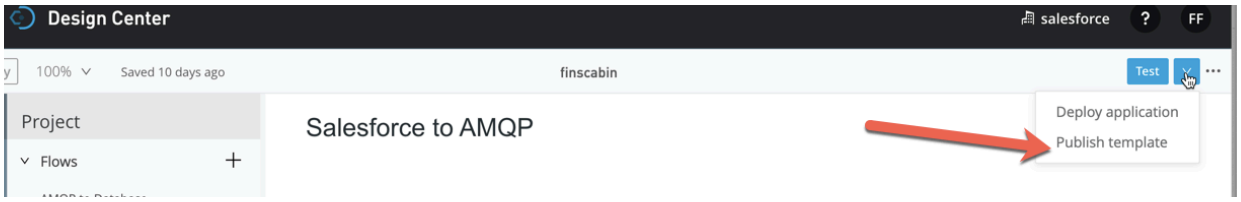
This demo assumes also assumes a basic working knowledge of Heroku CLI.

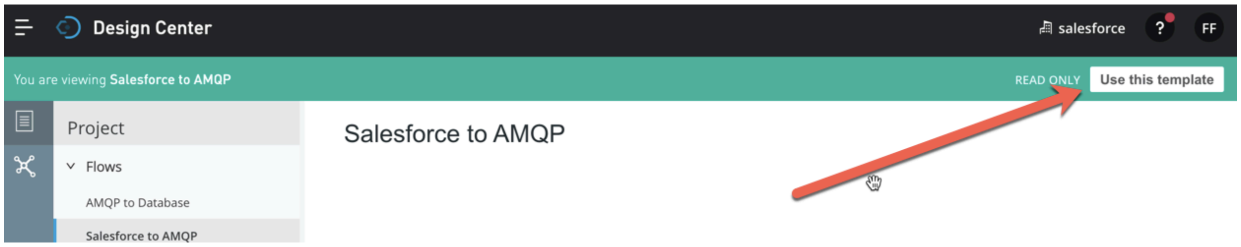
Configure a new Mule Flow
Download this file and import it in Cloud Flow Designer: https://github.com/lightningexperience/finscabin/blob/main/resources/kafka/sf-events-kafka-db-1.0.0-mule-application-template.jar

Please note that the following steps (in this sub-section - "Configure a Mule Flow") are for your reference only. You should ideally download the above jar file as it contains Kafka-specific Mule DataWeave transformations to make it work with your existing Salesforce Account Address Change flow and the Heroku FinsCabin Heroku app.

Here is a brief explanation of how I created a new Mule flow for Kafka from the existing Event-Driven Architecture Workshop flow which I had originally used for RabbitMQ (instead of Kafka).

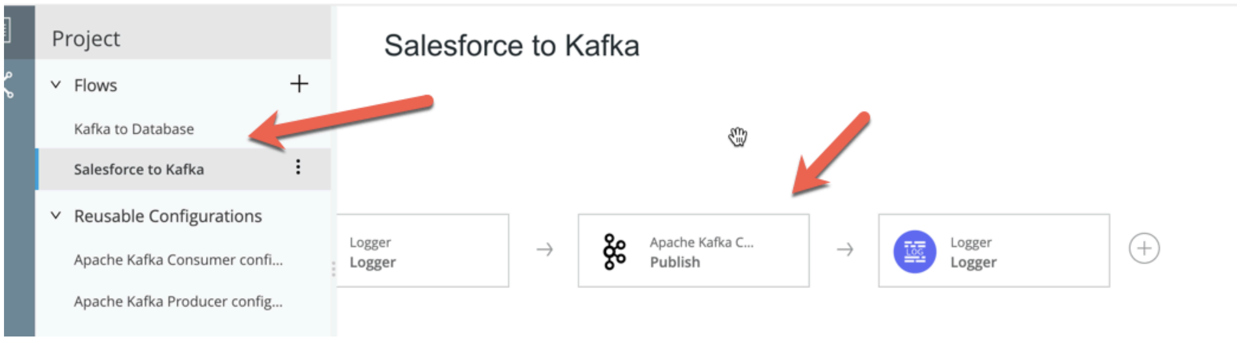
Clone your existing Mule demo flow which connects Salesforce to Database. For this, you can save your existing demo as a template and then use this template to create a new Mule flow.



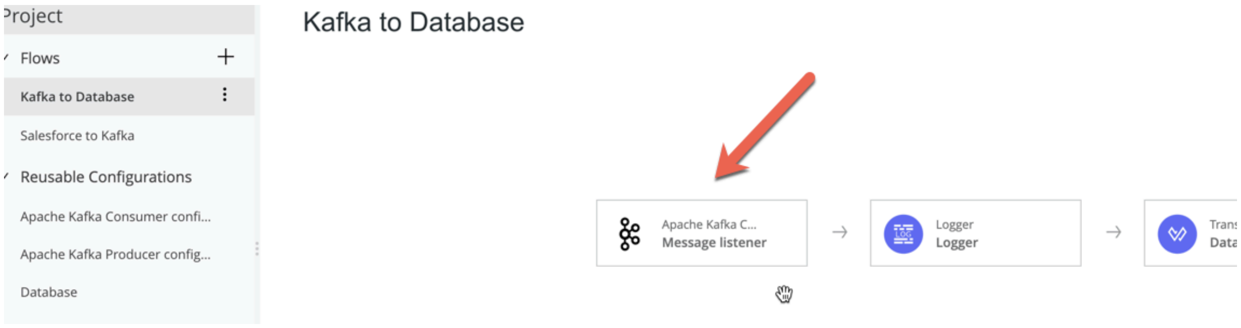


Now in the new Mule flow, that you have created from your template - delete AMQP connectors. You can optionally rename the sub-flows in the Cloud Flow designer(e.g instead of Salesforce to AMQP you can rename it as Salesforce to Kafka

In the Cloud Flow Designer insert the Kafka Publish Connector where you had AMQP Publish Connecter.



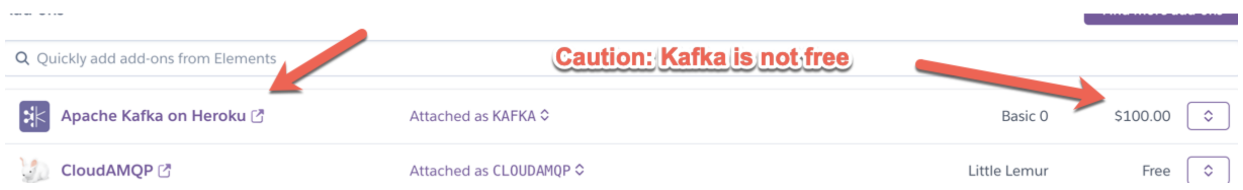
In the AMQP to Database subflow - remove AQMP and insert Kafka Listener



We will now need to connect these Mule Connectors to Kafka - let’s first install Kafka on Heroku.

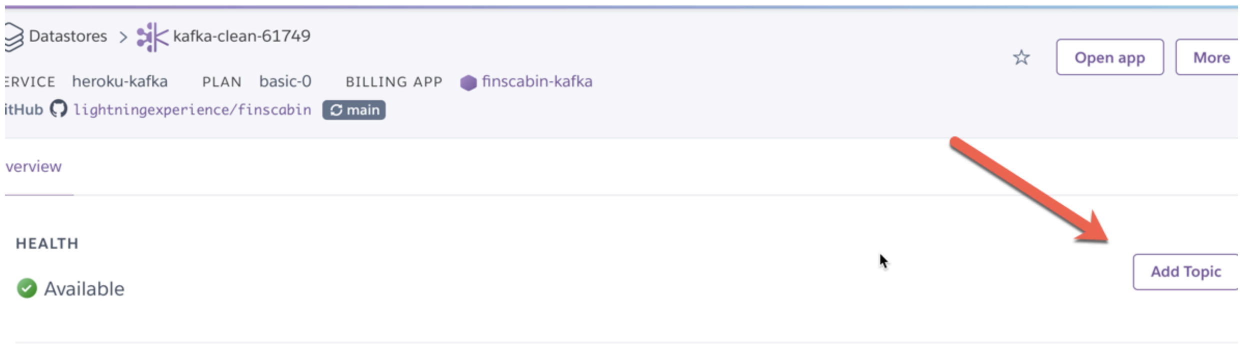
INSTALL & CONFIGURE KAFKA ON HEROKU

Go to your Heroku instance and provision Kafka in your existing FinsCabin app. Be aware that Kafka does not have a free tier on Heroku - please check with your manager before installing.



Apache Kafka on Heroku	Attached as KAFKA	Basic 0	$100.00
CloudAMQP	Attached as CLOUDAMQP	Little Lemur	Free

Now create a topic in Heroku (you can do it through the Heroku GUI). Let's call it eda01. It will also ask you for number of partitions (Just select 1).


Datastores > kafka-clean-61749
ERVICE heroku-kafka PLAN basic-0 BILLING APP finscabin-kafka
itHub lightningexperience/finscabin main
Open app
More
verview
HEALTH
Available
Add Topic

The topic needs nearly 30 minutes to get created and become visible in Heroku CLI (You won't see topics listed in the dashboard).

Install Heroku CLI - You would need it in this demo: https://devcenter.heroku.com/categories/command-line

Login to Heroku via CLI
heroku login

Install Kafka Plugin for Heroku CLI
heroku plugins:install heroku-kafka

Create a Consumer Group in your Kafka instance.
heroku kafka:consumer-groups:create consumer01 -a finscabin-kafka

Now you can view your Kafka instance from CLI
heroku kafka:info -a finscabin-kafka

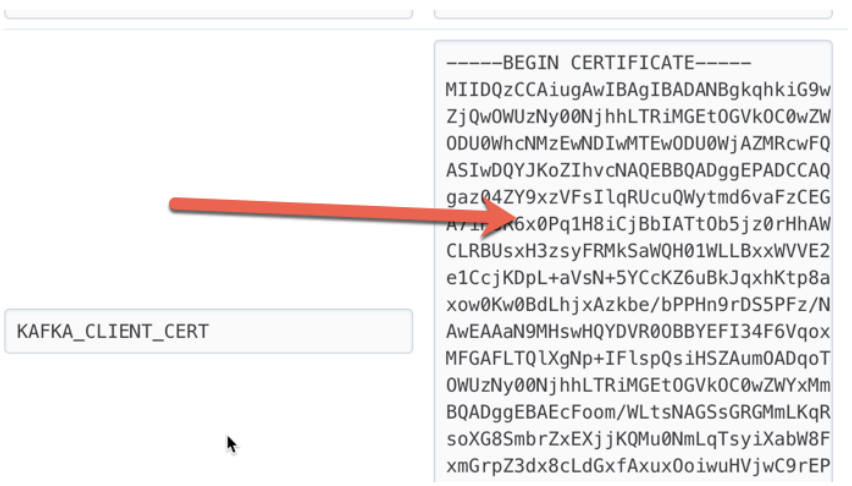
CONNECT TO HEROKU KAFKA FROM MULE CONNECTOR

We are making this assumption that your environment is in Heroku multi-tenant Common Runtime (not Private Spaces). In Private Spaces you have some more options for connecting including direct access to Zookeeper. Let's keep things simple and assume you have the simplest Heroku configuration (default in Personal/demo/trial accounts).

Heroku provides the certificates to connect to Kafka in PEM format. The MuleSoft Kafka Connector requires the certificates to be in JKS format. Therefore, we need to convert the PEM certificates and keys to a PKCS12 file first before we convert them to JKS.

Go to your Heroku app -> Settings and view the config variables.

You will see certificates as in the following screenshot.



KAFKA_CLIENT_CERT

```
-----BEGIN CERTIFICATE-----
MIIDQzCCAiugAwIBAgIBADANBgkqhkiG9w
ZjQwOWUzNy00NjhhLTRiMGEtOGVkOC0wZW
ODU0WhcNMzEwNDIwMTEwODU0WjAZMRcwFQ
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQ
gaz04ZY9xzVFsIlqRUcuQWytmd6vaFzCEG
A71[illegible]6x0Pq1H8iCjBbIATtOb5jz0rHhAW
CLRBUsxH3zsyFRMkSaWQH01WLLBxxWVVE2
e1CcjKDpL+aVsN+5YCcKZ6uBkJqxhKtp8a
xow0Kw0BdLhjxAzkbe/bPPHn9rDS5PFz/N
AwEAAaN9MHswHQYDVR0OBBYEFI34F6Vqox
MFGAFLTQlXgNp+IFlspQsiHSZAumOADqoT
OWUzNy00NjhhLTRiMGEtOGVkOC0wZWYxMm
BQADggEBAEcFoom/WLtsNAGSsGRGMmLKqR
soXG8SmbrZxEXjjKQMu0NmLqTsyiXabW8F
xmGrpZ3dx8cLdGxfAxuxOoiwuHVjwC9rEP
```

Copy and paste them in individual files using a text editor choosing your respective file names (e.g. cert.pem) as I have done below.

- KAFKA_CLIENT_CERT –> cert.pem
- KAFKA_CLIENT_CERT_KEY –> key.pem
- KAFKA_TRUSTED_CERT –> trusted_cert.pem

Save all the three files in one directory and then navigate to that directory from command line.

Run the following command to generate a **pkcs12** file from the **cert.pem** and **key.pem** file:
openssl pkcs12 -export -out cert.pkcs12 -in cert.pem -inkey key.pem

The command will ask you for an export password. Don't forget to write that down. You'll need it for the next step as well as for the configuration of the Kafka Connector.

Once that's done, run the following tool to convert the **pkcs12** file to a JKS.

keytool -importkeystore -srckeystore cert.pkcs12 -srcstoretype pkcs12 -destkeystore keystore.jks -deststoretype jks -deststorepass mypassword

Make a note of all the passwords you use in the above commands - you will need it in the Mule Kafka Connector configuration.

Create Truststore

1 - Truststores require the certificate to be in binary format. The current format exported from Heroku Kafka is PEM format. Run the following command to convert the PEM certificate to binary format (DER). This creates the binary file with the name cert.der

```

```

```
openssl x509 -in trusted_cert.pem -out cert.der -outform der
```

2 - Import the binary certificate into a new jks truststore called truststore.jks by running the following. When prompted, type yes.

```
keytool -importcert -file cert.der -keystore truststore.jks  -storepass <your-key>
```

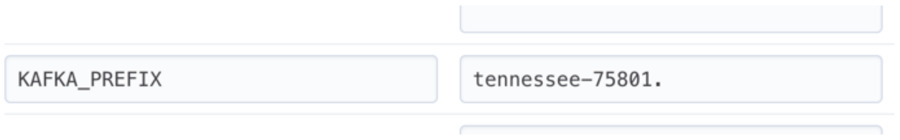
Output
Only the last 2 lines of the output are shown for brevity.

```
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

The keystore.jks and the truststore.jks files are what we need for the next section. In my connector configuration I am using heroku01.jks as the keystore file name and truststore01.jks as the trustore file name.

CONFIGURE MULE CONNECTORS TO CONNECT TO KAFKA

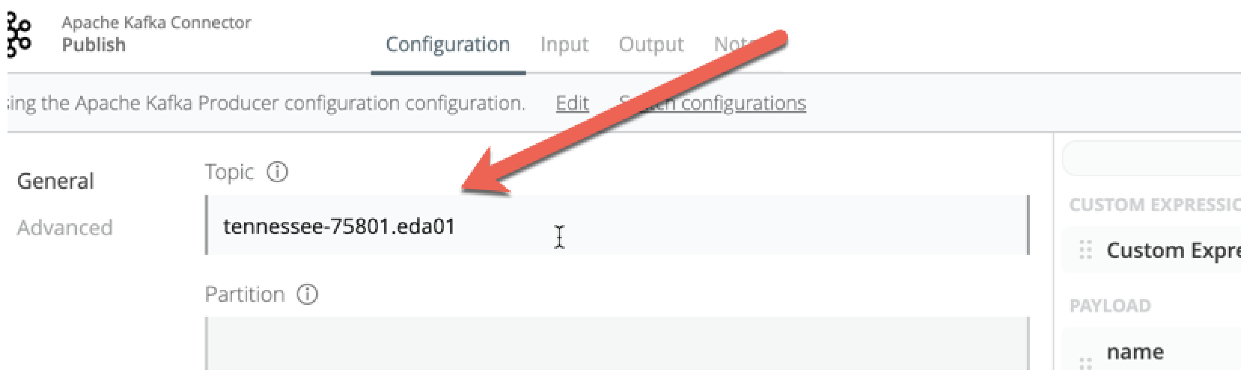
In the Kafka installed in the Common Runtime (multi-tenant) Heroku you need to prefix the topic and consumer name with the Kafka Prefix environment variable which you will find in your Heroku app's setting.



KAFKA_PREFIX	tennessee-75801.

When putting in the name of topic in your Mule Kafka connector, you will include the suffix - e.g. tennessee-75801.eda01 instead of simply eda01. This also goes for the consumer group (e.g. tennessee-75801.consumer01 instead of simply consumer01).

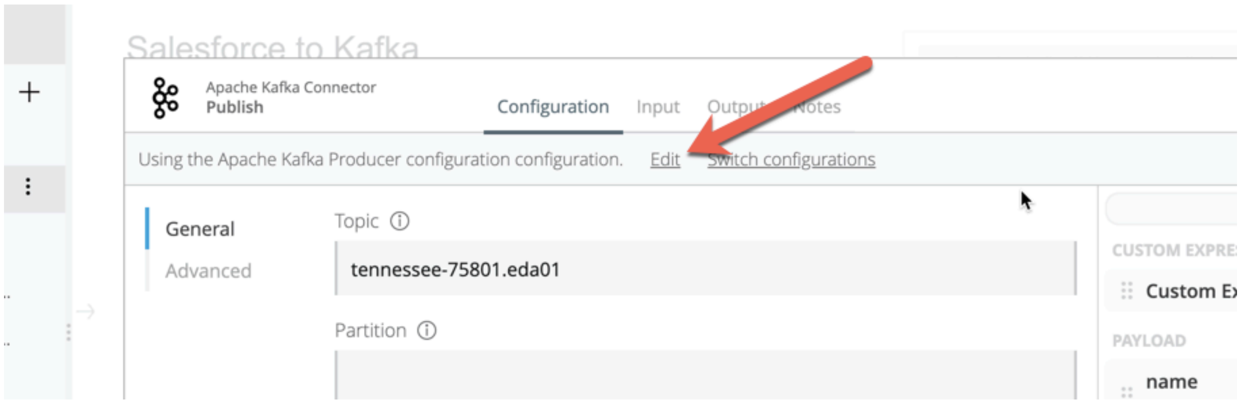
Open the Kafka Publish Connector and put in the value as shown below.



Topic: tennessee-75801.eda01

Partition:

Now edit to add a connection.

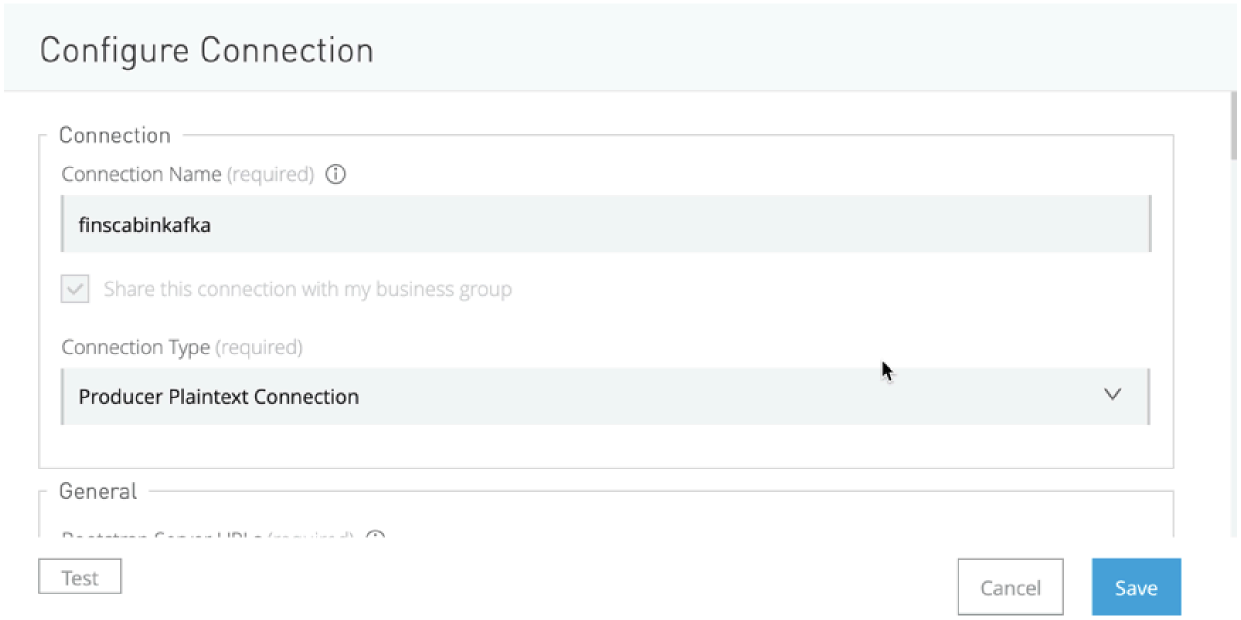


Using the Apache Kafka Producer configuration configuration.

Topic: tennessee-75801.eda01

Partition:

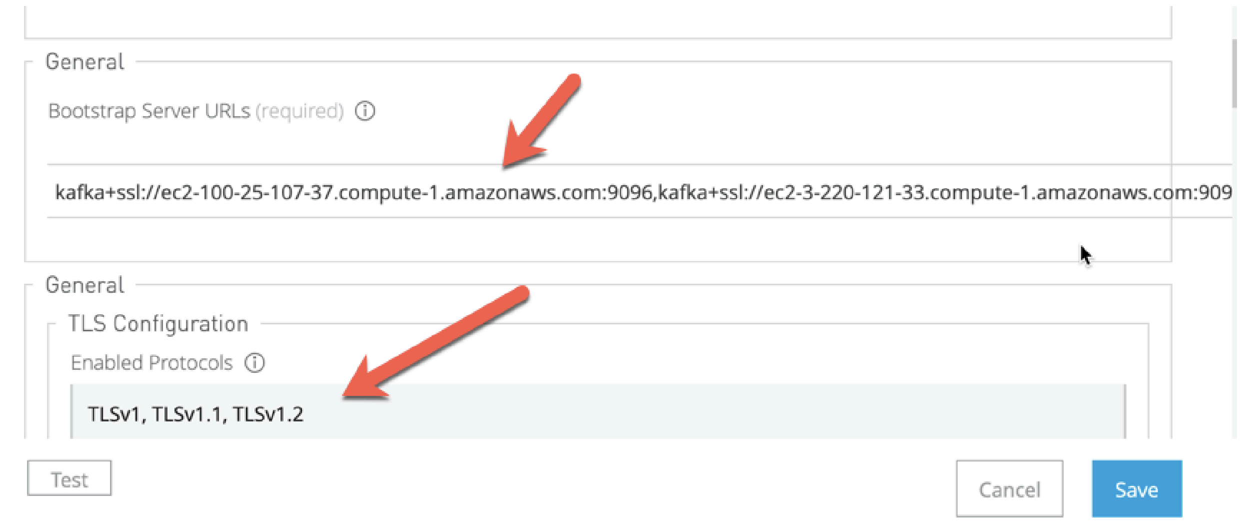
Copy the Heroku "KAFKA_URL" from your apps' setting page and put it in the Bootstrap Server URL field as shown below. Fill in other details as below.

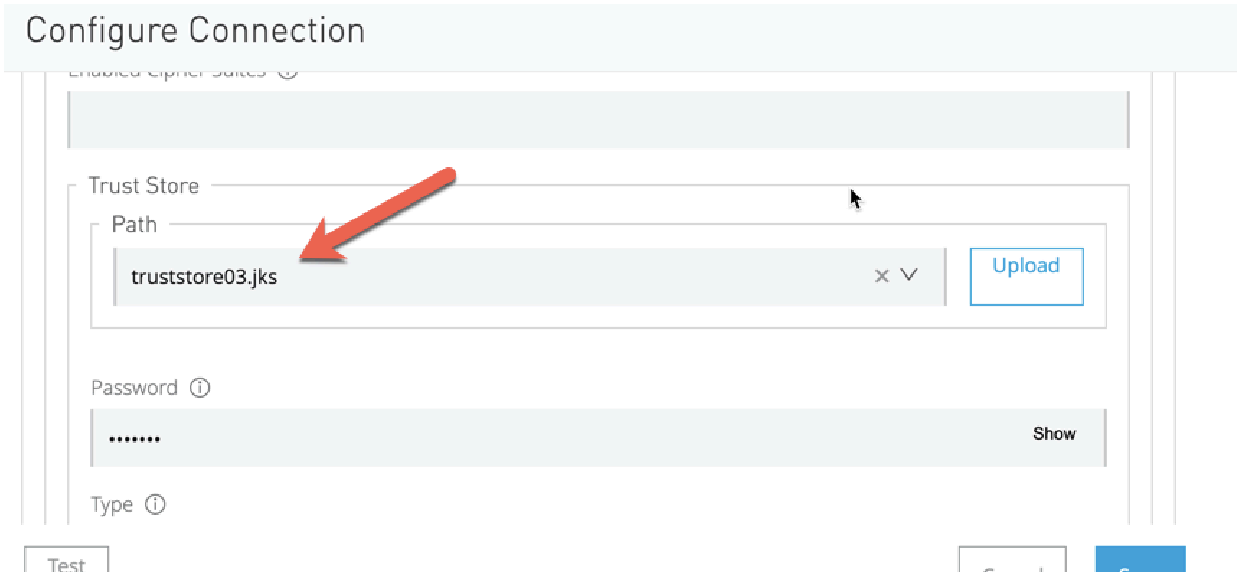


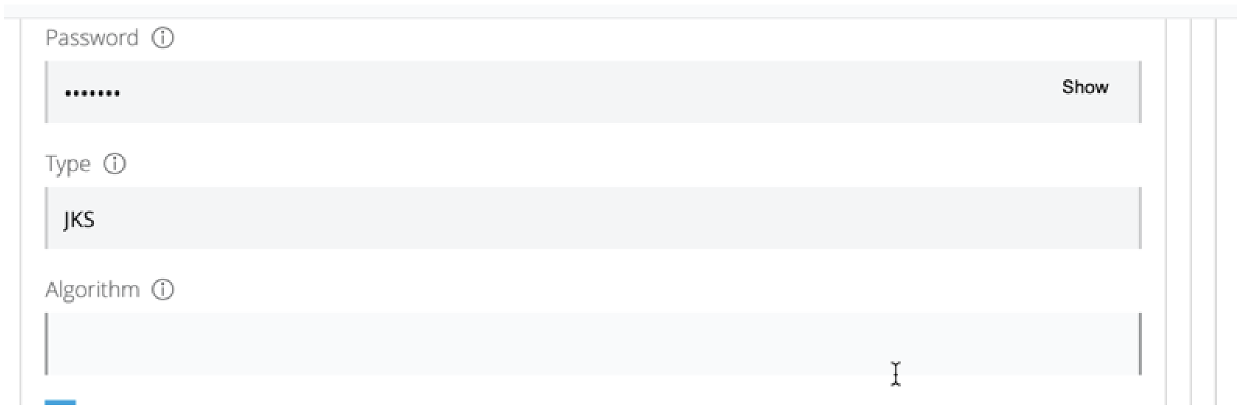
Connection Name (required): finscabinkafka

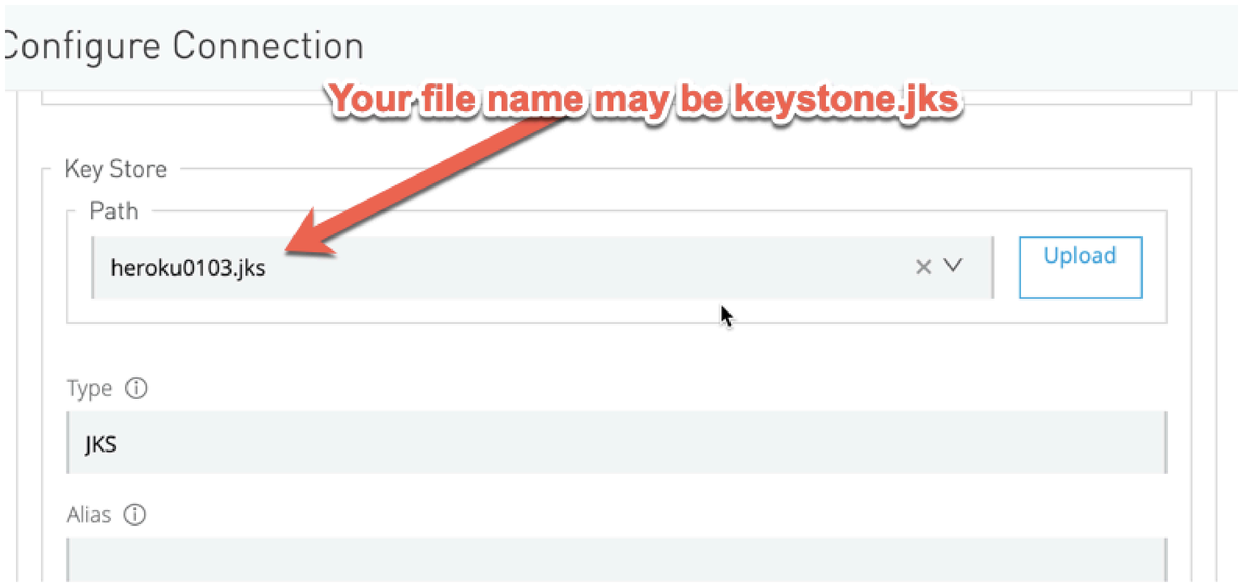
Share this connection with my business group

Connection Type (required): Producer Plaintext Connection


General
Bootstrap Server URLs (required)
kafka+ssl://ec2-100-25-107-37.compute-1.amazonaws.com:9096,kafka+ssl://ec2-3-220-121-33.compute-1.amazonaws.com:909
General
TLS Configuration
Enabled Protocols
TLSv1, TLSv1.1, TLSv1.2


Configure Connection
Trust Store
Path
truststore03.jks
Password
•••••••


Password
•••••••
Type
JKS
Algorithm


Configure Connection

Your file name may be keystone.jks

Key Store

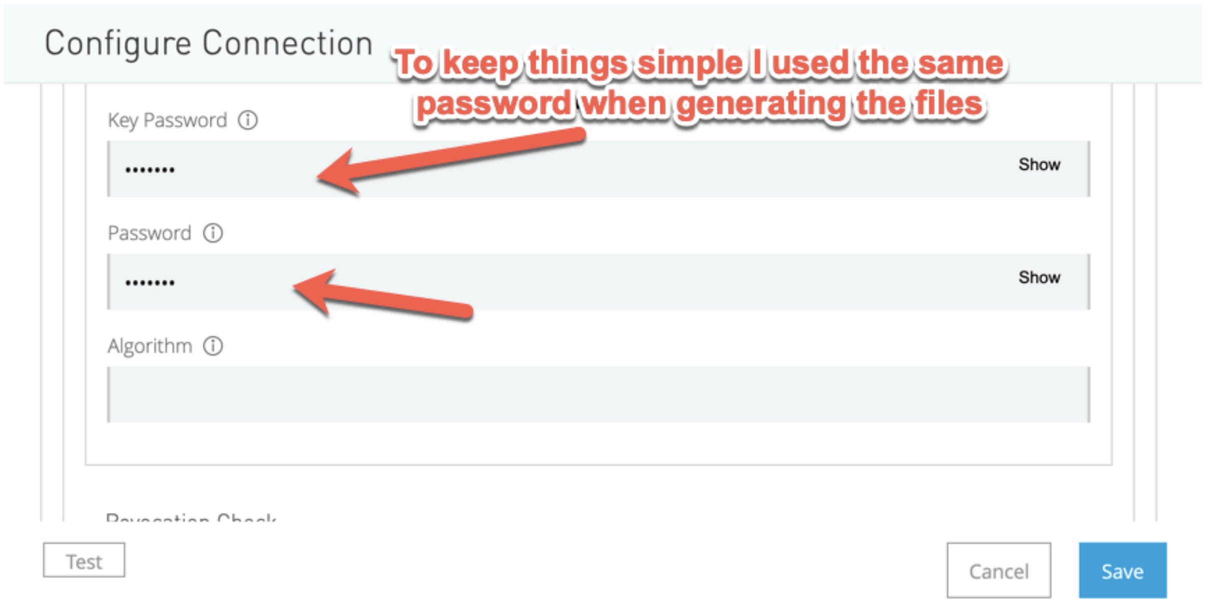
Path

heroku0103.jks

Type

JKS

Alias


Configure Connection

To keep things simple I used the same password when generating the files

Key Password

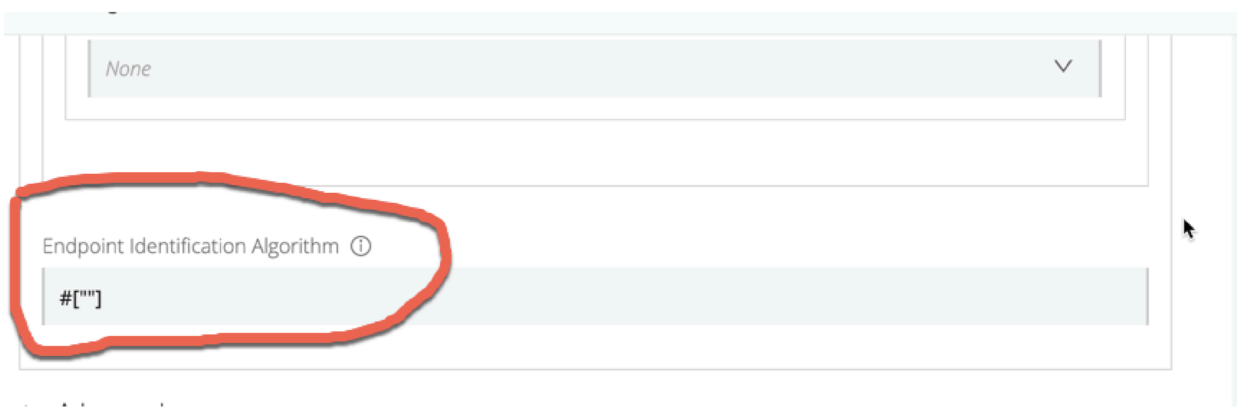
•••••••

Password

•••••••

Algorithm

For the endpoint identification algorithm I using space (i.e. double quotes with nothing between them as in the following screenshot).

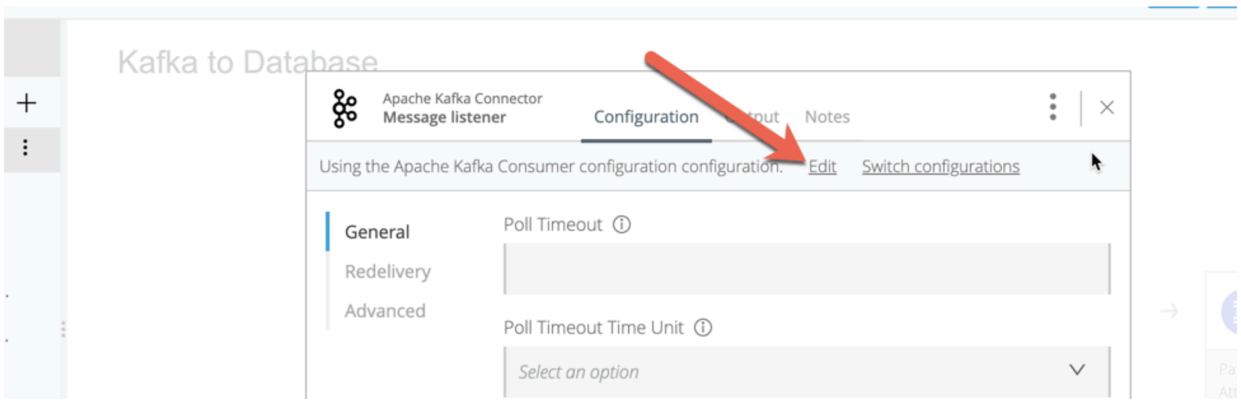

None

Endpoint Identification Algorithm

#[""]

Now test (you should get success). Save and again Save.

Now let’s configure the other sub-flow - the Kafka Consumer Connector’s connection information to Kafka.

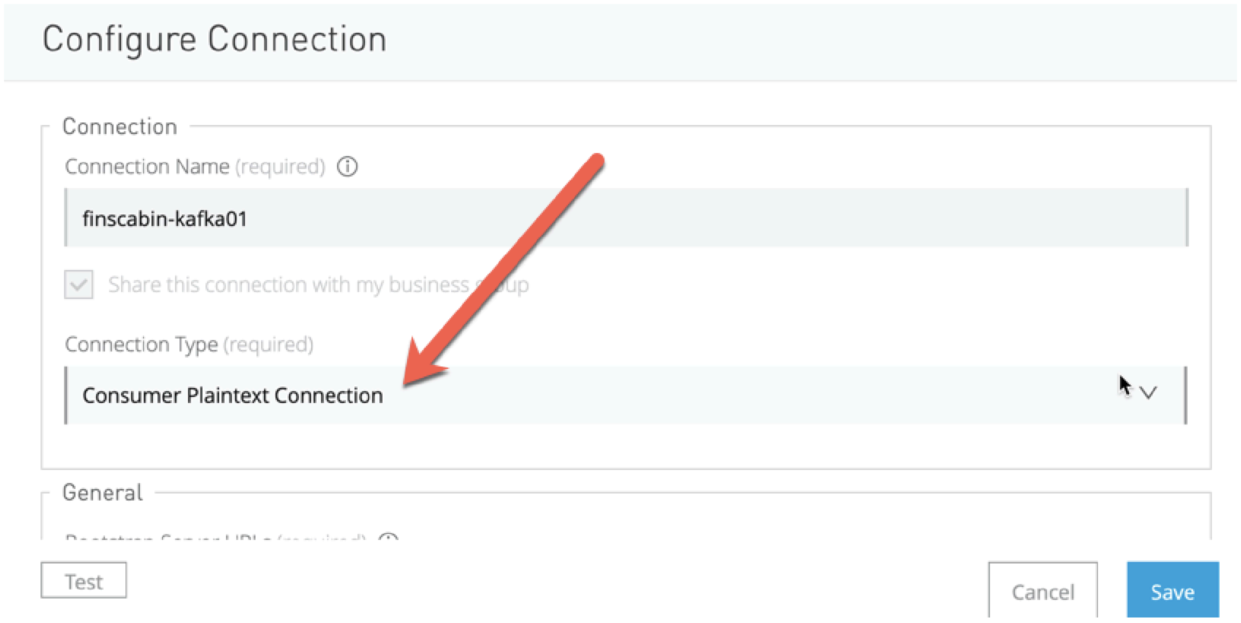


Using the Apache Kafka Consumer configuration configuration. Edit Switch configurations

Poll Timeout

Poll Timeout Time Unit

Select an option


Configure Connection

Connection

Connection Name (required)

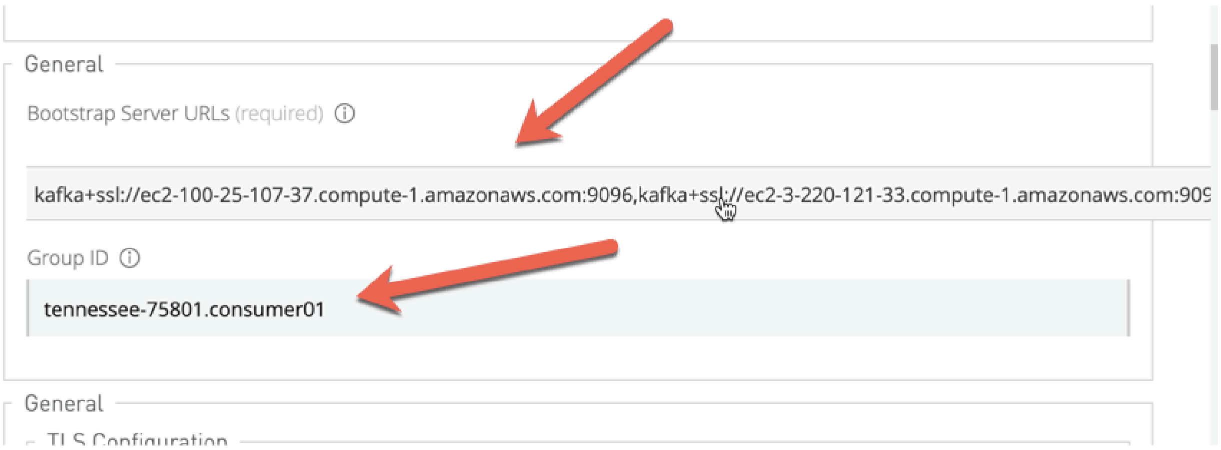
finscabin-kafka01

Share this connection with my business group

Connection Type (required)

Consumer Plaintext Connection

General



General

Bootstrap Server URLs (required)

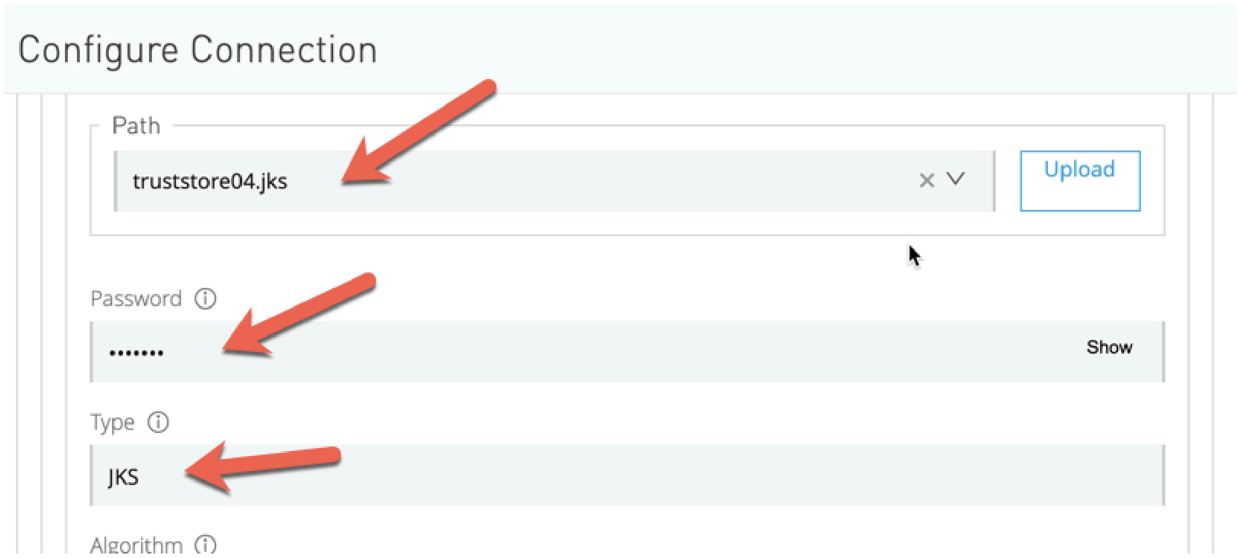
kafka+ssl://ec2-100-25-107-37.compute-1.amazonaws.com:9096,kafka+ssl://ec2-3-220-121-33.compute-1.amazonaws.com:909

Group ID

tennessee-75801.consumer01

General

TLS Configuration



Configure Connection

Path

truststore04.jks

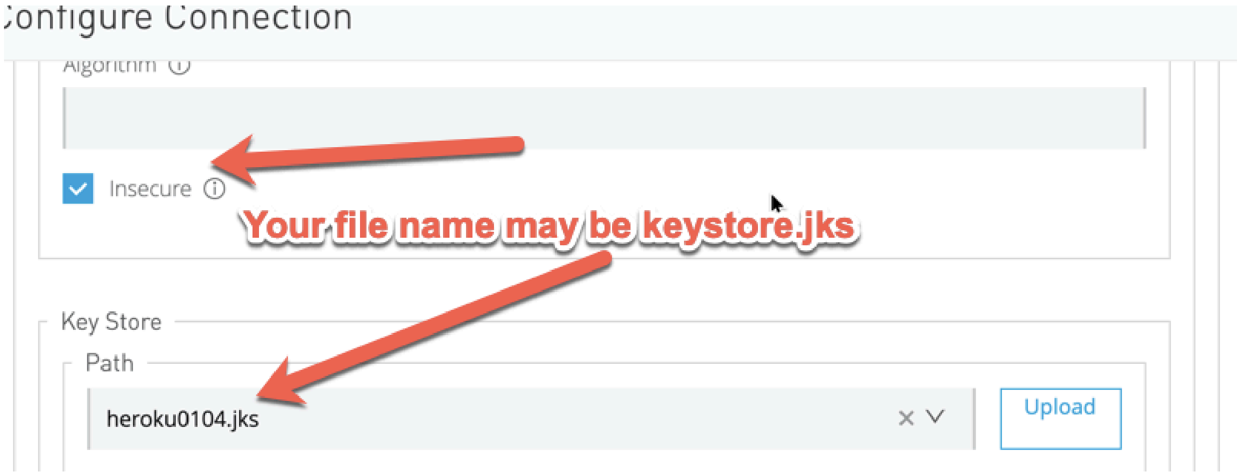
Password

•••••••

Type

JKS

Algorithm



Configure Connection

Algorithm

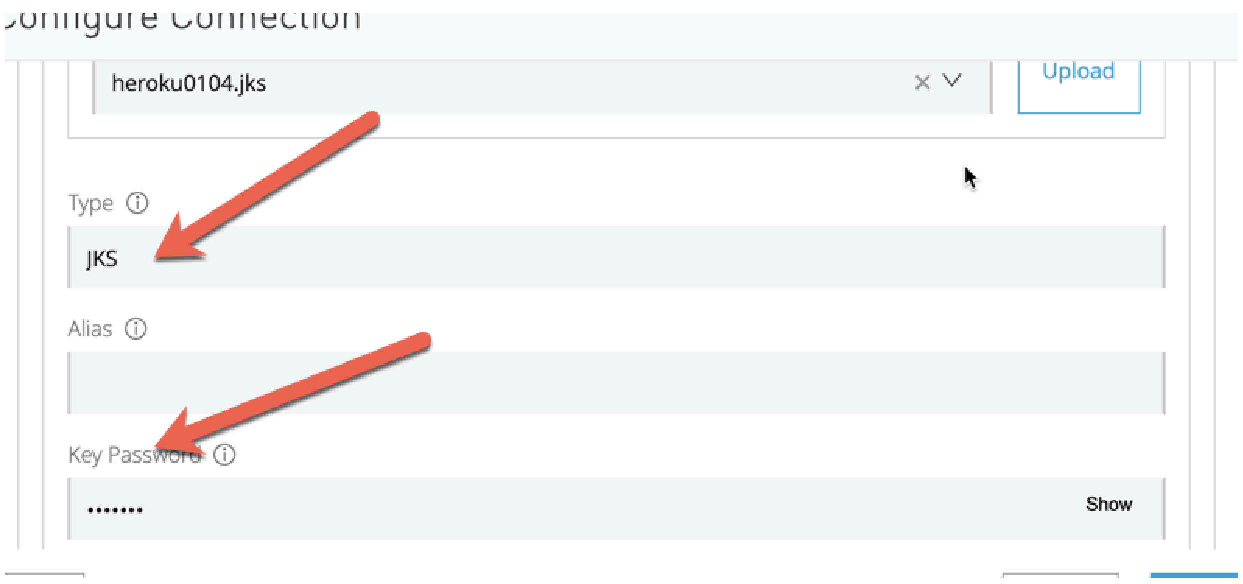
✓ Insecure

Your file name may be keystore.jks

Key Store

Path

heroku0104.jks

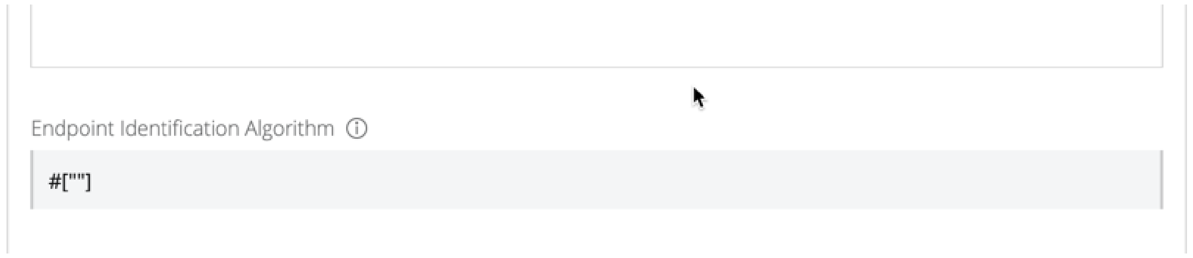


heroku0104.jks

Type: JKS

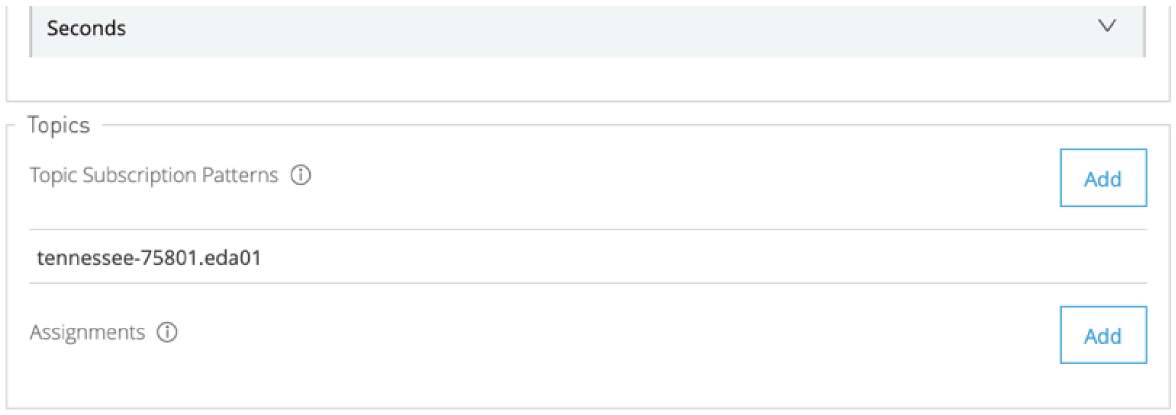
Alias:

Key Password: •••••••



Endpoint Identification Algorithm: #[""]

Now as you continue filling in the connection info in the Kafka Listener Mule connector - click on Advanced and scroll down to the bottom. Fill in the topic name as shown below.



Seconds

Topics

Topic Subscription Patterns

tennessee-75801.eda01

Assignments

Now test (you should get success). Save and again Save.

The rest of the workshop would work as explained in the Event-Driven Architecture Student Guide: Salesforce Event-Driven Architecture Hands-on Workshop: Student Workbook

Resources

- Salesforce Event-Driven Architecture Hands-on Workshop: Student Workbook
- How to connect to Heroku Kafka from Mule - In my example above I have used the compact keytool command and did not need Jetty as this blog advises: https://blogs.mulesoft.com/dev-guides/api-connectors-templates/how-to-connect-to-apache-kafka-on-heroku/