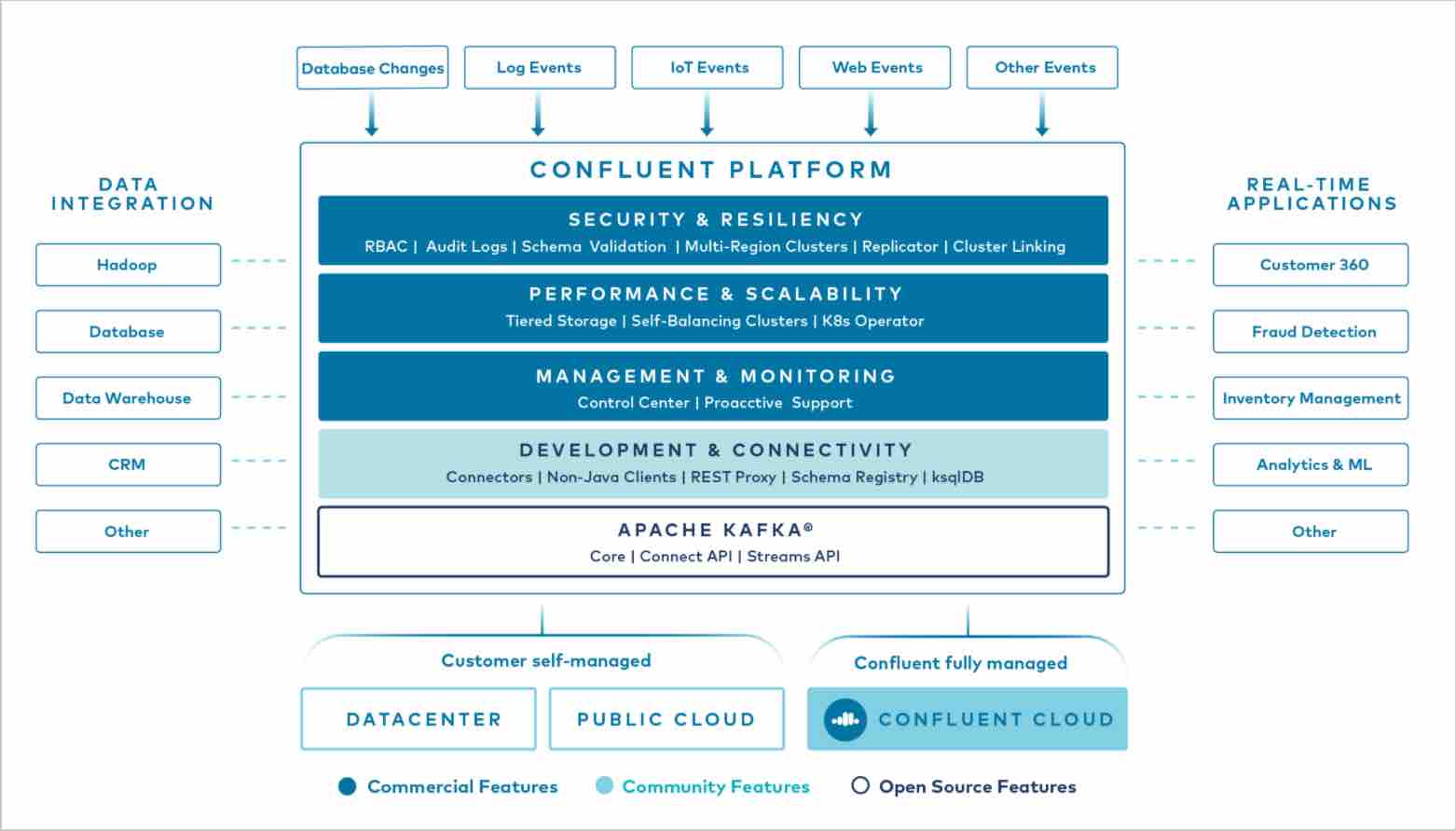
**Confluent Kafka KSQL on Topic**

**Overview**

Confluent Platform is a full-scale data streaming platform that enables you to easily access, store, and manage both data from many different sources with one reliable, high performance system and data as continuous, real-time streams. Built by the original creators of Apache Kafka®, Confluent expands the benefits of Kafka with enterprise-grade features while removing the burden of Kafka management or monitoring.

* ksqlDB is a database purpose-built to help developers create stream processing applications on top of Apache Kafka.
  + KSTREAM: An abstraction of a record stream, where each data record represents a self-contained datum in the unbounded data set.
  + KTABLE: An abstraction of a changelog stream, where each data record represents an update. More precisely, the value in a data record is interpreted as an “UPSERT”
* Manage key operations and monitor the health and performance of Kafka clusters using curated dashboards
* Schema Registry provides a RESTful interface for storing and retrieving your Avro®, JSON Schema, and Protobuf schemas.
* Kafka Connect is a framework for connecting Apache Kafka® with external systems such as databases, key-value stores, search indexes, and file systems.
* Use connectors to copy data between Apache Kafka® and other systems that you want to pull data from or push data to.

[](file:///Users/nnagaraju/sleeve/hcsc/docker-demo/ksql-demo/cp-00.jpg)

**Use case Scenario with Example**

* In this use case a sample json was ingested into Kafka using streaming fashion. A KStream is built on top of Kafka Topic to aggregate or filter or enrich messages and enriched messages saved as Kafka topic by KStream feature. The data from enriched topic ingested into Elasticsearch index. Finally, Kibana presents the insigths in dashboards using wizards and displays in live fashion.

**JSON Data for KSQL**

{"bot":true,"domain":"www.wikidata.org","dt":1626573055000,"id":" 5a8e8d9c-257d-4e2c-9654-d47c3db24d96","length":{"new":4517,"old":null},"request\_id":"fd2371b6-71fe-48a8- ad5a-d56db0ed9c10","stream":"mediawiki.recentchange","uri":"https://www. wikidata.org/wiki/Q107561662","minor":false,"server\_name":"www.wikidata.org"," server\_url":"https://www.wikidata.org","timestamp":1626573055,"title":" Q107561662","type":"edit","user":"SuccuBot","wiki":"wikidatawiki"}

**Load JSON Data into Topic**

* Either use Kafka Connector or Spark Streaming to load data into Kafka Topic wikipedia.parsed

{

"name": "wikipedia-sse",

"config": {

"connector.class": "com.github.cjmatta.kafka.connect.sse.ServerSentEventsSourceConnector",

"sse.uri": "https://stream.wikimedia.org/v2/stream/recentchange",

"topic": "wikipedia.parsed",

"transforms": "extractData, parseJSON",

"transforms.extractData.type": "org.apache.kafka.connect.transforms.ExtractField\$Value",

"transforms.extractData.field": "data",

"transforms.parseJSON.type": "com.github.jcustenborder.kafka.connect.json.FromJson\$Value",

"value.converter": "io.confluent.connect.avro.AvroConverter"

}

}

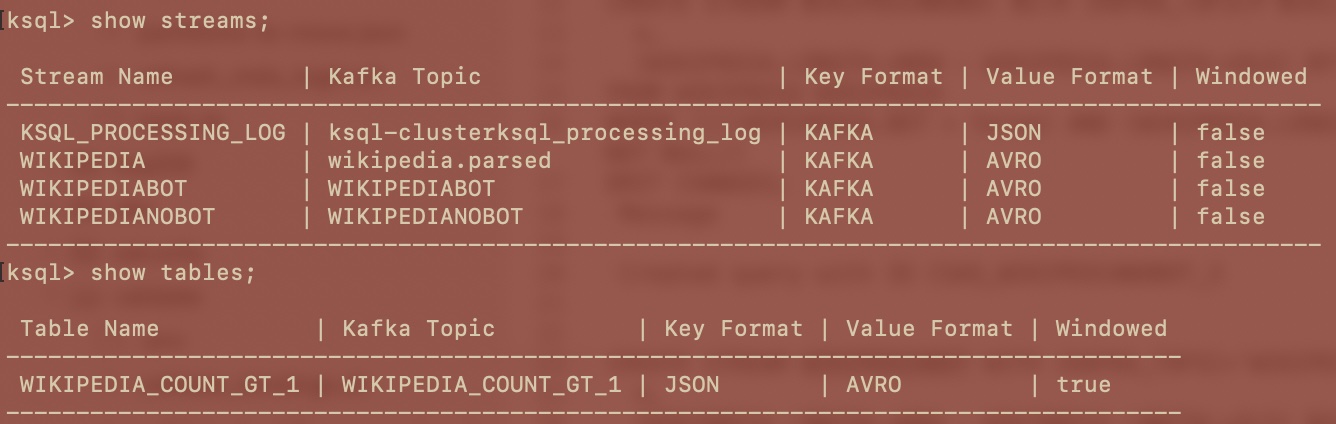
**Create KStream on Topic**

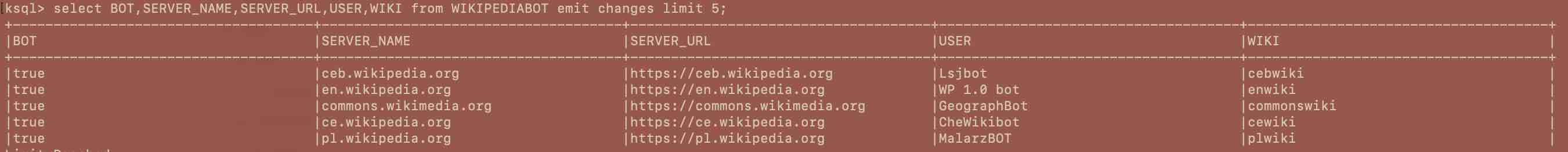
ksql> CREATE STREAM WIKIPEDIA (BOT BOOLEAN, COMMENT STRING, ID BIGINT, LENGTH STRUCT<NEW BIGINT, OLD BIGINT>, LOG\_ACTION STRING, LOG\_ACTION\_COMMENT STRING, LOG\_ID BIGINT, LOG\_TYPE STRING, META STRUCT<DOMAIN STRING, DT BIGINT, ID STRING, REQUEST\_ID STRING, `STREAM` STRING, URI STRING>, MINOR BOOLEAN, `NAMESPACE` BIGINT, PARSEDCOMMENT STRING, PATROLLED BOOLEAN, REVISION STRUCT<NEW BIGINT, OLD BIGINT>, SERVER\_NAME STRING, SERVER\_SCRIPT\_PATH STRING, SERVER\_URL STRING, TIMESTAMP BIGINT, TITLE STRING, TYPE STRING, USER STRING, WIKI STRING) WITH (KAFKA\_TOPIC='wikipedia.parsed', KEY\_FORMAT='KAFKA', VALUE\_FORMAT='AVRO', VALUE\_SCHEMA\_ID=1);

ksql> CREATE STREAM WIKIPEDIABOT WITH (KAFKA\_TOPIC='WIKIPEDIABOT') AS SELECT \* FROM WIKIPEDIA WIKIPEDIA

WHERE ((((WIKIPEDIA.BOT = true) AND (WIKIPEDIA.LENGTH IS NOT NULL)) AND (WIKIPEDIA.LENGTH->NEW IS NOT NULL)) AND (WIKIPEDIA.LENGTH->OLD IS NOT NULL))

EMIT CHANGES;

[](file:///Users/nnagaraju/sleeve/hcsc/docker-demo/ksql-demo/ksql-00.jpg)

[](file:///Users/nnagaraju/sleeve/hcsc/docker-demo/ksql-demo/ksql-01.jpg)

**Load KStream (Topic) Data into Elasticsearch**

* Either use Kafka Connector or Spark Streaming to load data into Elasticksearch index to visualize data using Kibana

{

"name": "elasticsearch-ksqldb",

"config": {

"connector.class": "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",

"consumer.interceptor.classes": "io.confluent.monitoring.clients.interceptor.MonitoringConsumerInterceptor",

"topics": "WIKIPEDIABOT",

"topic.index.map": "WIKIPEDIABOT:wikipediabot",

"connection.url": "http://elasticsearch:9200",

"type.name": "\_doc",

"key.ignore": true,

"value.converter": "io.confluent.connect.avro.AvroConverter"

}

}

**Kibana on Elasticsearch**

* Build wizards and dashboards with Kibana components on top of Elasticsearch index

**Confluent Platform’s Enterprise Features**

* Confluent Control Center
* Confluent for Kubernetes
* Confluent Connectors to Kafka
* Self-Balancing Clusters
* Confluent Cluster Linking
* Confluent Auto Data Balancer
* Confluent Replicator
* Tiered Storage
* Confluent JMS Client
* Confluent MQTT Proxy
* Confluent Security Plugins

**Community Features**

* ksqlDB
  + Streaming ETL
  + Materialized cache / views
  + Event-driven Microservices
* Confluent Clients
  + C/C++ Client Library
  + Python Client Library
  + Go Client Library
  + .NET Client Library
* Confluent Schema Registry
* Confluent REST Proxy
* CLI Tools