Confluent Platform Quick Start (Docker)

This quick start shows you how to get up and running with Confluent Platform and its main components using Docker containers. This quick start demonstrates both the basic and most powerful capabilities of Confluent Platform, including using Control Center for topic management and event stream processing using KSQL. In this quick start, you create Apache Kafka® topics, use Kafka Connect to generate mock data to those topics, and create KSQL streaming queries on those topics. You then go to Control Center to monitor and analyze the event streaming queries.

❗ See also

You can also run an automated version of this quick start (https://github.com/confluentinc/examples/tree/5.4.0-post/cp-quickstart/README.md) designed for Confluent Platform local installs.

Prerequisites:

- Docker:

- Docker version 1.11 or later is installed and running (https://docs.docker.com/engine/installation/)

the default Docker memory allocation is 2 GB. You can change the default allocation to 8 GB in **Docker** > **Preferences** > **Advanced**.

- Git (https://git-scm.com/downloads).
- Internet connectivity.
- Ensure you are on an Operating System (../installation/versions-interoperability.html#operating-systems) currently supported by Confluent Platform.
- Networking and Kafka on Docker: Configure your hosts and ports to allow both internal and external components to the Docker network to communicate. For more details, see this article (https://rmoff.net/2018/08/02/kafka-listeners-explained/).

Step 1: Download and Start Confluent Platform Using Docker

1. Clone the Confluent Platform Docker Images GitHub Repository and check out the `5.4.0-post` branch.

```
git clone https://github.com/confluentinc/examples
cd examples
git checkout 5.4.0-post
```

2. Navigate to `cp-all-in-one` examples directory.

3. Start Confluent Platform specifying two options: (`-d`) to run in detached mode and (`--build`) to build the Kafka Connect image with the source connector `kafka-connect-datagen` from Confluent Hub (https://www.confluent.io/connector/kafka-connect-datagen/).

Important

You must allocate a minimum of 8 GB of Docker memory resource. The default memory allocation on Docker Desktop for Mac is 2 GB and must be changed.

```
docker-compose up -d --build
```

This starts Confluent Platform with separate containers for all Confluent Platform components. Your output should resemble the following:

```
Creating network "cp-all-in-one_default" with the default driver
Creating zookeeper ... done
Creating broker    ... done
Creating schema-registry ... done
Creating rest-proxy      ... done
Creating connect         ... done
Creating ksql-datagen    ... done
Creating ksql-server     ... done
Creating control-center  ... done
Creating ksql-cli        ... done
```

4. Optional: Run this command to verify that the services are up and running.

v5.4.0 v5.3.2 v5.3.1

```
     Name                  Command               State                P
rts
---------------------------------------------------------------------------
----------------
broker            /etc/confluent/docker/run        Up      0.0.0.0:29092->
29092/tcp,
                                                           0.0.0.0:9092->9
092/tcp
connect           /etc/confluent/docker/run        Up      0.0.0.0:8083->8
083/tcp,
                                                           9092/tcp
control-center    /etc/confluent/docker/run        Up      0.0.0.0:9021->9
021/tcp
ksql-cli          ksql http://localhost:8088       Up
ksql-datagen      bash -c echo Waiting for K ...   Up
ksql-server       /etc/confluent/docker/run        Up      0.0.0.0:8088->8
088/tcp
rest-proxy        /etc/confluent/docker/run        Up      0.0.0.0:8082->8
082/tcp
schema-registry   /etc/confluent/docker/run        Up      0.0.0.0:8081->8
081/tcp
zookeeper         /etc/confluent/docker/run        Up      0.0.0.0:2181->2
181/tcp,
                                                           2888/tcp, 3888/
tcp
```

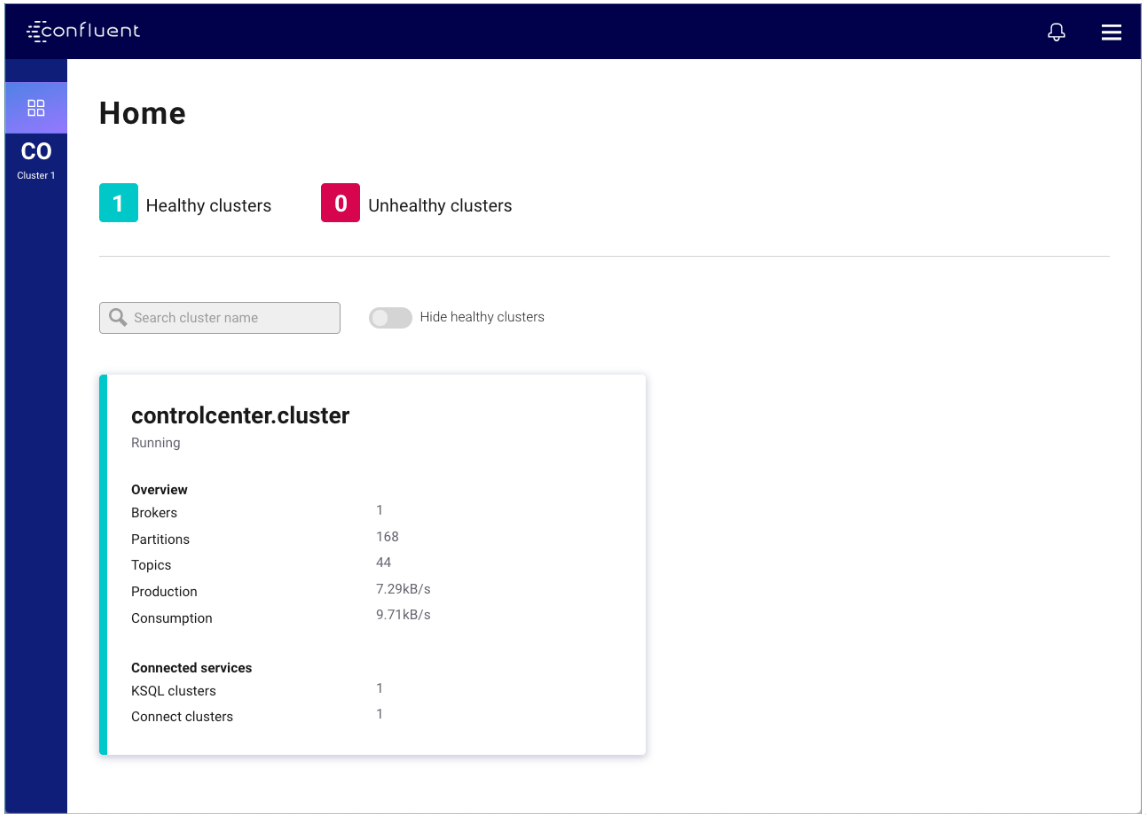
If the state is not `Up`, rerun the `docker-compose up -d` command.

monitoring production data pipelines and event streaming applications.

1. Navigate to the Control Center web interface at http://localhost:9021/ (http://localhost:9021/) and select your cluster.

   Important

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



Home

1 Healthy clusters

0 Unhealthy clusters

Search cluster name

Hide healthy clusters

controlcenter.cluster

Running

Overview

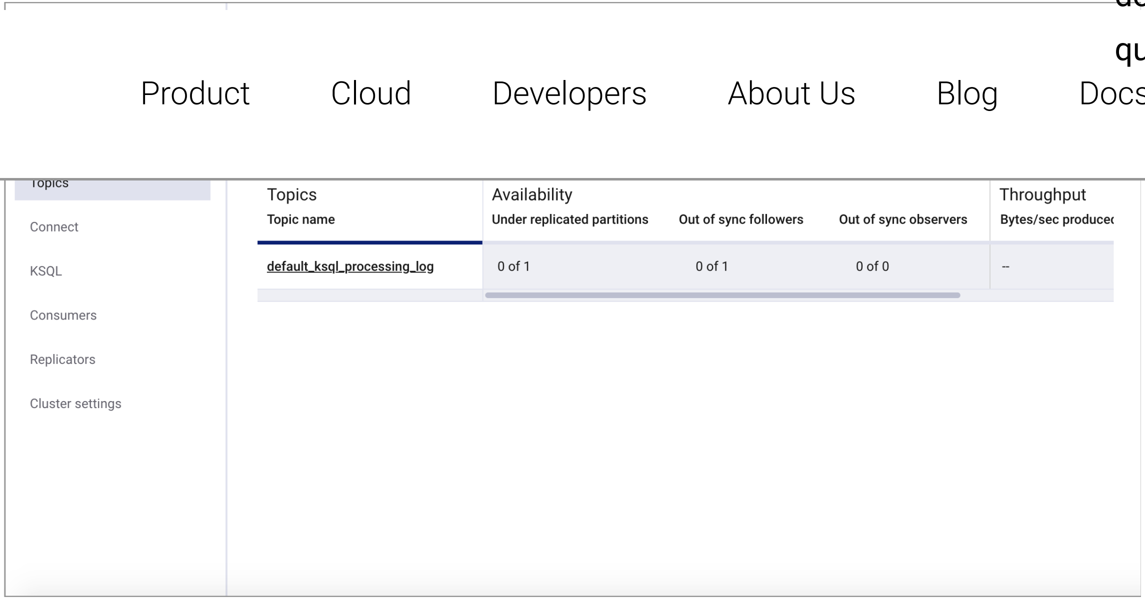
Brokers	1
Partitions	168
Topics	44
Production	7.29kB/s
Consumption	9.71kB/s

Connected services

KSQL clusters	1
Connect clusters	1

(../_images/c3-landing-page.png)

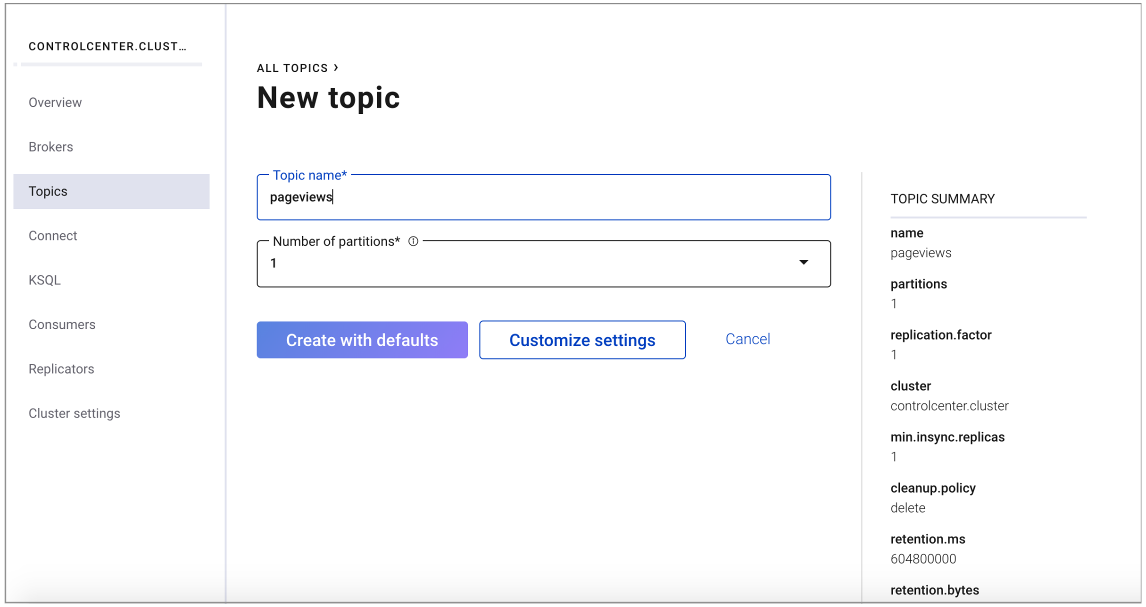
2. Select **Topics** from the cluster submenu and click **Add a topic**.



Topics	Availability			Throughput
Topic name	Under replicated partitions	Out of sync followers	Out of sync observers	Bytes/sec produced
default_ksql_processing_log	0 of 1	0 of 1	0 of 0	--

(../_images/c3-create-topic.png)

3. Create a topic named `pageviews` and click **Create with defaults**.



New topic

Topic name*: pageviews

Number of partitions*: 1

TOPIC SUMMARY

name: pageviews

partitions: 1

replication.factor: 1

cluster: controlcenter.cluster

min.insync.replicas: 1

cleanup.policy: delete

retention.ms: 604800000

retention.bytes

(../_images/c3-create-topic-name.png)

4. Repeat the previous steps and create a topic named `users` and click **Create with defaults**.

v5.4.0 v5.3.2 v5.3.1 v5.3.0

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`.

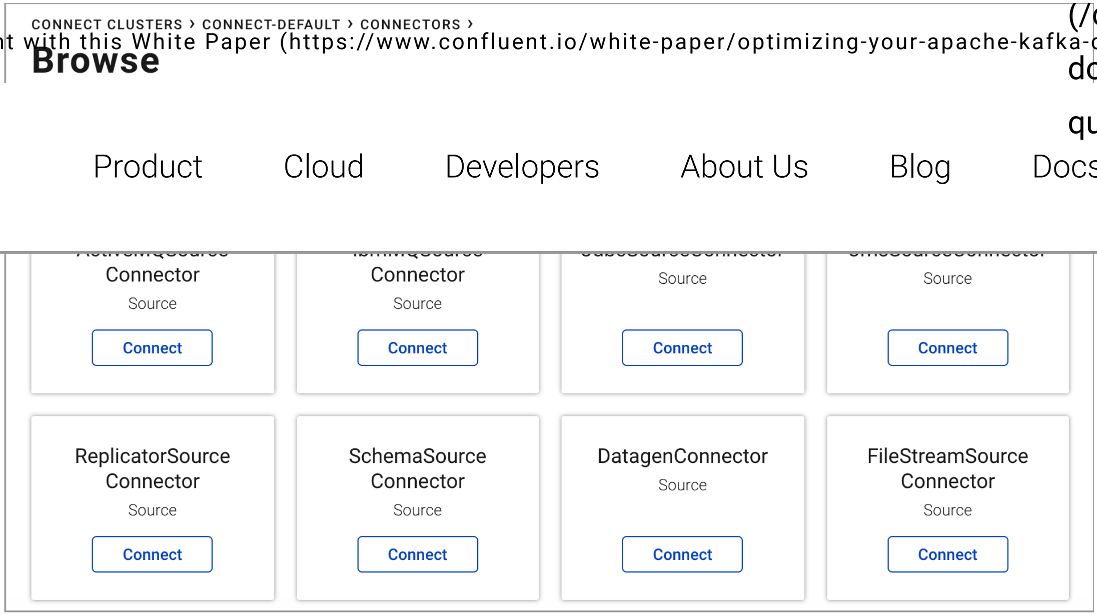
Tip

The Kafka Connect Datagen connector was installed automatically when you started Docker Compose with the `--build` argument in Step 1: Download and Start Confluent Platform Using Docker. If you encounter issues locating the Datagen Connector, refer to the Issue: Cannot locate the Datagen Connector in the Troubleshooting section.

1. Run one instance of the Kafka Connect Datagen (https://www.confluent.io/connector/kafka-connect-datagen/) connector to produce Kafka data to the `pageviews` topic in AVRO format.
   1. From your cluster, click **Connect**.
   2. Select the `connect-default` cluster and click **Add connector**.
   3. Find the DatagenConnector tile and click **Connect**.

      Tip

      To narrow displayed connectors, click **Filter by type** -> **Sources**.



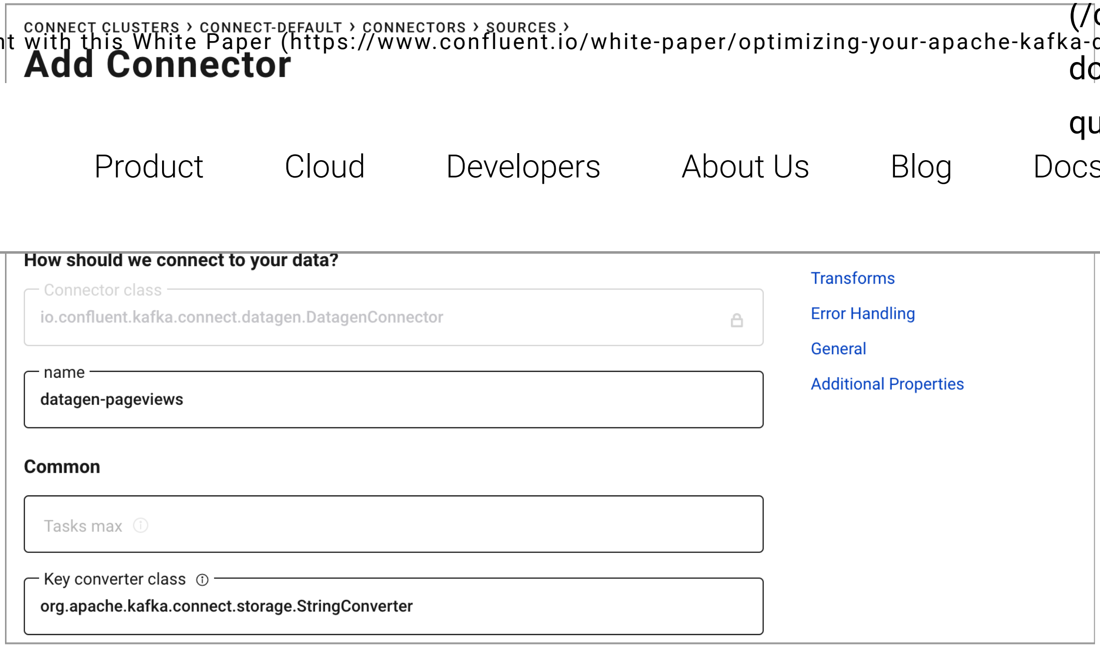
CONNECT CLUSTERS › CONNECT-DEFAULT › CONNECTORS ›

Browse

(../_images/connect-page-new-source.png)

4. Name the connector `datagen-pageviews`. After naming the connector, new fields appear. Scroll down and specify the following configuration values:

   - In the **Key converter class** field, type `org.apache.kafka.connect.storage.StringConverter`.
   - In the **kafka.topic** field, type `pageviews`.
   - In the **max.interval** field, type `100`.
   - In the **iterations** field, type `1000000000`.
   - In the **quickstart** field, type `pageviews`.



CONNECT CLUSTERS › CONNECT-DEFAULT › CONNECTORS › SOURCES ›

Add Connector

How should we connect to your data?

Connector class: io.confluent.kafka.connect.datagen.DatagenConnector

name: datagen-pageviews

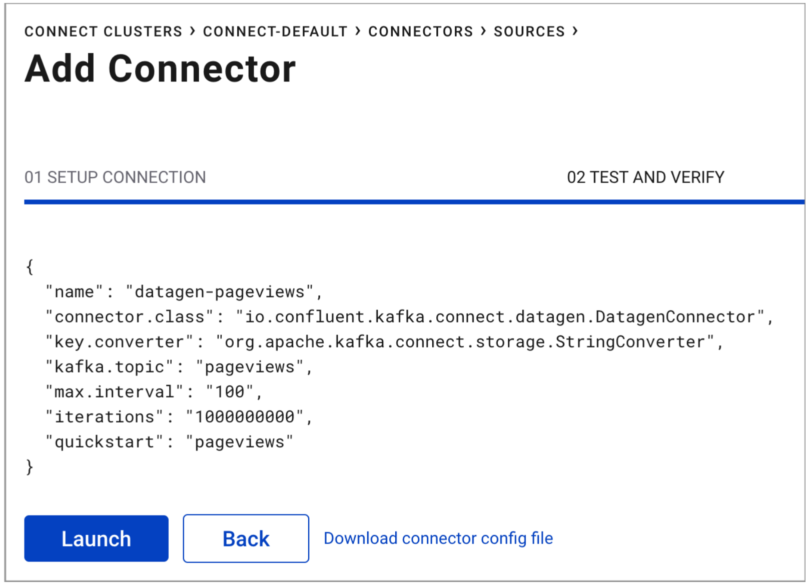
Common

Tasks max

Key converter class: org.apache.kafka.connect.storage.StringConverter

(../_images/connect-configure-pageviews.png)

5. Click **Continue**.
6. Review the connector configuration and click **Launch**.



CONNECT CLUSTERS › CONNECT-DEFAULT › CONNECTORS › SOURCES ›

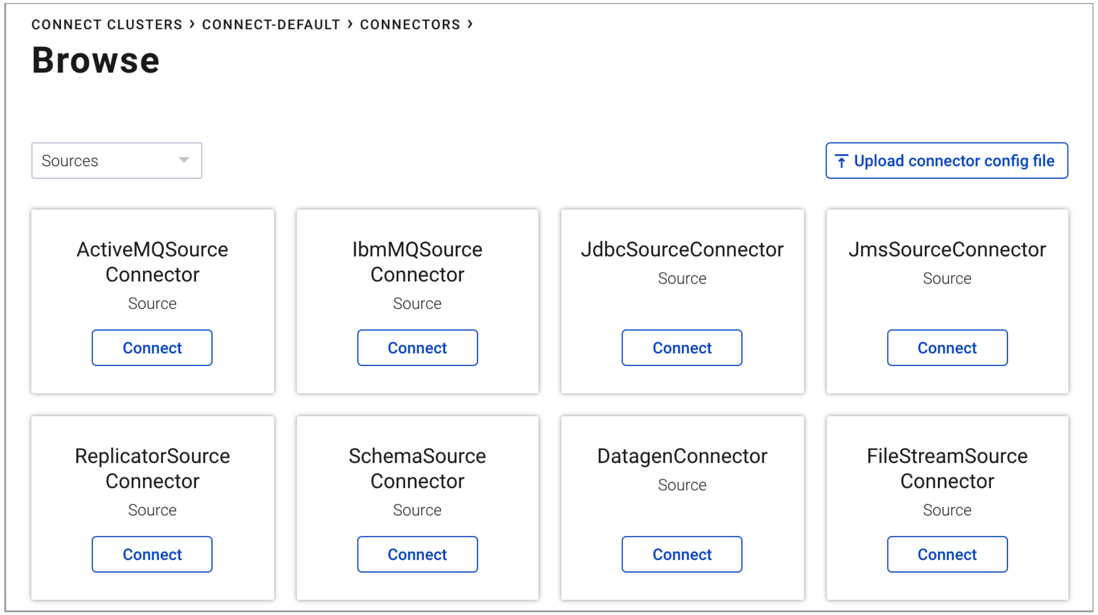
Add Connector

01 SETUP CONNECTION 02 TEST AND VERIFY

```
{
  "name": "datagen-pageviews",
  "connector.class": "io.confluent.kafka.connect.datagen.DatagenConnector",
  "key.converter": "org.apache.kafka.connect.storage.StringConverter",
  "kafka.topic": "pageviews",
  "max.interval": "100",
  "iterations": "1000000000",
  "quickstart": "pageviews"
}
```

(../_images/connect-review-pageviews.png)

2. Run another instance of the Kafka Connect Datagen (https://www.confluent.io/connector/kafka-connect-datagen/) connector to produce Kafka



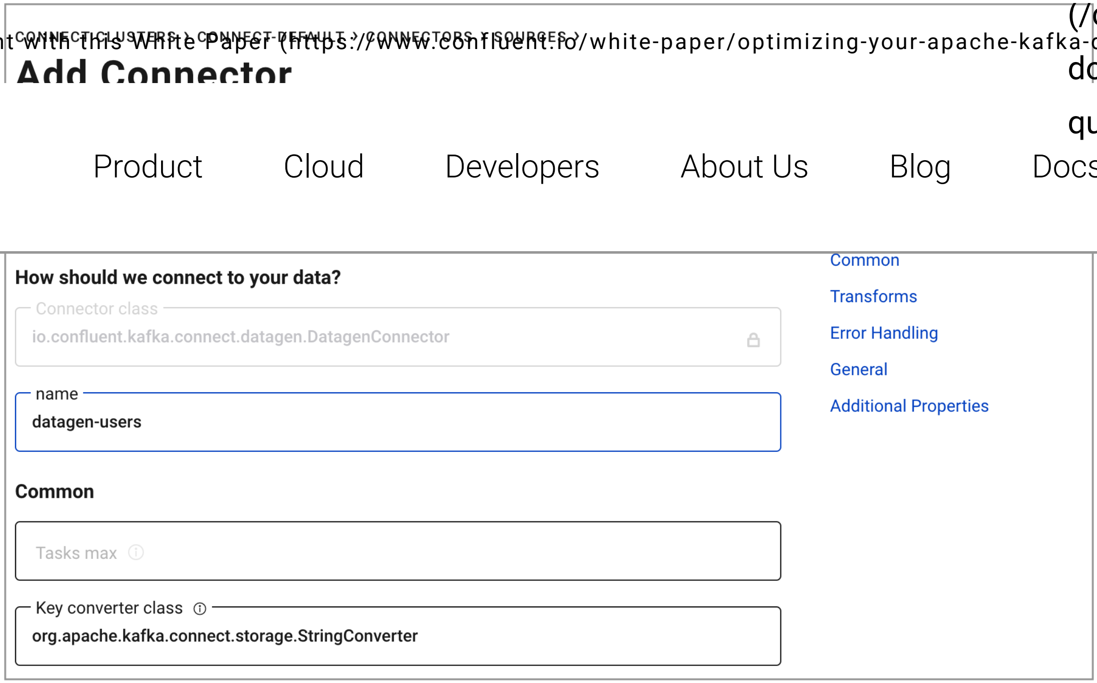
CONNECT CLUSTERS › CONNECT-DEFAULT › CONNECTORS ›

Browse

(../_images/connect-page-new-source.png)

3. Name the connector `datagen-users`. After naming the connector, new fields appear. Scroll down and specify the following configuration values:

   - In the **Key converter class** field, type `org.apache.kafka.connect.storage.StringConverter`.
   - In the **kafka.topic** field, type `users`.
   - In the **max.interval** field, type `1000`.
   - In the **iterations** field, type `1000000000`.
   - In the **quickstart** field, type `users`.



CONNECT CLUSTERS > CONNECT-DEFAULT > CONNECTORS > SOURCES >

Add Connector

How should we connect to your data?

Connector class: io.confluent.kafka.connect.datagen.DatagenConnector

name: datagen-users

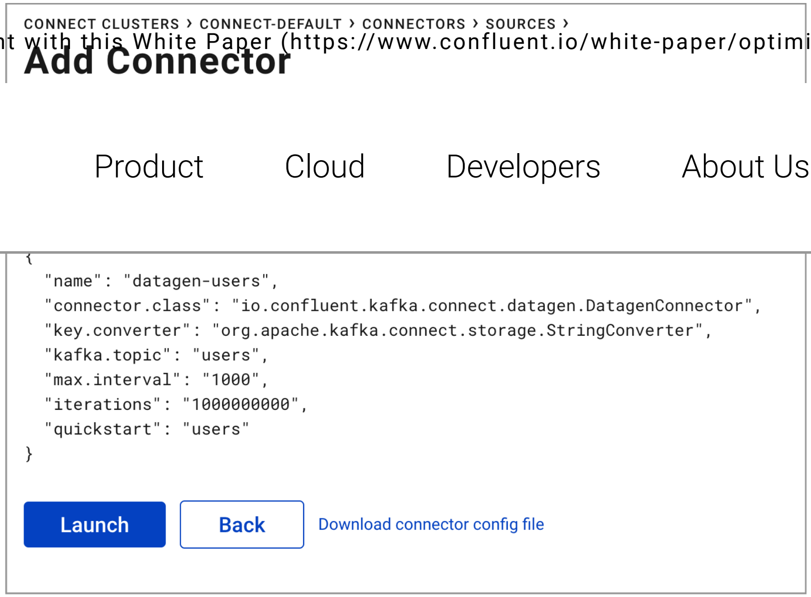
Common

Tasks max

Key converter class: org.apache.kafka.connect.storage.StringConverter

(../_images/connect-configure-users.png)

4. Click **Continue**.
5. Review the connector configuration and click **Launch**.



CONNECT CLUSTERS › CONNECT-DEFAULT › CONNECTORS › SOURCES ›

Add Connector

```
{
  "name": "datagen-users",
  "connector.class": "io.confluent.kafka.connect.datagen.DatagenConnector",
  "key.converter": "org.apache.kafka.connect.storage.StringConverter",
  "kafka.topic": "users",
  "max.interval": "1000",
  "iterations": "1000000000",
  "quickstart": "users"
}
```

(../_images/connect-review-users.png)

Step 4: Create and Write to a Stream and Table using KSQL

In this step, KSQL queries are run on the `pageviews` and `users` topics that were created in the previous step. The KSQL commands are run using the KSQL tab in Control Center.

Tip

Optimize your Apache Kafka Deployment with this White Paper (https://www.confluent.io/white-paper/optimizing-your-apache-kafka-deployment/)

You can also run these commands using the KSQL CLI
(/ksql/docs/installation/index html#install-ksql) from your Docker container with this

docker-docker-docker-
quickstart.htm

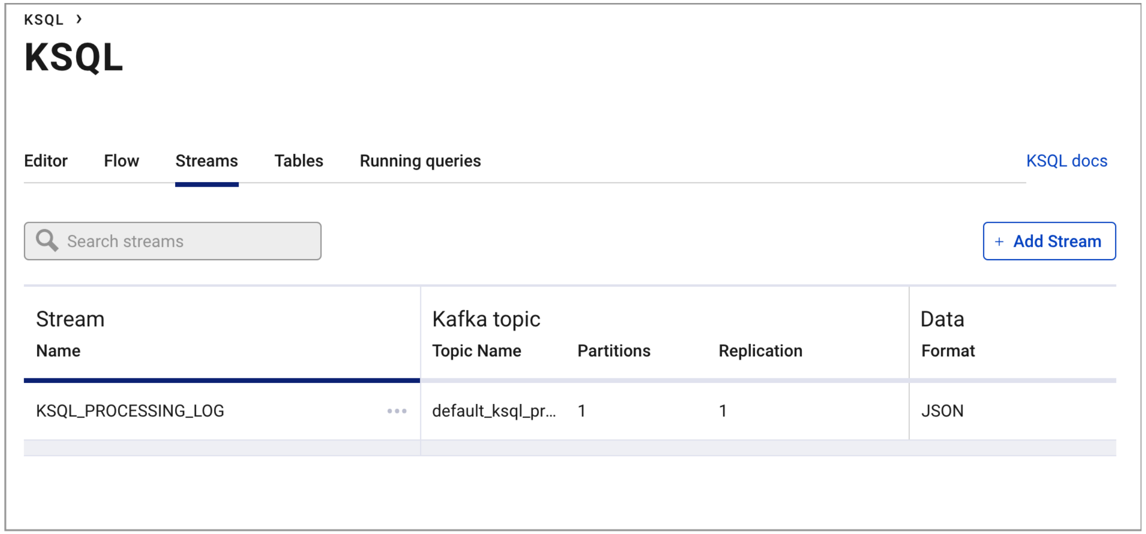
(https://docs.confluent.io)

Product Cloud Developers About Us Blog Docs Download

Create Streams and Tables

In this step, KSQL is used to create a stream for the `pageviews` topic, and a table for the `users` topic.

1. From your cluster, click **KSQL** and choose the **KSQL** application.
2. From the **KSQL EDITOR** page, click the **Streams** tab and **Add Stream**.



KSQL ›

KSQL

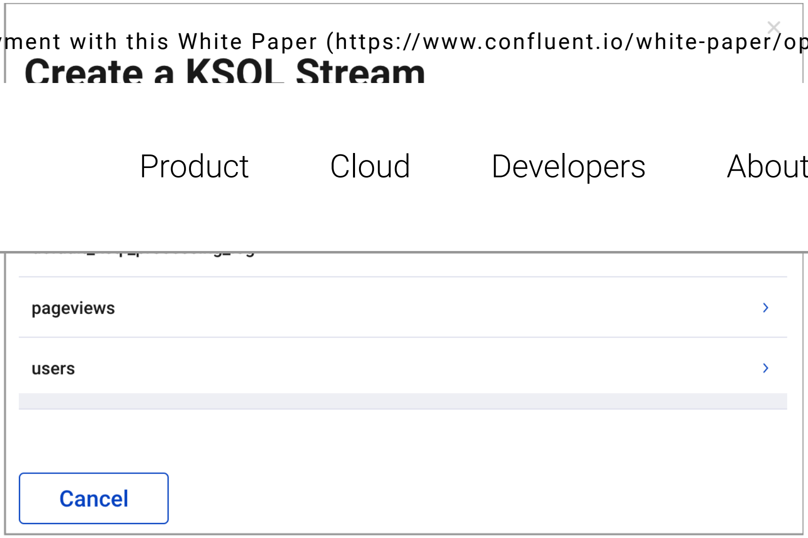
Stream Name	Kafka topic Topic Name	Partitions	Replication	Data Format
KSQL_PROCESSING_LOG	default_ksql_pr...	1	1	JSON

(../_images/ksql-interface-create-stream2.png)

3. Select the `pageviews` topic.

Expand Content

v5.4.0 v5.3.2 v5.3.1

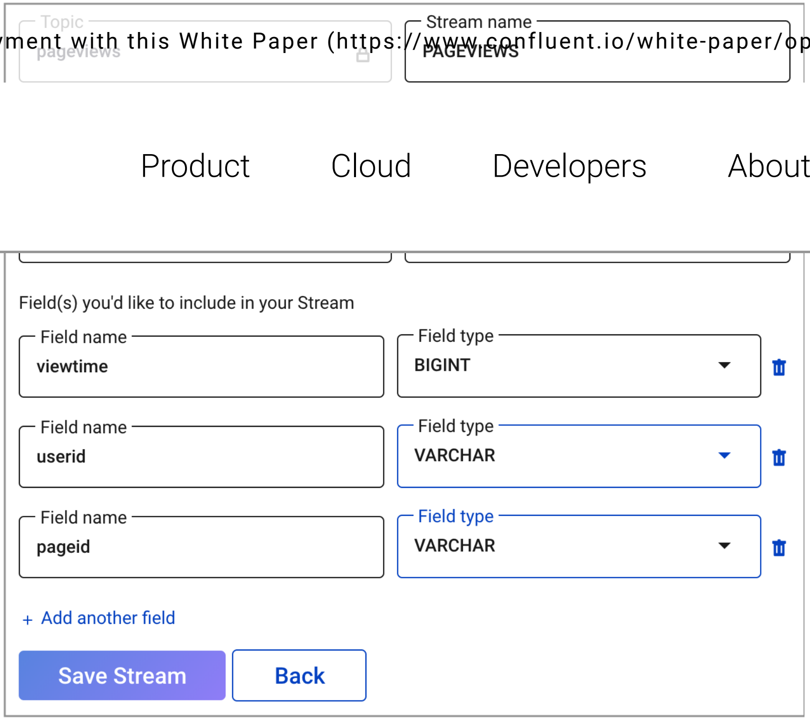


Create a KSQL Stream

(../_images/c3-ksql-create-stream-pageview.png)

4. Choose your stream options:

   - In the **Encoding** field, select `AVRO`.
   - In the **Field(s) you'd like to include in your STREAM** field, ensure fields are set as follows:
     - `viewtime` with type `BIGINT`
     - `userid` with type `VARCHAR`
     - `pageid` with type `VARCHAR`



Topic
pageviews

Stream name
PAGEVIEWS

Field(s) you'd like to include in your Stream

Field name: viewtime — Field type: BIGINT

Field name: userid — Field type: VARCHAR

Field name: pageid — Field type: VARCHAR

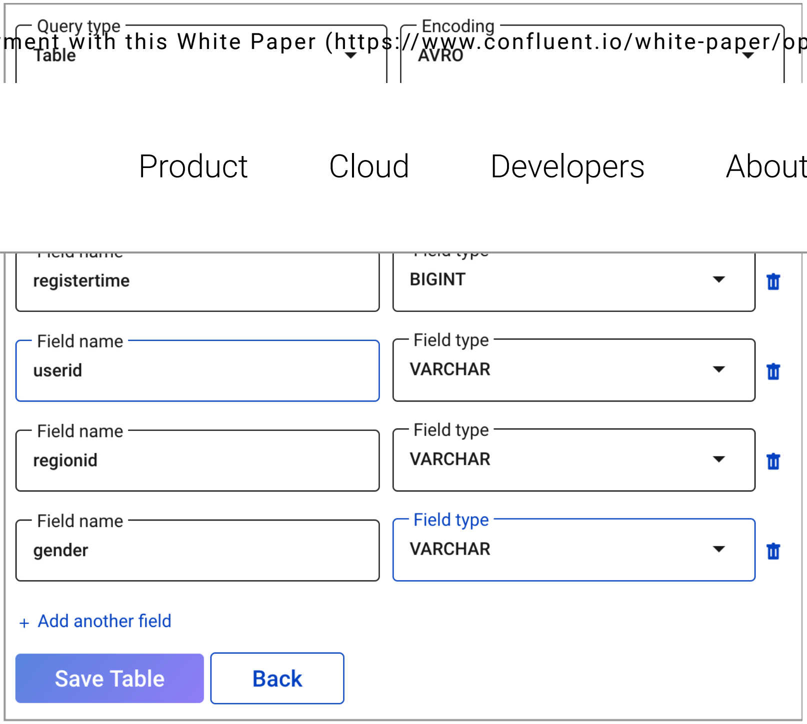
(../_images/c3-ksql-create-stream-pageview-2.png)

5. Click **Save Stream**.
6. Click the **Tables** tab -> **Add a Table** and select the `users` topic.

Create a KSQL Table

(../_images/c3-ksql-create-stream-users.png)

7. Choose your table options:
   - In the **Encoding** field, select `AVRO`.
   - In the **Key** field, select `userid`.
   - In the **Field(s) you'd like to include in your TABLE** field, ensure fields are set as follows:
     - `registertime` with type `BIGINT`
     - `userid` with type `VARCHAR`
     - `regionid` with type `VARCHAR`
     - `gender` with type `VARCHAR`



Query type: Table

Encoding: AVRO

Field name	Field type
registertime	BIGINT
userid	VARCHAR
regionid	VARCHAR
gender	VARCHAR

(../_images/c3-ksql-create-table-users.png)

8. Click **Save Table**.

Write Queries

These examples write queries using the **KSQL** tab in Control Center.

1. From your cluster, click **KSQL** and choose the **Editor** page.
2. From the **KSQL EDITOR** page, click **Add query properties** to add a custom query property. Set the `auto.offset.reset` parameter to `earliest`.

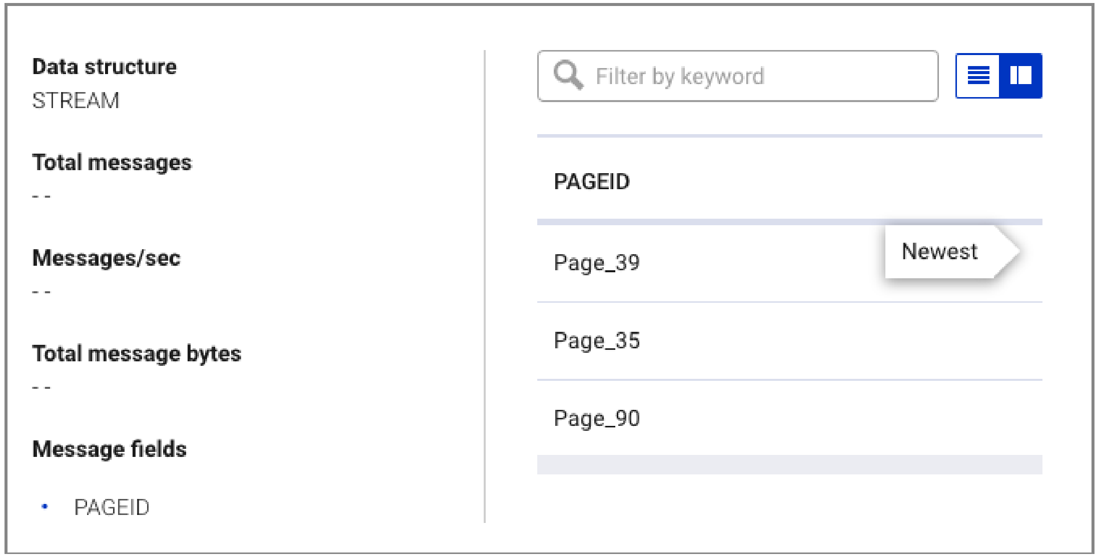
   This instructs KSQL queries to read all available topic data from the beginning. This configuration is used for each subsequent query. For more information, see the KSQL Configuration Parameter Reference (../ksql/docs/installation/server-config/config-query-properties.png)

3. Run the following queries.

   1. Create a non-persistent query that returns data from a stream with the results limited to a maximum of three rows.

```
SELECT pageid FROM pageviews EMIT CHANGES LIMIT 3;
```

   Your output should resemble:



Data structure
STREAM

Total messages
- -

Messages/sec
- -

Total message bytes
- -

Message fields
- PAGEID

PAGEID
Page_39
Page_35
Page_90

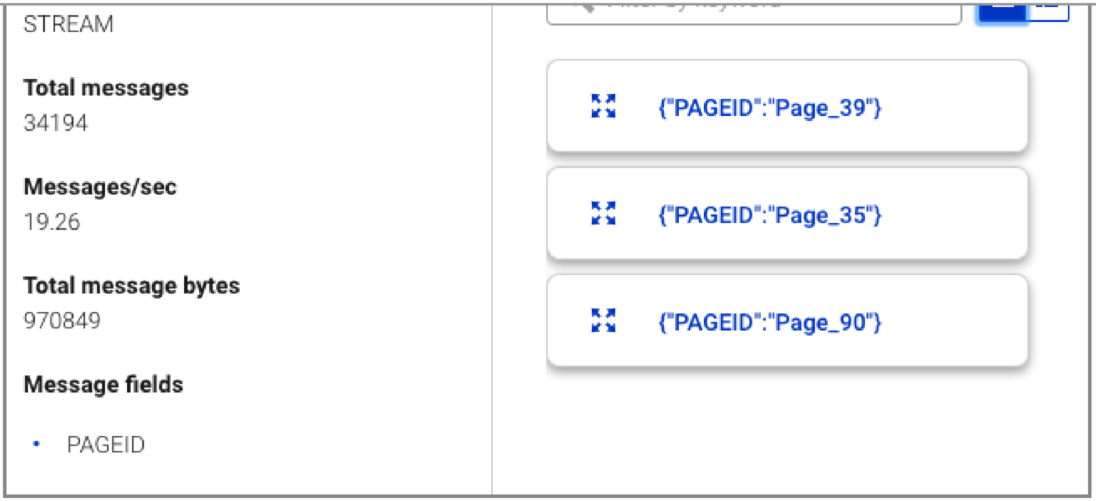
(../_images/c3-ksql-query-results-pageid.png)

❗ Tip

Expand Content

v5.4.0 v5.3.2 v5.3.1

Optimize your Apache Kafka Deployment with this White Paper (https://www.confluent.io/white-paper/optimizing-your-apache-kafka-deployment/)
(/current/5.3.2/5.3.1/quickstart/ce-docker-docker-docker-quickstart.html)
Click the **Card view** or **Tabular view** icon to change the layout. Click the expand icon to expand a message
(https://docs.confluent.io)
Product Cloud Developers About Us Blog Docs Download



STREAM

Total messages
34194

Messages/sec
19.26

Total message bytes
970849

Message fields

- PAGEID

{"PAGEID":"Page_39"}

{"PAGEID":"Page_35"}

{"PAGEID":"Page_90"}

(../_images/c3-ksql-query-results-pageid-card.png)

2. Create a persistent query that filters for female users. The results from this query are written to the Kafka `PAGEVIEWS_FEMALE` topic. This query enriches the `pageviews` STREAM by doing a `LEFT JOIN` with the `users` TABLE on the user ID, where a condition (`gender = 'FEMALE'`) is met.

```
CREATE STREAM pageviews_female AS SELECT users.userid AS userid, pageid
, regionid, gender FROM pageviews LEFT JOIN users ON pageviews.userid =
users.userid WHERE gender = 'FEMALE';
```

Your output should resemble:

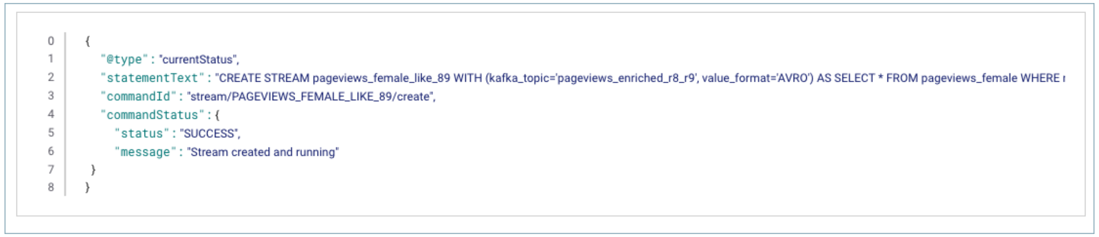
```
"@type" : "currentStatus",
"statementText" : "CREATE STREAM pageviews_female AS SELECT users.userid AS userid, pageid, regionid, gender FROM pageviews LEFT JOIN users ON pageviews.userid = users.u
```

(../_images/c3-ksql-persist-query-pv-female-results.png)

3. Create a persistent query where a condition (`regionid`) is met, using `LIKE` . Results from this query are written to a Kafka topic named `pageviews_enriched_r8_r9` .

   ```
   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';
   ```

   Your output should resemble:

   ```
   {
     "@type" : "currentStatus",
     "statementText" : "CREATE STREAM pageviews_female_like_89 WITH (kafka_topic='pageviews_enriched_r8_r9', value_format='AVRO') AS SELECT * FROM pageviews_female WHERE r
     "commandId" : "stream/PAGEVIEWS_FEMALE_LIKE_89/create",
     "commandStatus" : {
       "status" : "SUCCESS",
       "message" : "Stream created and running"
     }
   }
   ```

   (../_images/c3-ksql-persist-query-pv-female89-results.png)

4. Create a persistent query that counts the pageviews for each region and gender combination in a tumbling window (../streams/developer-guide/dsl-api.html#windowing-tumbling) 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` .

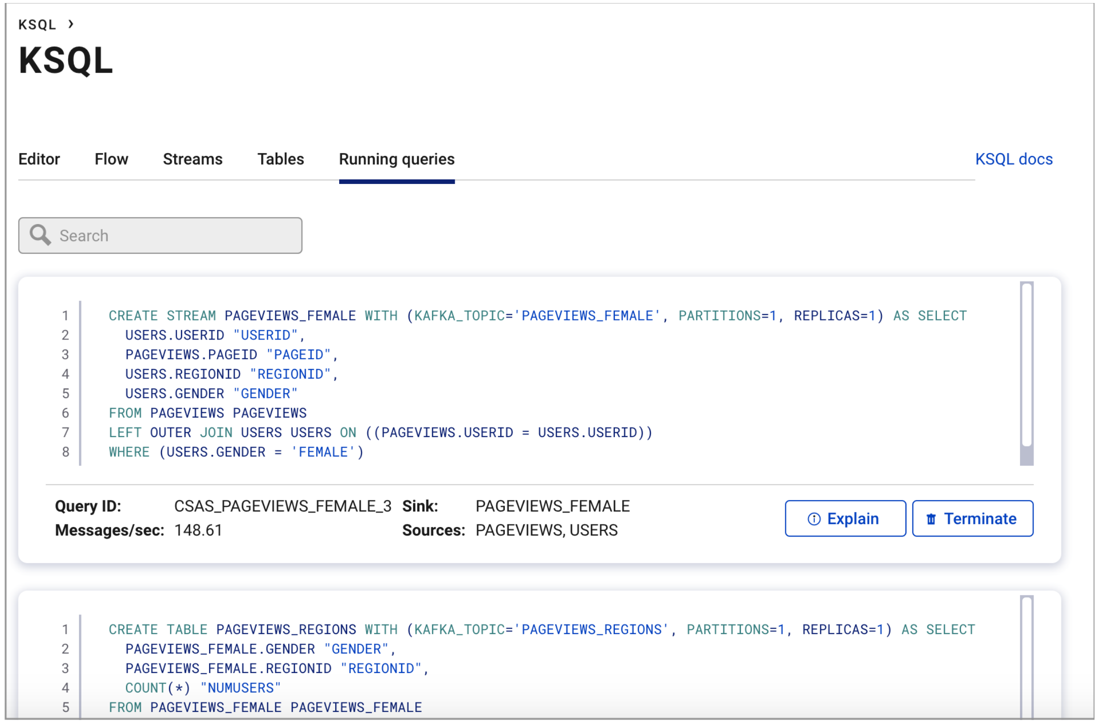
   ```
   CREATE TABLE pageviews_regions AS SELECT gender, regionid , COUNT(*) AS numusers FROM pageviews_female WINDOW TUMBLING (size 30 second) GROUP BY gender, regionid HAVING COUNT(*) > 1;
   ```

Your output should resemble:

```
}
```

(../_images/c3-ksql-persist-query-table-results.png)

5. Click **Running queries**. You should see the following persisted queries:



KSQL ›

KSQL

```
CREATE STREAM PAGEVIEWS_FEMALE WITH (KAFKA_TOPIC='PAGEVIEWS_FEMALE', PARTITIONS=1, REPLICAS=1) AS SELECT
  USERS.USERID "USERID",
  PAGEVIEWS.PAGEID "PAGEID",
  USERS.REGIONID "REGIONID",
  USERS.GENDER "GENDER"
FROM PAGEVIEWS PAGEVIEWS
LEFT OUTER JOIN USERS USERS ON ((PAGEVIEWS.USERID = USERS.USERID))
WHERE (USERS.GENDER = 'FEMALE')
```

Query ID: CSAS_PAGEVIEWS_FEMALE_3 **Sink:** PAGEVIEWS_FEMALE
Messages/sec: 148.61 **Sources:** PAGEVIEWS, USERS

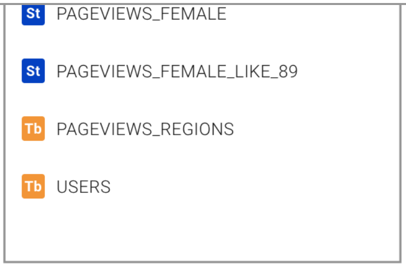
```
CREATE TABLE PAGEVIEWS_REGIONS WITH (KAFKA_TOPIC='PAGEVIEWS_REGIONS', PARTITIONS=1, REPLICAS=1) AS SELECT
  PAGEVIEWS_FEMALE.GENDER "GENDER",
  PAGEVIEWS_FEMALE.REGIONID "REGIONID",
  COUNT(*) "NUMUSERS"
FROM PAGEVIEWS_FEMALE PAGEVIEWS_FEMALE
```

(../_images/c3-ksql-persistent-query1.png)

6. Click **Editor**. On the right side of the page, find the **All available streams and tables** pane, which shows all of the streams and tables that you can access.

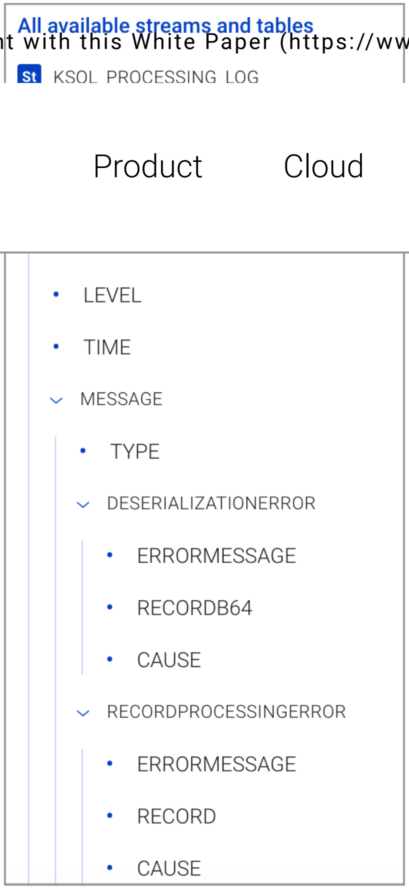
Expand Content

v5.4.0 v5.3.2 v5.3.1


PAGEVIEWS_FEMALE
PAGEVIEWS_FEMALE_LIKE_89
PAGEVIEWS_REGIONS
USERS

(../_images/c3-ksql-stream-table-view-1.png)

7. In the **All available streams and tables** section, click **KSQL_PROCESSING_LOG** to view the stream's schema, including nested data structures.



All available streams and tables

St KSQL_PROCESSING_LOG

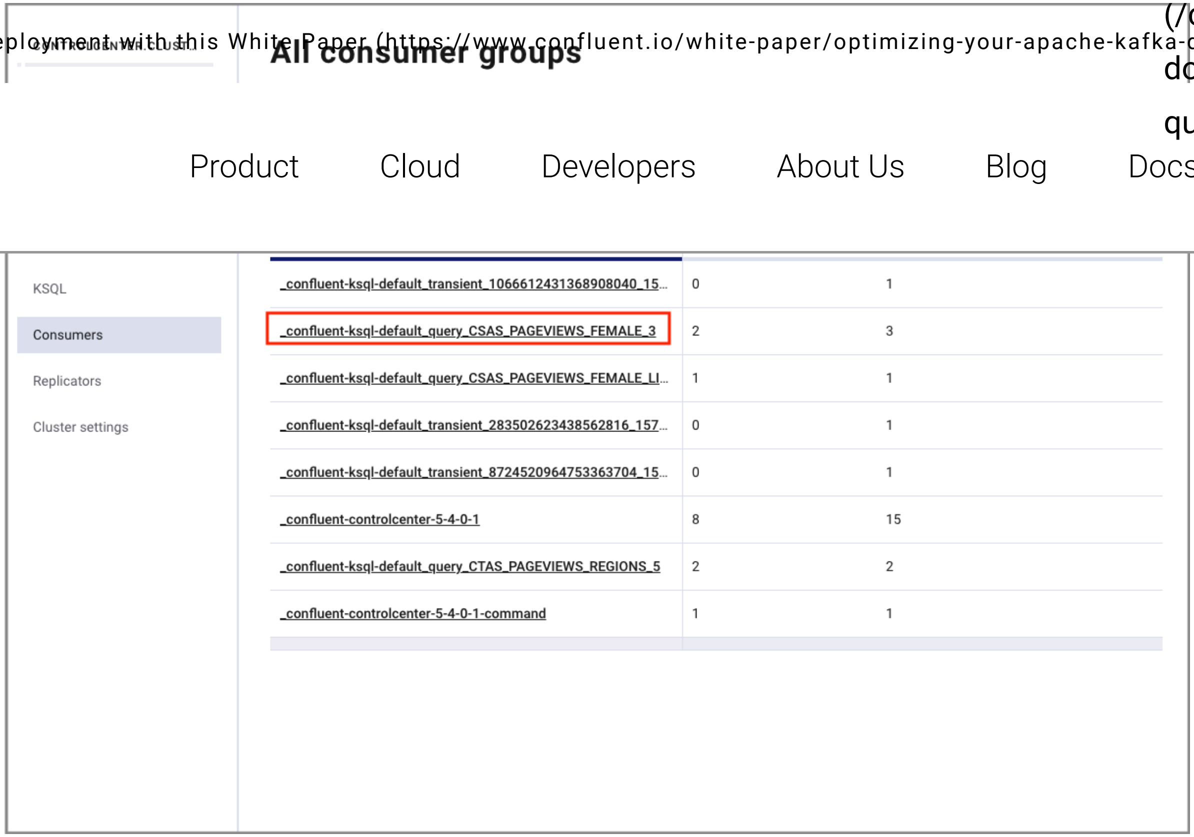
- LEVEL
- TIME
- MESSAGE
  - TYPE
  - DESERIALIZATIONERROR
    - ERRORMESSAGE
    - RECORDB64
    - CAUSE
  - RECORDPROCESSINGERROR
    - ERRORMESSAGE
    - RECORD
    - CAUSE

(../_images/c3-ksql-stream-table-view-2.png)

Step 5: Monitor Consumer Lag

Navigate to the **Consumers** tab to view the consumers created by KSQL.

Click the consumer group ID to view details for the `_confluent-ksql-default_query_CSAS_PAGEVIEWS_FEMALE_3` consumer group.

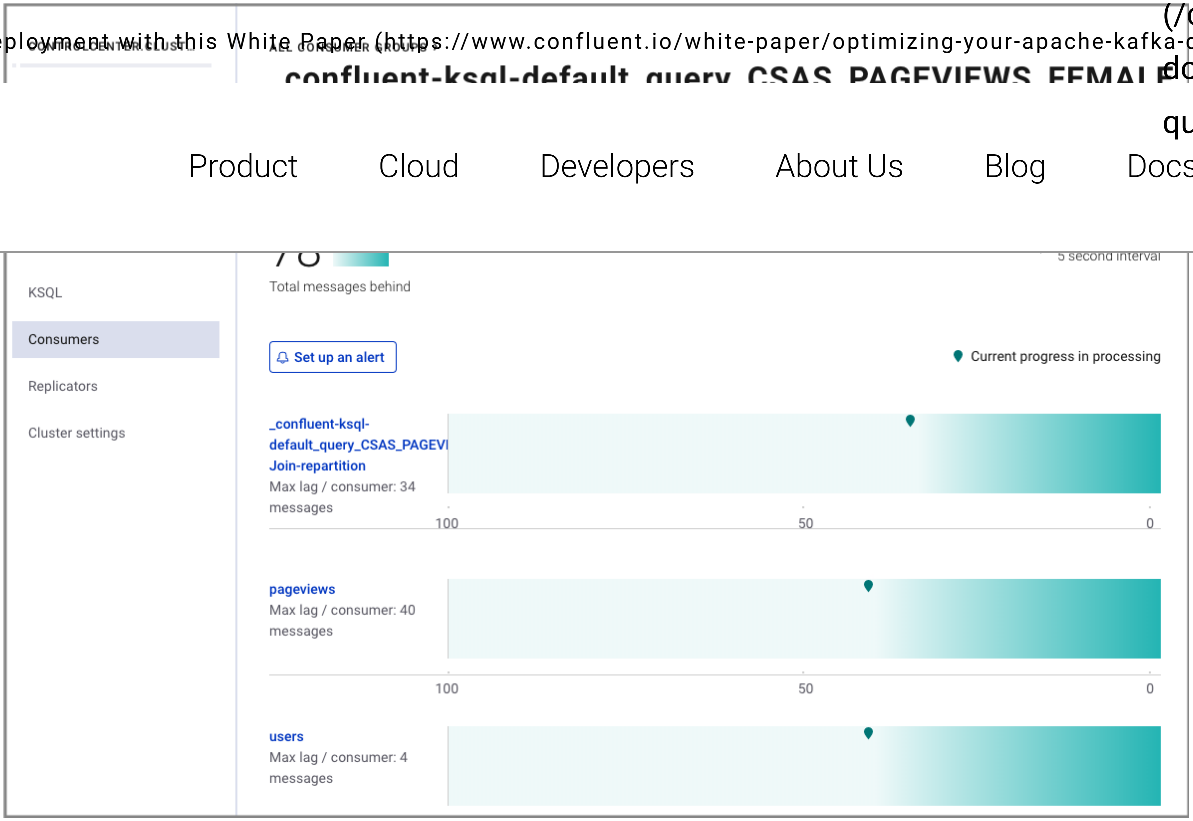


All consumer groups

_confluent-ksql-default_transient_1066612431368908040_15...	0	1
_confluent-ksql-default_query_CSAS_PAGEVIEWS_FEMALE_3	2	3
_confluent-ksql-default_query_CSAS_PAGEVIEWS_FEMALE_LI...	1	1
_confluent-ksql-default_transient_283502623438562816_157...	0	1
_confluent-ksql-default_transient_8724520964753363704_15...	0	1
_confluent-controlcenter-5-4-0-1	8	15
_confluent-ksql-default_query_CTAS_PAGEVIEWS_REGIONS_5	2	2
_confluent-controlcenter-5-4-0-1-command	1	1

(../_images/ksql-interface-monitor.png)

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


CONTROL CENTER CLUSTER
ALL CONSUMER GROUPS >
confluent-ksql-default_query_CSAS_PAGEVIEWS_FEMALE
78
Total messages behind
5 second interval
KSQL
Consumers
Replicators
Cluster settings
Set up an alert
Current progress in processing
_confluent-ksql-default_query_CSAS_PAGEVI Join-repartition
Max lag / consumer: 34 messages
100
50
0
pageviews
Max lag / consumer: 40 messages
100
50
0
users
Max lag / consumer: 4 messages

(../_images/ksql-interface-monitor-cnsmgp.png)

For more information, see the Control Center Consumers (../control-center/consumers.html#controlcenter-userguide-consumers) documentation.

Step 6: Stop Docker

When you are done working with Docker, you can stop and remove Docker containers and images.

Expand Content

v5.4.0 v5.3.2 v5.3.1

1. View a list of all Docker container IDs.

2. Run the following command to stop the Docker containers for Confluent:

```
docker container stop $(docker container ls -a -q -f "label=io.confluent.d
ocker")
```

3. Run the following commands to stop the containers and prune the Docker system. Running these commands deletes containers, networks, volumes, and images; freeing up disk space:

```
docker container stop $(docker container ls -a -q -f "label=io.confluent.d
ocker") && docker system prune -a -f --volumes
```

❶ Tip

Remove the filter label for Confluent Docker (`-f "label=io.confluent.docker"`) to clear all Docker containers from your system.

You can rebuild and restart the containers at any time using the `docker-compose up -d --build` command.

For more information, refer to the official Docker (https://docs.docker.com/) documentation.

Issue: Cannot locate the Datagen Connector

Resolution: Make sure to run the `docker-compose` command with the `--build` option:

```
docker-compose up -d --build
```

For details, see Step 1: Download and Start Confluent Platform Using Docker.

Resolution: Run the `build` command just for connect.

```
docker-compose build --no-cache connect
```

Your output should resemble:

```
Building connect
...
Completed
Removing intermediate container cdb0af3550c8
---> 36d00047d29b
Successfully built 36d00047d29b
Successfully tagged confluentinc/kafka-connect-datagen:latest
```

Resolution: Check the Connect log for `Datagen`.

```
connect  | [2019-04-17 20:03:26,137] INFO Loading plugin from: /usr/share/co
luent-hub-components/confluentinc-kafka-connect-datagen (org.apache.kafka.conn
ect.runtime.isolation.DelegatingClassLoader)
connect  | [2019-04-17 20:03:26,206] INFO Registered loader: PluginClassLoader
{pluginLocation=file:/usr/share/confluent-hub-components/confluentinc-kafka-co
nnect-datagen/} (org.apache.kafka.connect.runtime.isolation.DelegatingClassLoa
der)
connect  | [2019-04-17 20:03:26,206] INFO Added plugin 'io.confluent.kafka.con
nect.datagen.DatagenConnector' (org.apache.kafka.connect.runtime.isolation.Del
egatingClassLoader)
connect  | [2019-04-17 20:03:28,102] INFO Added aliases 'DatagenConnector' and
'Datagen' to plugin 'io.confluent.kafka.connect.datagen.DatagenConnector' (org
.apache.kafka.connect.runtime.isolation.DelegatingClassLoader)
```

Resolution: Check the Connect log for a warning and reminder to run the `docker-compose up -d --build` command properly.

```
docker-compose logs connect | grep -i Datagen
```

If the following warning is present, re-run the `docker-compose up -d --build` command:

```
connect  | WARNING: Did not find directory for kafka-connect-datagen (did yo
remember to run: docker-compose up -d --build ?)
```

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

```
docker-compose exec connect ls /usr/share/confluent-hub-components/confluent
c-kafka-connect-datagen/lib/
```

Expand Content
v5.4.0 v5.3.2 v5.3.1

Resolution: Verify the plugin exists in the connector path.

```
docker-compose exec connect bash -c 'echo $CONNECT_PLUGIN_PATH'
```

Your output should resemble:

```
/usr/share/java,/usr/share/confluent-hub-components
```

Confirm its contents are present:

```
docker-compose exec connect ls /usr/share/confluent-hub-components/confluentinc-kafka-connect-datagen
```

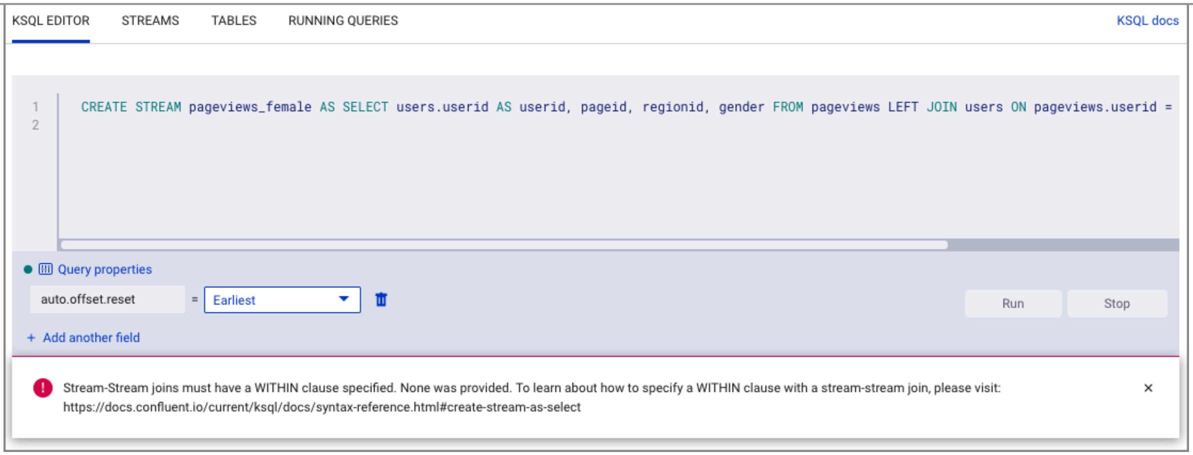
Your output should resemble:

```
assets   doc  etc  lib  manifest.json
```

Resolution: In **Kafka Connect > Setup Connection**, scroll down through the list of connectors to locate **DatagenConnector**; there are multiple connectors in the menu.

Issue: Stream-Stream joins error

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



```
CREATE STREAM pageviews_female AS SELECT users.userid AS userid, pageid, regionid, gender FROM pageviews LEFT JOIN users ON pageviews.userid =
```

auto.offset.reset = Earliest

Stream-Stream joins must have a WITHIN clause specified. None was provided. To learn about how to specify a WITHIN clause with a stream-stream join, please visit: https://docs.confluent.io/current/ksql/docs/syntax-reference.html#create-stream-as-select

(../_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 KSQL.

Issue: Unable to successfully complete KSQL query steps

Java errors or other severe errors were encountered.

Resolution: Ensure you are on an Operating System (../installation/versions-interoperability.html#operating-systems) currently supported by Confluent Platform.

Resolution: Ensure that the Docker memory was increased to 8 MB. Go to **Docker > Preferences > Advanced**. If Docker memory is insufficient, other unpredictable issues could occur.

Optimize your Apache Kafka Deployment with this White Paper (https://www.confluent.io/white-paper/optimizing-your-apache-kafka-deployment/)

(/current/quickstart/ce-docker-quickstart.html)
docker-docker-docker-
quickstart.html

(https://docs.confluent.io)

Product Cloud Developers About Us Blog Docs Download

Next Steps

Learn more about the components shown in this quick start:

- KSQL documentation (../ksql/docs/index.html#ksql-home) Learn about processing your data with KSQL for use cases such as streaming ETL, real-time monitoring, and anomaly detection. You can also learn how to use KSQL with this collection of scripted demos (https://github.com/confluentinc/examples).
- Stream Processing Cookbook (https://www.confluent.io/stream-processing-cookbook/) Try out in-depth KSQL tutorials and recommended deployment scenarios.
- Kafka Streams documentation (../streams/index.html#kafka-streams) Learn how to build stream processing applications in Java or Scala.
- Kafka Connect documentation (../connect/index.html#kafka-connect) Learn how to integrate Kafka with other systems and download ready-to-use connectors (https://www.confluent.io/product/connectors/) to easily ingest data in and out of Kafka in real-time.
- Kafka Clients documentation (../clients/index.html#kafka-clients) Learn how to read and write data to and from Kafka using programming languages such as Go, Python, .NET, C/C++.
- Videos, Demos, and Reading Material (../tutorials/index.html#tutorials) Try out the Confluent Platform tutorials and examples, watch demos and screencasts, and learn with white papers and blogs.

(http://www.apache.org/). All other trademarks, servicemarks, and copyrights are the property of their respective owners.

Please report any inaccuracies on this page or suggest an edit. (mailto:docs@confluent.io?subject=Documentation Feedback)

Last updated on Feb 26, 2020.

Expand Content

v5.4.0 v5.3.2 v5.3.1

Expand Content