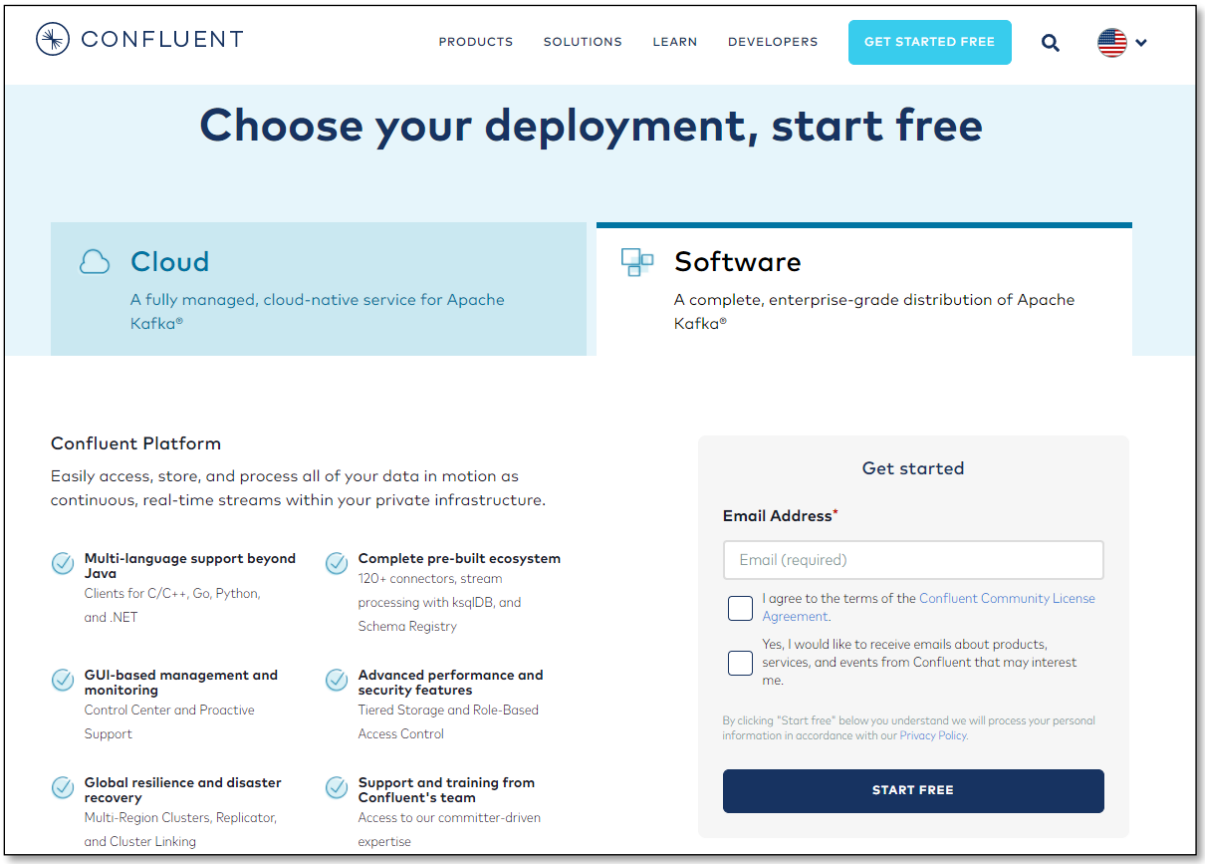
Confluent Kafka Community Edition

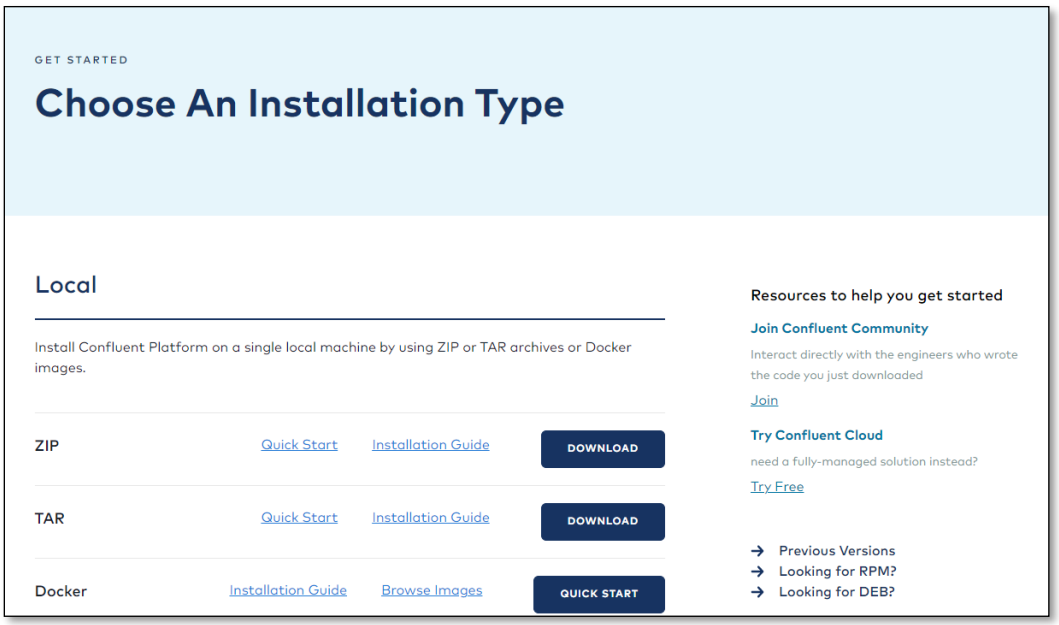
CONFLUENT

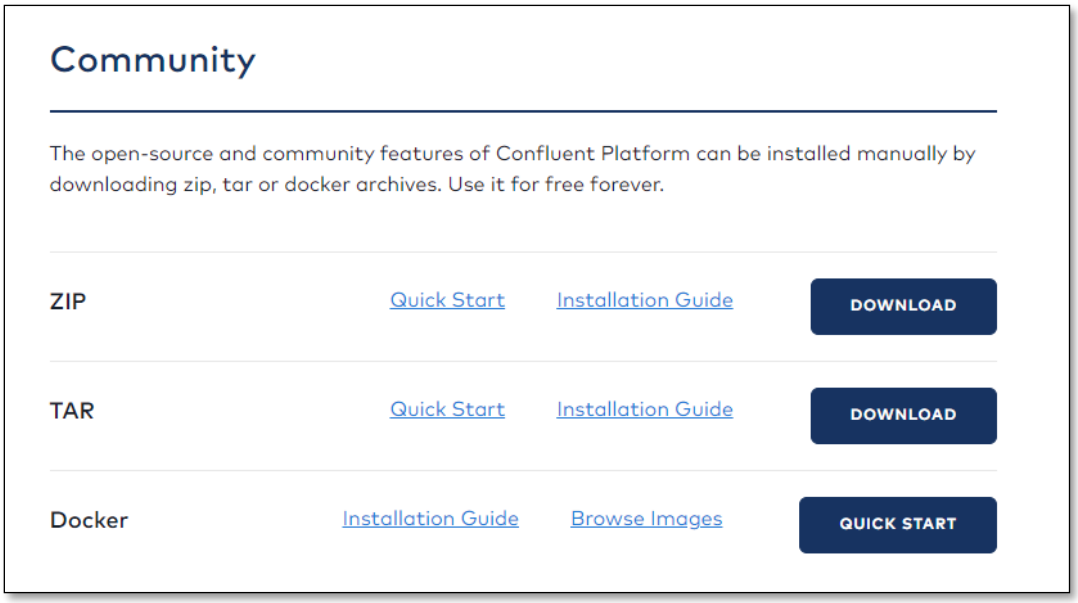
Hands-On Guide

https://www.confluent.io/get-started-v1/?product=software

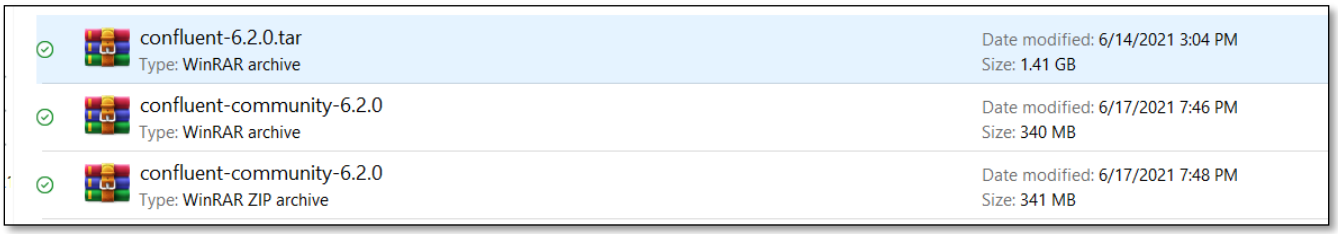


https://www.confluent.io/installation





https://docs.confluent.io/platform/current/quickstart/cos-quickstart.html?_ga=2.71798547.1843236294.1624254443-1839113997.1623662671&_gac=1.7015174.1624254443.EAIaIQobChMIhK7BpoOo8QIVSpNmAh1n3A3tEAAYASAAEgKyQ_D_BwE



Download the File in Linux or Windows

```
$ wget https://packages.confluent.io/archive/6.2/confluent-community-6.2.0.tar.gz
```

```
vagrant@master:~$ ls
bigdata                confluent-community-6.2.0.tar  dataset.zip  hs_err_pid2769.log  hs_err_pid3553.log  kafka-connect-files
confluent-6.2.0.tar.gz dataset                        hosts        hs_err_pid2956.log  hs_err_pid3897.log  replay_pid3553.log
vagrant@master:~$
```

vagrant@master:~$ tar -xzvf confluent-community-6.2.0.tar.gz

```
vagrant@master:~$ ls
bigdata                confluent-community-6.2.0.tar.gz
confluent-6.2.0        dataset
confluent-6.2.0.tar.gz dataset.zip
```

vagrant@master:~$ mv confluent-6.2.0 bigdata/confluent

vagrant@master:~$ vi .bashrc

```
export CONFLUENT_HOME=/home/vagrant/bigdata/confluent
export PATH=$PATH:$CONFLUENT_HOME/bin
```

vagrant@master:~$ **source .bashrc**

```
vagrant@master:~/bigdata/confluent/bin$ ls
connect-distributed          kafka-features                      kafka-rest-stop-service          ksql-server-stop
connect-mirror-maker         kafka-json-schema-console-consumer  kafka-run-class                  ksql-stop
connect-standalone           kafka-json-schema-console-producer  kafka-server-start               ksql-test-runner
kafka-acls                   kafka-leader-election               kafka-server-stop                schema-registry-run-class
kafka-avro-console-consumer  kafka-log-dirs                      kafka-storage                    schema-registry-start
kafka-avro-console-producer  kafka-metadata-shell                kafka-streams-application-reset  schema-registry-stop
kafka-broker-api-versions    kafka-mirror-maker                  kafka-topics                     schema-registry-stop-service
kafka-cluster                kafka-preferred-replica-election    kafka-verifiable-consumer        trogdor
kafka-configs                kafka-producer-perf-test            kafka-verifiable-producer        windows
kafka-console-consumer       kafka-protobuf-console-consumer     ksql                             zookeeper-security-migration
kafka-console-producer       kafka-protobuf-console-producer     ksql-datagen                     zookeeper-server-start
kafka-consumer-groups        kafka-reassign-partitions           ksql-migrations                  zookeeper-server-stop
kafka-consumer-perf-test     kafka-replica-verification          ksql-print-metrics               zookeeper-shell
kafka-delegation-tokens      kafka-rest-run-class                ksql-restore-command-topic
kafka-delete-records         kafka-rest-start                    ksql-run-class
kafka-dump-log               kafka-rest-stop                     ksql-server-start
```

Installing Confluent Hub Client - (Required for connecting to the Dataset Repository)

Download and unzip the Confluent Hub tarball.

1. Download and unzip confluent-hub-client-latest.tar.gz. Add the contents of the `bin` directory to your PATH environment variable so that `which confluent-hub` finds the `confluent-hub` command.

vagrant@master:~/bigdata/confluent/bin$ cd ..

vagrant@master:~/bigdata/confluent$ ls

bin etc lib README share src

vagrant@master:~/bigdata/confluent$ **wget** **http://client.hub.confluent.io/confluent-hub-client-latest.tar.gz**

```
vagrant@master:~/bigdata/confluent$ tar -xzvf confluent-hub-client-latest.tar.gz
```

if required : run the mv command

```
vagrant@master:~/bigdata/confluent$ mv confluent-hub bin/
```

```
vagrant@master:~/bigdata/confluent$ ls
bin  confluent-hub-client-latest.tar.gz  etc  lib  README  share  src
vagrant@master:~/bigdata/confluent$ ls bin/
confluent-hub                kafka-dump-log                      kafka-rest-stop
connect-distributed          kafka-features                      kafka-rest-stop-service
connect-mirror-maker         kafka-json-schema-console-consumer  kafka-run-class
```

2. Optional: Verify your installation by typing `confluent-hub` in your terminal.

```
confluent-hub
```

Your output should look like this:

```
usage: confluent-hub <command> [ <args> ]

Commands are:
    help      Display help information
    install   install a component from either Confluent Hub or from a local file

See 'confluent-hub help <command>' for more information on a specific command.
```

Install the Confluent CLI,

Install the Confluent CLI, `confluent`, using the following script.

On Microsoft Windows, an appropriate Linux environment may need to be installed in order to have the `curl` and `sh` commands available, such as the Windows Subsystem for Linux.

```
curl -L --http1.1 https://cnfl.io/cli | sh -s -- -b $CONFLUENT_HOME/bin
```

```
vagrant@master:~/bigdata/confluent$ curl -L --http1.1 https://cnfl.io/cli | sh -s -- -b $CONFLUENT_HOME/bin
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   162  100   162    0     0     32      0  0:00:05  0:00:05 --:--:--    46
  0 10699    0     0    0     0      0      0 --:--:--  0:00:14 --:--:--     0sh: 9: [[: not found
100 10699  100 10699    0     0    720      0  0:00:14  0:00:14 --:--:--  2620
confluentinc/cli info checking S3 for latest tag
confluentinc/cli info found version: latest for latest/linux/amd64
confluentinc/cli info NOTICE: see licenses located in /tmp/tmp.KCzfjzdOZq/confluent
confluentinc/cli info installed /home/vagrant/bigdata/confluent/bin/confluent
confluentinc/cli info please ensure /home/vagrant/bigdata/confluent/bin is in your PATH
```

Tarball or Zip installation for Confluent CLI (if curl command doesn't work)

Linux

```
wget https://s3-us-west-2.amazonaws.com/confluent.cloud/confluent-cli/archives/latest/confluent_latest_linux_amd64.tar.gz

vagrant@master:~$ tar xzvf confluent_latest_linux_amd64.tar.gz

vagrant@master:~$ mv confluent/* /home/vagrant/bigdata/confluent/bin/

vagrant@master:~$ ls confluent

vagrant@master:~$ rm -r confluent

vagrant@master:~$ ls bigdata/confluent/bin/
```

Windows

https://s3-us-west-2.amazonaws.com/confluent.cloud/confluent-cli/archives/latest/confluent_latest_windows_amd64.zip

```
vagrant@master:~/bigdata/confluent$ ls bin/
confluent                    kafka-delete-records          kafka-rest-start
confluent-hub                kafka-dump-log                kafka-rest-stop
```

Contents of Confluent Directories

Folder	Description
/bin/	Driver scripts for starting and stopping services
/etc/	Configuration files
/lib/	System services
/logs/	Log files
/share/	Jars and licenses
/src/	Source files that require a platform-dependent build

Start Confluent Platform

Start Confluent Platform using the `confluent local services start` command.

This command will start all of the Confluent Platform components, including Kafka, ZooKeeper, Schema Registry, HTTP REST Proxy for Kafka, Kafka Connect, and ksqlDB.

Note : Schema Registry
Schema Registry is a central repository with a RESTful interface for developers to define standard schemas and register applications to enable compatibility. Schema Registry is available as a software component of Confluent Platform or as a managed component of Confluent Cloud.

```
vagrant@master:~$ cd
```

```
vagrant@master:~$ confluent local services start
```

```
The local commands are intended for a single-node development environment only,
NOT for production usage. https://docs.confluent.io/current/cli/index.html

Using CONFLUENT_CURRENT: /tmp/confluent.280176
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 ksqlDB Server
ksqlDB Server is [UP]
```

```
vagrant@master:~$ jps
```

```
3316 SchemaRegistryMain
3397 KafkaRestMain
3621 Jps
3446 ConnectDistributed
3224 Kafka
3161 QuorumPeerMain
3550 KsqlServerMain
```

Please note that "Control Center" is not available in Confluent Community Edition

Install the Kafka Connect Datagen source connector

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
```

```
Running in a "--no-prompt" mode
Implicit acceptance of the license below:
Apache License 2.0
https://www.apache.org/licenses/LICENSE-2.0
Downloading component Kafka Connect Datagen 0.5.0, provided by Confluent, Inc. from Confluent Hub and installing into /home/vagrant/
bigdata/confluent/share/confluent-hub-components
Adding installation directory to plugin path in the following files:
  /home/vagrant/bigdata/confluent/etc/kafka/connect-distributed.properties
  /home/vagrant/bigdata/confluent/etc/kafka/connect-standalone.properties
  /home/vagrant/bigdata/confluent/etc/schema-registry/connect-avro-distributed.properties
  /home/vagrant/bigdata/confluent/etc/schema-registry/connect-avro-standalone.properties
  /tmp/confluent.280176/connect/connect.properties
  /tmp/confluent.280176/connect/connect.properties

Completed
```

Create Kafka Topics

In this step, you create Kafka topics using the Kafka CLI.

1. Create a topic named `users`:

```
kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1
--topic users
```

2. Create a topic named `pageviews`:

```
kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions
1 --topic pageviews
```

$ kafka-topics --list --bootstrap-server localhost:9092

```
__consumer_offsets
__transaction_state
_confluent-ksql-default__command_topic
_confluent_balancer_partition_samples
_schemas
connect-configs
connect-offsets
connect-statuses
default_ksql_processing_log
pageviews
users
```

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`.

1. Run the first instance of the Kafka Connect Datagen connector to produce Kafka data to the `pageviews` topic in AVRO format.
2. **Note : Avro** is a row-oriented remote procedure call and data serialization framework developed within Apache's Hadoop project. It uses JSON for defining data types and protocols, and serializes data in a compact binary **format**. ... **Avro** uses a schema to structure the data that is being encoded.

```
curl -L -O -H 'Accept: application/vnd.github.v3.raw' https://api.github.com/repos/confluentinc/kafka-connect-datagen/contents/config/connector_pageviews_cos.config
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   341  100   341    0     0    394      0 --:--:-- --:--:-- --:--:--   394
```

RESTART THE CONFLUENT SERVICES

```
curl -X POST -H "Content-Type: application/json" --data @connector_pageviews_cos.config http://localhost:8083/connectors
```

```
{"name":"datagen-pageviews","config":{"connector.class":"io.confluent.kafka.connect.datagen.DatagenConnector","key.converter":"org.apache.kafka.connect.storage.StringConverter","kafka.topic":"pageviews","quickstart":"pageviews","max.interval":"100","iterations":"10000000","tasks.max":"1","name":"datagen-pageviews"},"tasks":[],"type":"source"}vagrant@master:~$
```

3. Run the second instance of the Kafka Connect Datagen connector to produce Kafka data to the `users` topic in AVRO format.

```
curl -L -O -H 'Accept: application/vnd.github.v3.raw' https://api.github.com/repos/confluentinc/kafka-connect-datagen/contents/config/connector_users_cos.config
```

RESTART THE CONFLUENT SERVICES

```
curl -X POST -H "Content-Type: application/json" --data @connector_users_cos.config http://localhost:8083/connectors
```

```
{"name":"datagen-users","config":{"connector.class":"io.confluent.kafka.connect.datagen.DatagenConnector","key.converter":"org.apache.kafka.connect.storage.StringConverter","kafka.topic":"users","quickstart":"users","max.interval":"1000","iterations":"10000000","tasks.max":"1","name":"datagen-users"},"tasks":[],"type":"source"}vagrant@master:~$
```

Get the data in JSON Pretty Format

vagrant@master:~$ sudo apt-get install jq

Check the "pageviews" topic data

```
kafka-console-consumer.sh --topic pageviews --from-beginning --bootstrap-server localhost:9092
```

```
User_1Page_96
User_4Page_93
User_2Page_30
User_5Page_97
```

Check the "users" topic data

```
kafka-console-consumer.sh --topic users --from-beginning --bootstrap-server localhost:9092
```

```
User_1Region_2
OTHER
User_4Region_5
FEMALE
User_3Region_MALE
User_7Region_9
FEMALE
```

Create and Write to a Stream and Table using ksqlDB

In this step, you create streams, tables, and queries using ksqlDB SQL.

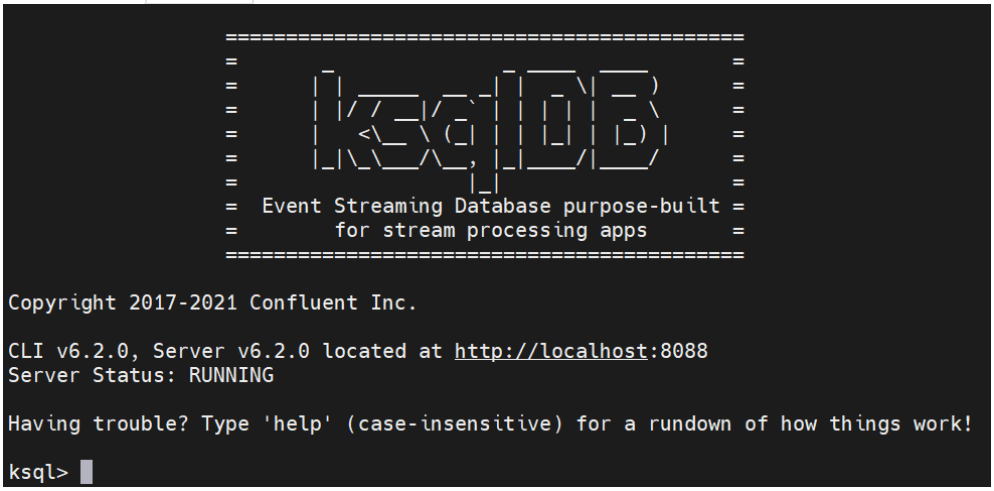
Create Streams and Tables

1. Start the ksqlDB CLI in your terminal with this command.

```
LOG_DIR=$CONFLUENT_HOME/ksql_logs ksql
```

Important

By default ksqlDB attempts to store its logs in a directory called `logs` that is relative to the location of the `ksql` executable. For example, if `ksql` is installed at `/usr/local/bin/ksql`, then it would attempt to store its logs in `/usr/local/logs`. If you are running `ksql` from the default Confluent Platform location, `$CONFLUENT_HOME/bin`, you must override this default behavior by using the `LOG_DIR` variable.



2. Create a stream `PAGEVIEWS` from the Kafka topic `pageviews`, specifying the `value_format` of `AVRO`:

```
CREATE STREAM pageviews WITH (KAFKA_TOPIC='pageviews', VALUE_FORMAT='AVRO');
```

```
ksql> show streams;
```

```
ksql> show streams;

 Stream Name         | Kafka Topic                 | Key Format | Value Format | Windowed
-----------------------------------------------------------------------------------------
 KSQL_PROCESSING_LOG | default_ksql_processing_log | KAFKA      | JSON         | false
 PAGEVIEWS           | pageviews                   | KAFKA      | AVRO         | false
-----------------------------------------------------------------------------------------
```

3. Create a table `USERS` with several columns from the Kafka topic `users`, with the `value_format` of `AVRO`:

```
CREATE TABLE users (id VARCHAR PRIMARY KEY) WITH (KAFKA_TOPIC='users', VALUE_FORMAT='AVRO');
```

```
ksql> show tables;

 Table Name | Kafka Topic | Key Format | Value Format | Windowed
-------------------------------------------------------------------
 USERS      | users       | KAFKA      | AVRO         | false
-------------------------------------------------------------------
```

Note : A **stream** can be considered a changelog **of** a **table**, as the aggregation **of** a **stream of** updates over time yields a **table**. A **table** can be considered a snapshot, at a point in time, **of** the latest value **for** each key **in a stream** (a **stream's** data records are key-value pairs).

Write Queries

In this step, you run ksqlDB SQL queries.

1. Set the `auto.offset.reset` query property to ``earliest`.

   This instructs ksqlDB queries to read all available topic data from the beginning.

   This configuration is used for each subsequent query.

   For more information, see the [ksqlDB Configuration Parameter Reference](#).

```
SET 'auto.offset.reset'='earliest';
```

```
Successfully changed local property 'auto.offset.reset' to 'earliest'. Use the UNSET command to revert your change.
```

2. 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:

```
Page_45
Page_38
Page_11
LIMIT reached
Query terminated
```

3. 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` Stream :

```
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;
```

```
 Message
----------------------------------------------
 Created query with ID CSAS_PAGEVIEWS_FEMALE_5
----------------------------------------------
```

select * from pageviews_female emit changes limit 10;

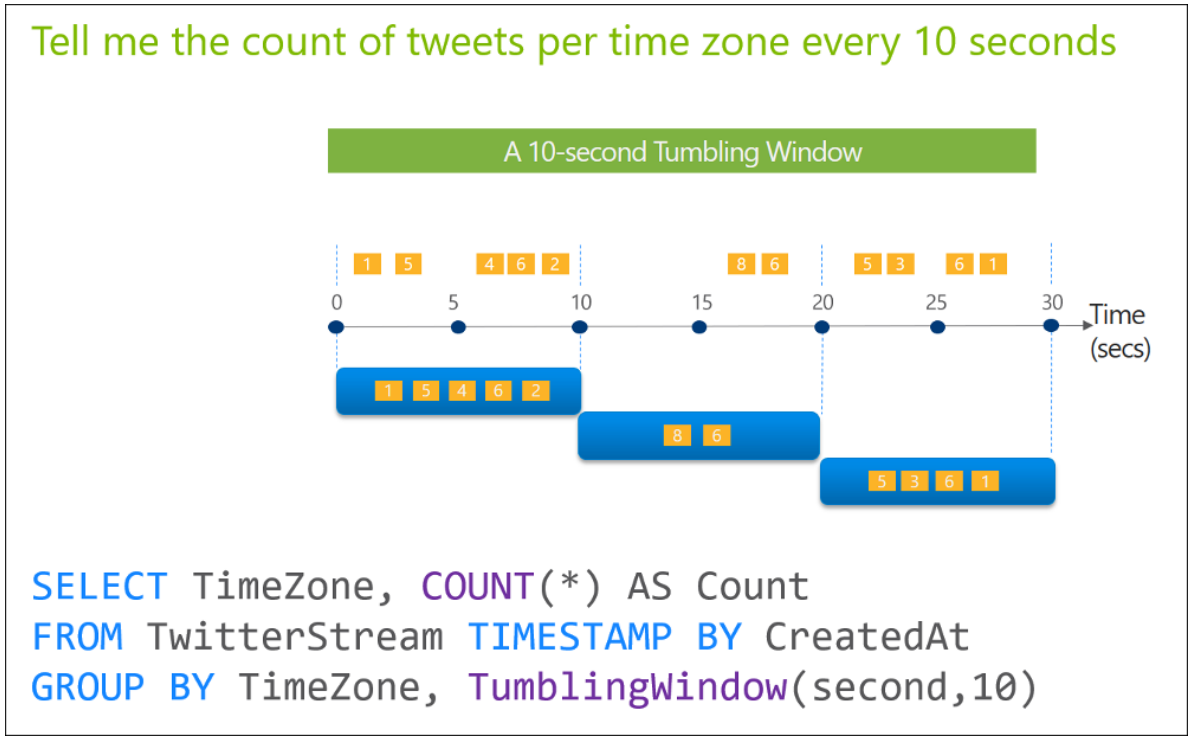
4. Create a persistent query where `REGIONID` ends with `8` or `9`. Results from this query are written to the Kafka Stream named `pageviews_enriched_r8_r9` as explicitly specified in the query:

```
CREATE STREAM pageviews_female_like_89 WITH (KAFKA_TOPIC='pageviews_enriched_r8_r9', value_f
ormat='AVRO') AS SELECT * FROM pageviews_female WHERE regionid LIKE '%_8' OR regionid LIKE '
%_9' EMIT CHANGES;
```

```
 Message
-------------------------------------------------------
 Created query with ID CSAS_PAGEVIEWS_FEMALE_LIKE_89_7
-------------------------------------------------------
```

select * from pageviews_female_like_89 emit changes;

Tumbling windows : are a series of fixed-sized, non-overlapping and contiguous time intervals. The following diagram illustrates a stream with a series of events and how they are mapped into 10-second tumbling windows.



5. 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 Table called `PAGEVIEWS_REGIONS`:

```
CREATE TABLE pageviews_regions WITH (KEY_FORMAT='JSON') \
  AS SELECT gender, regionid , COUNT(*) AS numbers \
  FROM pageviews LEFT JOIN users ON pageviews.userid = users.id \
  WINDOW TUMBLING (SIZE 30 SECOND) \
  GROUP BY gender, regionid \
  HAVING COUNT(*) > 1
  EMIT CHANGES;
```

```
 Message
----------------------------------------------
 Created query with ID CTAS_PAGEVIEWS_REGIONS_9
----------------------------------------------
```

select * from PAGEVIEWS_REGIONS emit changes;

Examine Streams, Tables, and Queries

- List the streams:

```
SHOW STREAMS;
```

```
Stream Name              | Kafka Topic                 | Key Format | Value Format | Windowed
-----------------------------------------------------------------------------------------------
KSQL_PROCESSING_LOG      | default_ksql_processing_log | KAFKA      | JSON         | false
PAGEVIEWS                | pageviews                   | KAFKA      | AVRO         | false
PAGEVIEWS_FEMALE         | PAGEVIEWS_FEMALE            | KAFKA      | AVRO         | false
PAGEVIEWS_FEMALE_LIKE_89 | pageviews_enriched_r8_r9    | KAFKA      | AVRO         | false
-----------------------------------------------------------------------------------------------
```

- List the tables:

```
SHOW TABLES;
```

```
Table Name        | Kafka Topic       | Key Format | Value Format | Windowed
-------------------------------------------------------------------------------
PAGEVIEWS_REGIONS | PAGEVIEWS_REGIONS | JSON       | AVRO         | true
USERS             | users             | KAFKA      | AVRO         | false
-------------------------------------------------------------------------------
```

- View the details of a stream or a table:

```
DESCRIBE users EXTENDED;
```

For example, to view the details of the `users` table:

```
DESCRIBE USERS EXTENDED;
```

- **List the running queries:**

```
SHOW QUERIES;
```

```
 Query ID                       | Query Type | Status    | Sink Name               | Sink Kafka Topic         | Query String


-----------------------------------------------------------------------------------------------------------------------------------------------------
 CSAS_PAGEVIEWS_FEMALE_5         | PERSISTENT | RUNNING:1 | PAGEVIEWS_FEMALE        | PAGEVIEWS_FEMALE         | CREATE STREAM PAGE
VIEWS_FEMALE WITH (KAFKA_TOPIC='PAGEVIEWS_FEMALE', PARTITIONS=1, REPLICAS=1) AS SELECT   USERS.ID USERID,   PAGEVIEWS.PAGEID PAGEID,
   USERS.REGIONID REGIONID FROM PAGEVIEWS PAGEVIEWS LEFT OUTER JOIN USERS USERS ON ((PAGEVIEWS.USERID = USERS.ID)) WHERE (USERS.GEND
ER = 'FEMALE') EMIT CHANGES;
 CSAS_PAGEVIEWS_FEMALE_LIKE_89_7 | PERSISTENT | RUNNING:1 | PAGEVIEWS_FEMALE_LIKE_89 | pageviews_enriched_r8_r9 | CREATE STREAM PAGE
VIEWS_FEMALE_LIKE_89 WITH (KAFKA_TOPIC='pageviews_enriched_r8_r9', PARTITIONS=1, REPLICAS=1, VALUE_FORMAT='AVRO') AS SELECT * FROM P
AGEVIEWS_FEMALE PAGEVIEWS_FEMALE WHERE ((PAGEVIEWS_FEMALE.REGIONID LIKE '%_8') OR (PAGEVIEWS_FEMALE.REGIONID LIKE '%_9')) EMIT CHANG
ES;
 CTAS_PAGEVIEWS_REGIONS_9        | PERSISTENT | RUNNING:1 | PAGEVIEWS_REGIONS       | PAGEVIEWS_REGIONS        | CREATE TABLE PAGEV
IEWS_REGIONS WITH (KAFKA_TOPIC='PAGEVIEWS_REGIONS', KEY_FORMAT='JSON', PARTITIONS=1, REPLICAS=1) AS SELECT   USERS.GENDER GENDER,
USERS.REGIONID REGIONID,   COUNT(*) NUMBERS FROM PAGEVIEWS PAGEVIEWS LEFT OUTER JOIN USERS USERS ON ((PAGEVIEWS.USERID = USERS.ID))
WINDOW TUMBLING ( SIZE 30 SECONDS )  GROUP BY USERS.GENDER, USERS.REGIONID HAVING (COUNT(*) > 1) EMIT CHANGES;
-----------------------------------------------------------------------------------------------------------------------------------------------------
For detailed information on a Query run: EXPLAIN <Query ID>;
```

- Review the query execution plan:

  Get a Query ID from the output of `SHOW QUERIES` and run `EXPLAIN` to view the query execution plan for the Query ID:

```
EXPLAIN <Query ID>;
```

ksql> explain CSAS_PAGEVIEWS_FEMALE_5;

Monitor Streaming Data

Now you can monitor the running queries created as streams or tables.

- The following query returns the page view information of female users:

```
SELECT * FROM pageviews_female EMIT CHANGES LIMIT 5;
```

```
+-----------------------------------------+-----------------------------------------+-------------
|USERID                                   |PAGEID                                   |REGIONID
+-----------------------------------------+-----------------------------------------+-------------
|User_8                                   |Page_82                                  |Region_3
|User_8                                   |Page_76                                  |Region_3
|User_2                                   |Page_75                                  |Region_5
|User_8                                   |Page_98                                  |Region_3
|User_8                                   |Page_22                                  |Region_3
Limit Reached
Query terminated
```

- The following query returns the page view information of female users in the regions whose `regionid` ends with `8` or `9`:

```
SELECT * FROM pageviews_female_like_89 EMIT CHANGES LIMIT 5;
```

```
+-----------------------------------------+-----------------------------------------+----------
|USERID                                   |PAGEID                                   |REGIONID
+-----------------------------------------+-----------------------------------------+----------
|User_3                                   |Page_24                                  |Region_8
|User_3                                   |Page_41                                  |Region_8
|User_3                                   |Page_20                                  |Region_8
|User_3                                   |Page_44                                  |Region_8
|User_3                                   |Page_87                                  |Region_8
Limit Reached
Query terminated
```

- The following query returns the page view counts for each region and gender combination in a tumbling window of 30 seconds. To see table updates, let the query run for a few seconds. Press Ctrl+C to stop the query.

```
SELECT * FROM pageviews_regions EMIT CHANGES;
```

```
+-----------------------+-----------------------+-----------------------+-----------------------+-----------
|GENDER                 |REGIONID               |WINDOWSTART            |WINDOWEND              |NUMBERS
+-----------------------+-----------------------+-----------------------+-----------------------+-----------
|OTHER                  |Region_4               |1624277880000          |1624277910000          |2
|OTHER                  |Region_7               |1624277880000          |1624277910000          |2
|MALE                   |Region_8               |1624277910000          |1624277940000          |16
|OTHER                  |Region_4               |1624277910000          |1624277940000          |38
|OTHER                  |Region_7               |1624277910000          |1624277940000          |42
|MALE                   |Region_6               |1624277910000          |1624277940000          |13
|OTHER                  |Region_2               |1624277910000          |1624277940000          |17
|OTHER                  |Region_1               |1624277910000          |1624277940000          |61
```

TERMINATE Running Queries

```
ksql> terminate CSAS_PAGEVIEWS_FEMALE_5;
```

```
 Message
--------------------
 Query terminated.
--------------------
```

```
ksql> show queries;
```

```
 Query ID                        | Query Type | Status    | Sink Name                | Sink Kafka Topic         | Query String

------------------------------------------------------------------------------------------------------------------------------------------------------------
 CSAS_PAGEVIEWS_FEMALE_LIKE_89_7 | PERSISTENT | RUNNING:1 | PAGEVIEWS_FEMALE_LIKE_89 | pageviews_enriched_r8_r9 | CREATE STREAM PAGE
VIEWS_FEMALE_LIKE_89 WITH (KAFKA_TOPIC='pageviews_enriched_r8_r9', PARTITIONS=1, REPLICAS=1, VALUE_FORMAT='AVRO') AS SELECT * FROM P
AGEVIEWS_FEMALE PAGEVIEWS_FEMALE WHERE ((PAGEVIEWS_FEMALE.REGIONID LIKE '%_8') OR (PAGEVIEWS_FEMALE.REGIONID LIKE '%_9')) EMIT CHANG
ES;
 CTAS_PAGEVIEWS_REGIONS_9        | PERSISTENT | RUNNING:1 | PAGEVIEWS_REGIONS        | PAGEVIEWS_REGIONS        | CREATE TABLE PAGEV
IEWS_REGIONS WITH (KAFKA_TOPIC='PAGEVIEWS_REGIONS', KEY_FORMAT='JSON', PARTITIONS=1, REPLICAS=1) AS SELECT   USERS.GENDER GENDER,
USERS.REGIONID REGIONID,   COUNT(*) NUMBERS FROM PAGEVIEWS PAGEVIEWS LEFT OUTER JOIN USERS USERS ON ((PAGEVIEWS.USERID = USERS.ID))
WINDOW TUMBLING ( SIZE 30 SECONDS )  GROUP BY USERS.GENDER, USERS.REGIONID HAVING (COUNT(*) > 1) EMIT CHANGES;
------------------------------------------------------------------------------------------------------------------------------------------------------------
For detailed information on a Query run: EXPLAIN <Query ID>;
```

```
ksql> exit

Exiting ksqlDB.

vagrant@master:~$
```

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
```

```
The local commands are intended for a single-node development environment only,
NOT for production usage. https://docs.confluent.io/current/cli/index.html

Using CONFLUENT_CURRENT: /tmp/confluent.280176
Stopping ksqlDB Server
ksqlDB Server is [DOWN]
Stopping Connect
Connect is [DOWN]
Stopping Kafka REST
Kafka REST is [DOWN]
Stopping Schema Registry
Schema Registry is [DOWN]
Stopping Kafka
Kafka is [DOWN]
Stopping ZooKeeper
ZooKeeper is [DOWN]
Deleting: /tmp/confluent.280176
vagrant@master:~$ confluent local destroy
The local commands are intended for a single-node development environment only,
NOT for production usage. https://docs.confluent.io/current/cli/index.html
```

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

```
confluent local destroy
```

You can start the local install of Confluent Platform again with the confluent local services start command.

vagrant@master:~$ jps

Confluent Control Centre – WebUI : reference

http://192.168.56.70:9021

https://docs.ksqldb.io/en/0.7.1-ksqldb/tutorials/basics-control-center/

Start Hadoop Services

vagrant@master:~$ start-dfs.sh

```
Starting namenodes on [master]
master: starting namenode, logging to /home/vagrant/bigdata/hadoop/logs/hadoop-vagrant-namenode-master.out
master: starting datanode, logging to /home/vagrant/bigdata/hadoop/logs/hadoop-vagrant-datanode-master.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/vagrant/bigdata/hadoop/logs/hadoop-vagrant-secondarynamenode-master.out
```

vagrant@master:~$ start-yarn.sh

```
starting yarn daemons
starting resourcemanager, logging to /home/vagrant/bigdata/hadoop/logs/yarn-vagrant-resourcemanager-master.out
master: starting nodemanager, logging to /home/vagrant/bigdata/hadoop/logs/yarn-vagrant-nodemanager-master.out
```

vagrant@master:~$ jps

```
5024 QuorumPeerMain
6529 Jps
5188 SchemaRegistryMain
5669 NameNode
5401 KsqlServerMain
5835 DataNode
6108 SecondaryNameNode
5310 ConnectDistributed
5262 KafkaRestMain
5087 Kafka
6431 NodeManager
```

Confluent Platform properties files

The following is a list of the default Confluent Platform services configuration properties files, where `$CONFLUENT_HOME` is the directory where you installed Confluent Platform. You reference or modify the appropriate file when you work with a Confluent Platform service.

- Connect: `$CONFLUENT_HOME/etc/schema-registry/connect-avro-distributed.properties`
- Control Center: `$CONFLUENT_HOME/etc/confluent-control-center/control-center-dev.properties`
- Kafka: `$CONFLUENT_HOME/etc/kafka/server.properties`
- REST Proxy: `$CONFLUENT_HOME/etc/kafka-rest/kafka-rest.properties`
- ksqlDB: `$CONFLUENT_HOME/etc/ksqldb/ksql-server.properties`
- Schema Registry: `$CONFLUENT_HOME/etc/schema-registry/schema-registry.properties`
- ZooKeeper: `$CONFLUENT_HOME/etc/kafka/zookeeper.properties`