**Quick Start for Confluent Platform (Local install)**

Use this quick start to get up and running with Confluent Platform and its main components in a development environment. This quick start uses Confluent Control Center included in Confluent Platform for topic management and event stream processing using ksqlDB.

In this quick start, you create Apache Kafka® topics, use Kafka Connect to generate mock data to those topics, and create ksqlDB streaming queries on those topics. You then go to Control Center to monitor and analyze the streaming queries.

**Prerequisites:**

* Internet connectivity.
* Operating System currently supported by Confluent Platform.
* A supported version of Java downloaded and installed.

Java 8 and Java 11 are supported in this version of Confluent Platform (Java 9 and 10 are not supported).

**Step 1: Download and Start Confluent Platform**

1. Go to the downloads page.
2. Scroll to the **Download Confluent Platform** section and provide the following:
   * Email: Your email address
   * Format: deb, rpm, tar, or zip
3. Click **DOWNLOAD FREE**.
4. Decompress the file. You should have the directories, such as bin and etc.
5. Set the environment variable for the Confluent Platform directory.

export CONFLUENT\_HOME=<path-to-confluent>

1. Add the Confluent Platform bin directory to your PATH.

export PATH=$PATH:$CONFLUENT\_HOME/bin

1. Install the Kafka Connect Datagen source connector using the Confluent Hub client. This connector generates mock data for demonstration purposes and is not suitable for production. Confluent Hub is an online library of pre-packaged and ready-to-install extensions or add-ons for Confluent Platform and Kafka.

confluent-hub install **\**

--no-prompt confluentinc/kafka-connect-datagen:latest

1. Start Confluent Platform using the Confluent CLI confluent local services start command. This command starts all of the Confluent Platform components, including Kafka, ZooKeeper, Schema Registry, HTTP REST Proxy for Kafka, Kafka Connect, ksqlDB, and Control Center.

Important

The confluent local commands are intended for a single-node development environment and are not suitable for a production environment. The data that are produced are transient and are intended to be temporary.

confluent local services start

Your output should resemble:

Starting Zookeeper

Zookeeper is **[**UP**]**

Starting Kafka

Kafka is **[**UP**]**

Starting Schema Registry

Schema Registry is **[**UP**]**

Starting Kafka REST

Kafka REST is **[**UP**]**

Starting Connect

Connect is **[**UP**]**

Starting KSQL Server

KSQL Server is **[**UP**]**

Starting Control Center

Control Center is **[**UP**]**

**Step 2: Create Kafka Topics**

In this step, you create Kafka topics using Confluent Control Center. Confluent Control Center provides the functionality for building and monitoring production data pipelines and event streaming applications.

1. Navigate to the Control Center web interface at http://localhost:9021.

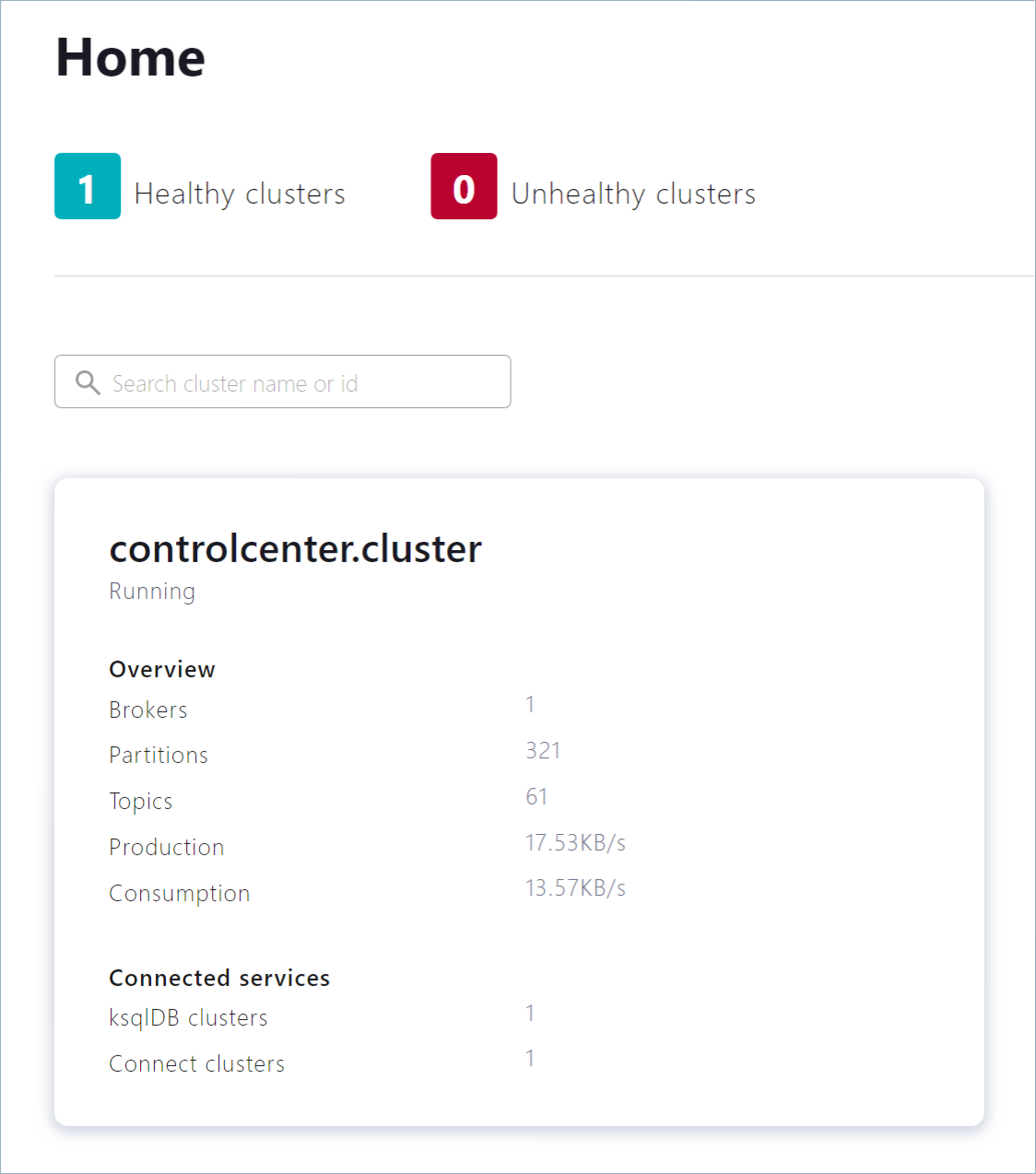
If you installed Confluent Platform on a different host, replace localhost with the host name in the address.

It may take a minute or two for Control Center to come online.

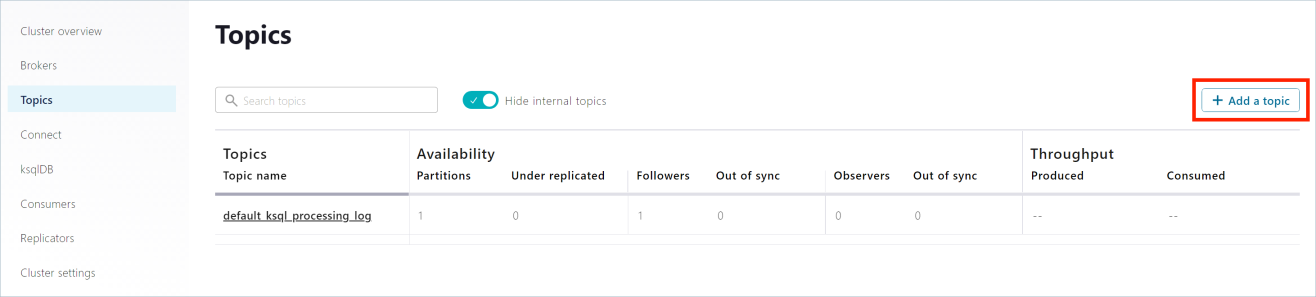
Note

Control Center won’t connect to ksqlDB if Control Center isn’t open and running in a localhost browser session.

1. Click the **controlcenter.cluster** tile.

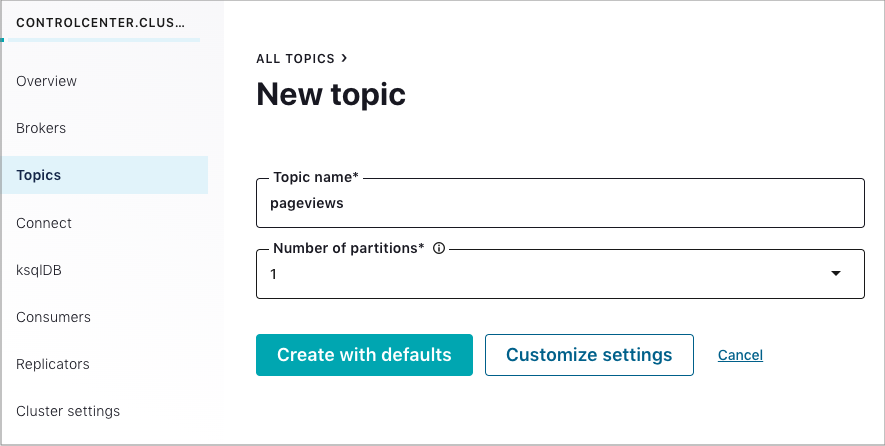
[](https://docs.confluent.io/platform/current/_images/c3-landing-page.png)

1. In the navigation bar, click **Topics** to open the topics list, and then click **Add a topic**.

[](https://docs.confluent.io/platform/current/_images/c3-create-topic.png)

1. In the Topic name field, specify pageviews and click **Create with defaults**.

Note that topic names are case-sensitive.

[](https://docs.confluent.io/platform/current/_images/c3-create-topic-name.png)

1. In the navigation bar, click **Topics** to open the topics list, and then click **Add a topic**.
2. In the Topic name field, specify users and click **Create with defaults**.

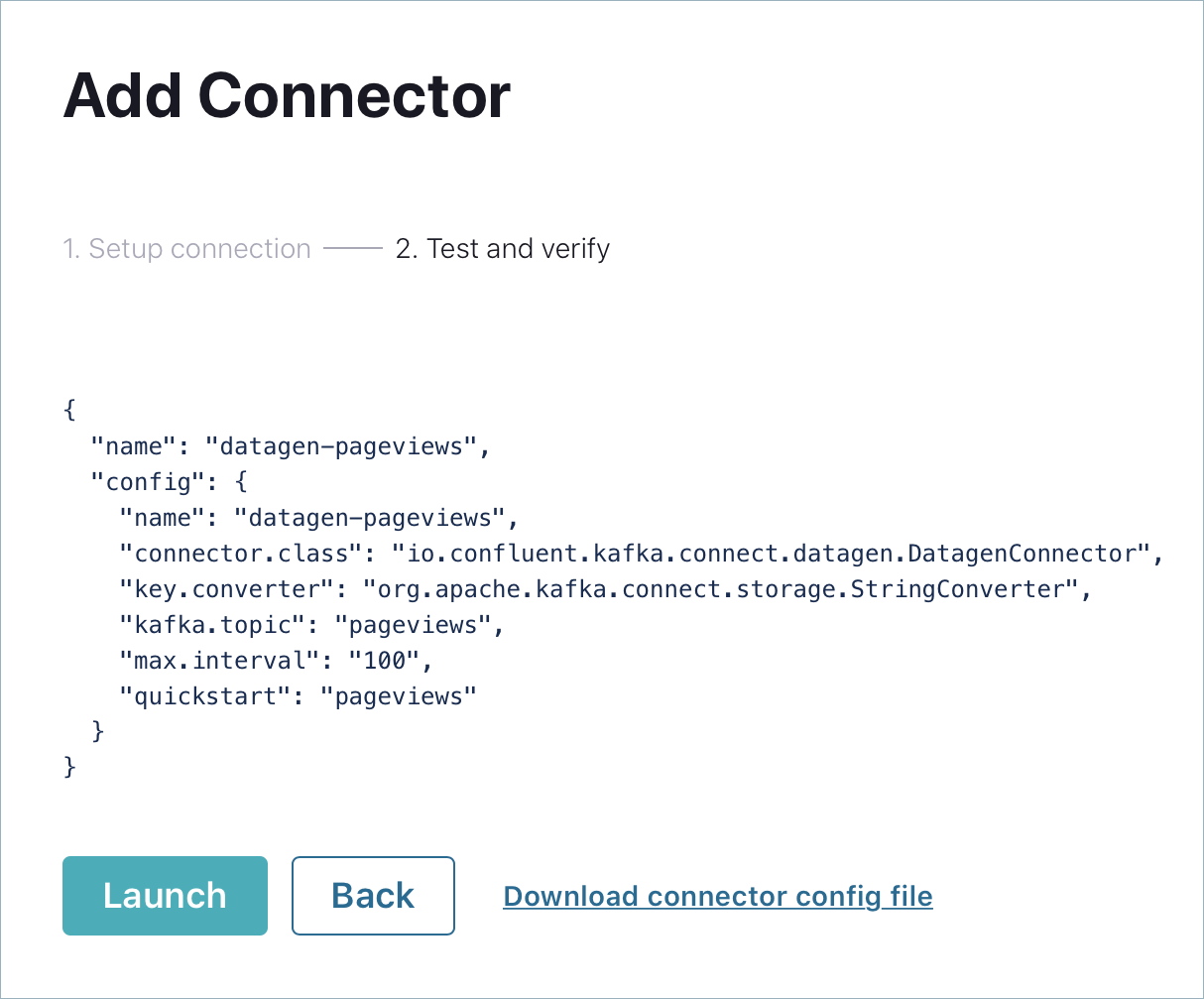
**Step 3: Install a Kafka Connector and Generate Sample Data**

In this step, you use Kafka Connect to run a demo source connector called kafka-connect-datagen that creates sample data for the Kafka topics pageviews and users.

Tip

The Kafka Connect Datagen connector was installed manually in Step 1: Download and Start Confluent Platform.

1. Run the first instance of the Kafka Connect Datagen connector to produce Kafka data to the pageviews topic in AVRO format.
   1. In the navigation bar, click **Connect**.
   2. Click the connect-default cluster in the **Connect Clusters** list.
   3. Click **Add connector**.
   4. Select the DatagenConnector tile.
   5. In the **Name** field, enter datagen-pageviews as the name of the connector.
   6. Enter the following configuration values:
      * **Key converter class:** org.apache.kafka.connect.storage.StringConverter.
      * **kafka.topic:** pageviews.
      * **max.interval:** 100.
      * **quickstart:** pageviews.
   7. Click **Next**.
   8. Review the connector configuration and click **Launch**.

[](https://docs.confluent.io/platform/current/_images/connect-review-pageviews.png)

1. Run the second instance of the Kafka Connect Datagen connector to produce Kafka data to the users topic in AVRO format.
   1. Click **Add connector**.
   2. Select the DatagenConnector tile.
   3. In the **Name** field, enter datagen-users as the name of the connector.
   4. Enter the following configuration values:
      * **Key converter class:** org.apache.kafka.connect.storage.StringConverter
      * **kafka.topic:** users
      * **max.interval:** 1000
      * **quickstart:** users
   5. Click **Next**.
   6. Review the connector configuration and click **Launch**.

**Step 4: Create and Write to a Stream and Table using ksqlDB**

Tip

You can also run these commands using the ksqlDB CLI from your terminal with this command: <path-to-confluent>/bin/ksql http://localhost:8088.

Create Streams and Tables

In this step, you use ksqlDB to create a stream for the pageviews topic and a table for the users topic.

1. In the navigation bar, click **ksqlDB**.
2. Select the ksqlDB application.
3. Copy the following code into the editor window and click **Run query** to create the pageviews stream. Stream names are not case-sensitive.

**CREATE** STREAM pageviews **WITH** (KAFKA\_TOPIC**=**'pageviews', VALUE\_FORMAT**=**'AVRO');

1. Copy the following code into the editor window and click **Run query** to create the users table. Table names are not case-sensitive.

**CREATE** **TABLE** users (id VARCHAR **PRIMARY** **KEY**)

**WITH** (KAFKA\_TOPIC**=**'users', VALUE\_FORMAT**=**'AVRO');

Write Queries

In this step, you create ksqlDB queries against the stream and the table you created above.

1. In the **Editor** tab, click **Add query properties** to add a custom query property.
2. Set the auto.offset.reset parameter to Earliest.

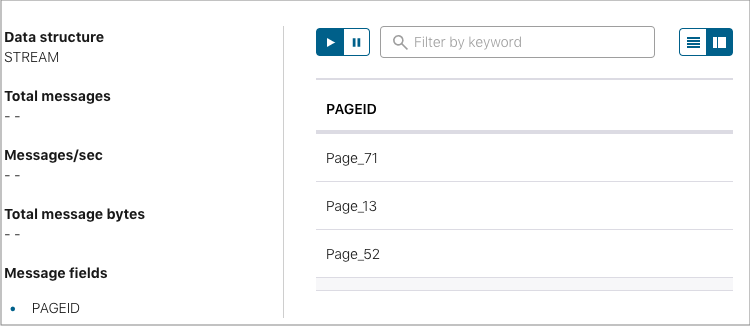
The setting instructs ksqlDB queries to read all available topic data from the beginning. This configuration is used for each subsequent query.

1. Create the following queries.
   1. Click **Stop** to stop the current running query.
   2. Create a non-persistent query that returns data from a stream with the results limited to a maximum of three rows:

Enter the following query in the editor:

SELECT pageid FROM pageviews EMIT CHANGES LIMIT 3;

* 1. Click **Run query**. Your output should resemble:

[](https://docs.confluent.io/platform/current/_images/c3-ksql-query-results-pageid.png)

Click the **Card view** or **Table view** icon to change the output layout.

* 1. Create a persistent query (as a stream) that filters the PAGEVIEWS stream for female users. The results from this query are written to the Kafka PAGEVIEWS\_FEMALE topic:

Enter the following query in the editor:

CREATE STREAM pageviews\_female

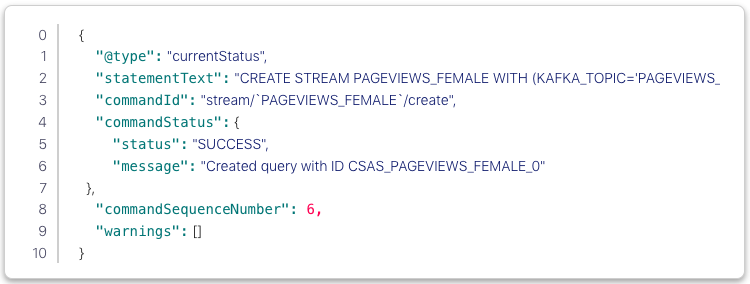
AS SELECT users.id AS userid, pageid, regionid

FROM pageviews LEFT JOIN users ON pageviews.userid **=** users.id

WHERE gender **=** 'FEMALE'

EMIT CHANGES;

* 1. Click **Run query**. Your output should resemble:

[](https://docs.confluent.io/platform/current/_images/c3-ksql-persist-query-pv-female-results.png)

* 1. Create a persistent query where REGIONID ends with 8 or 9. Results from this query are written to the Kafka topic named pageviews\_enriched\_r8\_r9 as explicitly specified in the query:

Enter the following query in the editor:

CREATE STREAM pageviews\_female\_like\_89

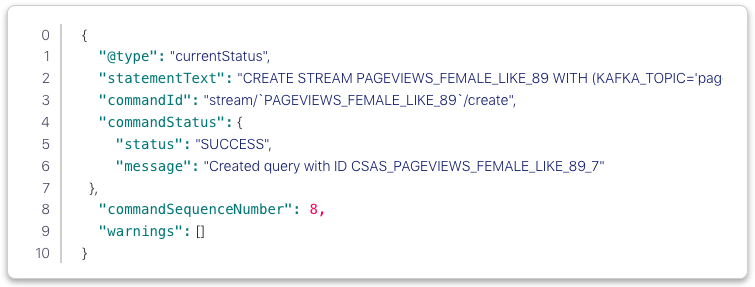
WITH **(**KAFKA\_TOPIC**=**'pageviews\_enriched\_r8\_r9', VALUE\_FORMAT**=**'AVRO'**)**

AS SELECT \* FROM pageviews\_female

WHERE regionid LIKE '%\_8' OR regionid LIKE '%\_9'

EMIT CHANGES;

* 1. Click **Run query**. Your output should resemble:

[](https://docs.confluent.io/platform/current/_images/c3-ksql-persist-query-pv-female89-results.png)

* 1. Create a persistent query that counts the PAGEVIEWS for each REGION and GENDER combination in a tumbling window of 30 seconds when the count is greater than 1. Because the procedure is grouping and counting, the result is now a table, rather than a stream. Results from this query are written to a Kafka topic called PAGEVIEWS\_REGIONS:

Enter the following query in the editor:

CREATE TABLE pageviews\_regions WITH **(**KEY\_FORMAT**=**'JSON'**)**

AS SELECT gender, regionid, COUNT**(**\***)** AS numusers

FROM pageviews LEFT JOIN users ON pageviews.userid **=** users.id

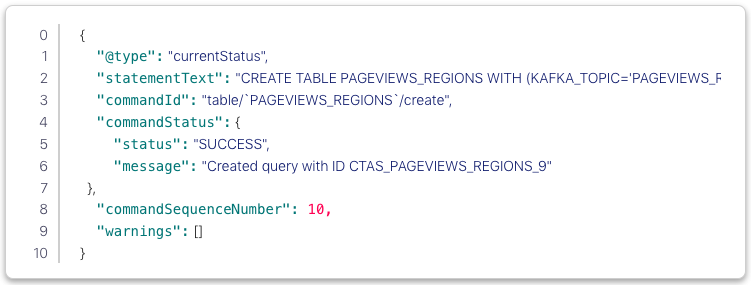
WINDOW TUMBLING **(**SIZE 30 SECOND**)**

GROUP BY gender, regionid

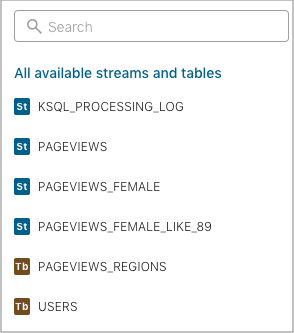
HAVING COUNT**(**\***)** > 1

EMIT CHANGES;

* 1. Click **Run query**. Your output should resemble:

[](https://docs.confluent.io/platform/current/_images/c3-ksql-persist-query-table-results.png)

* 1. Click the **Persistent queries** tab. You should see the following persisted queries:
     + PAGEVIEWS\_FEMALE
     + PAGEVIEWS\_FEMALE\_LIKE\_89
     + PAGEVIEWS\_REGIONS
  2. Click the **Editor** tab. The **All available streams and tables** pane shows all of the streams and tables that you can access.

[](https://docs.confluent.io/platform/current/_images/c3-ksql-stream-table-view-1.png)

* 1. In the **All available streams and tables** section, click **KSQL\_PROCESSING\_LOG** to view the stream’s schema, including nested data structures.

Run Queries

In this step, you run the ksqlDB queries you save as streams and tables above in the previous section.

1. In the **Streams** tab, select the PAGEVIEWS\_FEMALE stream.
2. Click **Query stream**.

The editor opens, and streaming output of the query displays.

1. Click **Stop** to stop the output generation.
2. In the **Tables** tab, select PAGEVIEWS\_REGIONS table.
3. Click **Query table**.

The editor opens, and streaming output of the query displays.

1. Click **Stop** to stop the output generation.

**Step 5: Monitor Consumer Lag**

1. In the navigation bar, click **Consumers** to view the consumers created by ksqlDB.
2. Click the consumer group ID to view details for the \_confluent-ksql-default\_query\_CSAS\_PAGEVIEWS\_FEMALE\_5 consumer group.

From the page, you can see the consumer lag and consumption values for your streaming query.

**Step 6: Stop Confluent Platform**

When you are done working with the local install, you can stop Confluent Platform.

1. Stop Confluent Platform using the Confluent CLI confluent local services connect stop command.

confluent local services stop

1. Destroy the data in the Confluent Platform instance with the confluent local destroy command.

confluent local destroy

**Troubleshooting**

If you encountered any issues while going through the quickstart workflow, review the following resolutions before trying the steps again.

Issue: Cannot locate the Datagen connector

**Resolution:** Verify that you have added the location of the Confluent Platform bin directory to your PATH as described in Step 1: Download and Start Confluent Platform:

export PATH**=**<path-to-confluent>/bin:$PATH

**Resolution:** Verify the DataGen connector is installed and running.

Ensure that the kafka-connect-datagen is installed and running as described in Step 1: Download and Start Confluent Platform.

confluent-hub install --no-prompt confluentinc/kafka-connect-datagen:latest

Your output should resemble:

Running **in** a "--no-prompt" mode

...

Completed

**Resolution:** Check the connect logs for Datagen using the Confluent CLI confluent local services connect log command.

confluent local services connect log | grep -i Datagen

Your output should resemble:

**[**2019-04-18 14:21:08,840**]** INFO Loading plugin from: /Users/user.name/Confluent/confluent-version/share/confluent-hub-components/confluentinc-kafka-connect-datagen **(**org.apache.kafka.connect.runtime.isolation.DelegatingClassLoader:215**)**

**[**2019-04-18 14:21:08,894**]** INFO Registered loader: PluginClassLoader**{**pluginLocation**=**file:/Users/user.name/Confluent/confluent-version/share/confluent-hub-components/confluentinc-kafka-connect-datagen/**}** **(**org.apache.kafka.connect.runtime.isolation.DelegatingClassLoader:238**)**

**[**2019-04-18 14:21:08,894**]** INFO Added plugin 'io.confluent.kafka.connect.datagen.DatagenConnector' **(**org.apache.kafka.connect.runtime.isolation.DelegatingClassLoader:167**)**

**[**2019-04-18 14:21:09,882**]** INFO Added aliases 'DatagenConnector' and 'Datagen' to plugin 'io.confluent.kafka.connect.datagen.DatagenConnector' **(**org.apache.kafka.connect.runtime.isolation.DelegatingClassLoader:386**)**

**Resolution:** Verify the .jar file for kafka-connect-datagen has been added and is present in the lib subfolder.

ls $CONFLUENT\_HOME/share/confluent-hub-components/confluentinc-kafka-connect-datagen/lib/

Your output should resemble:

...

kafka-connect-datagen-0.1.0.jar

...

**Resolution:** Verify the plugin exists in the connector path.

When you installed the kafka-connect-datagen file from Confluent hub, the installation directory is added to the plugin path of several properties files:

Adding installation directory to plugin path **in** the following files:

/Users/user.name/Confluent/confluent-version/etc/kafka/connect-distributed.properties

/Users/user.name/Confluent/confluent-version/etc/kafka/connect-standalone.properties

/Users/user.name/Confluent/confluent-version/etc/schema-registry/connect-avro-distributed.properties

/Users/user.name/Confluent/confluent-version/etc/schema-registry/connect-avro-standalone.properties

...

You can use any of them to check the connector path. This example uses the connect-avro-distributed.properties file.

grep plugin.path $CONFLUENT\_HOME/etc/schema-registry/connect-avro-distributed.properties

Your output should resemble:

plugin.path**=**share/java,/Users/user.name/Confluent/confluent-version/share/confluent-hub-components

Confirm its contents are present:

ls $CONFLUENT\_HOME/share/confluent-hub-components/confluentinc-kafka-connect-datagen

Your output should resemble:

assets doc lib manifest.json

Issue: Stream-Stream joins error

An error states Stream-Stream joins must have a WITHIN clause specified. This error can occur if you created both pageviews and users as streams by mistake.

[](https://docs.confluent.io/platform/current/_images/c3-ksql-stream-stream-join-error.png)

**Resolution:** Ensure that you created a *stream* for pageviews, and a *table* for users in Step 4: Create and Write to a Stream and Table using ksqlDB.

Issue: Unable to successfully complete ksqlDB query steps

Java errors or other severe errors were encountered.

**Resolution:** Ensure you are on an Operating System currently supported by Confluent Platform.