

DATASTAX APACHE KAFKA® CONNECTOR

QUICK REFERENCE GUIDE

A FEW FACTS

This cheat sheet is for a Kafka “sink” connector that:

- ✓ Automatically ingests from Kafka to a DataStax Distribution of Apache Cassandra (DDAC) and DataStax Enterprise (DSE) database
- ✓ Is a single jar file and is deployed on Kafka Connect worker machines
- ✓ Supports all Kafka message formats, including JSON and Avro
- ✓ Can read from many Kafka topics and write to many DDAC or DSE tables
- ✓ Works with Apache Kafka versions 0.11 and greater, Confluent versions 3.3 and greater, DSE versions 5.0 and greater, and DDAC

RESOURCES

Download: <https://downloads.datastax.com/#akc> | Documentation: <https://docs.datastax.com/en/kafka/doc/index.html>

Examples: <https://github.com/datastax/kafka-examples>

KAFKA CONNECT

Kafka Connect workers typically run in their own dedicated cluster separate from the Kafka Brokers and it is on these workers where the connector is deployed.

START KAFKA CONNECT

The `connect-distributed.properties` file configures the Kafka Connect worker. It is here where things like the broker addresses, the converters, and the `plugin.path` are configured. The path to the connector jar must be provided in the `plugin.path` option in this file.

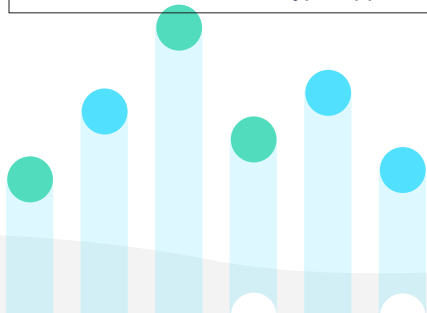
```
<kafka-home>/bin/connect-distributed.sh <kafka-home>/config/connect-distributed.properties
```

DATASTAX CONNECTOR

START KAFKA CONNECTOR

The connector is started via the Kafka Connect REST API. In the following example we are hitting the `/connectors` endpoint and using POST to send our connector configuration to the service.

```
curl -X POST -H "Content-Type: application/json" -d @connector-config.json "http://<worker-ip>:<worker-port>/connectors"
```



Example connector-config.json

```
{
  "name": "connector-example", // name of the connector instance
  "config": {
    "connector.class": "com.datastax.kafkaconnector.DseSinkConnector",
    "tasks.max": "1",
    "topics": "kafka-stream", // name of Kafka Topic(s) to read from
    "contactPoints": "127.0.0.1", // IP address of DSE or DDAC instance
    "loadBalancing.localDc": "Cassandra", // Name of DSE or DDAC data center to write to
    "topic.kafka-stream.stocks_keyspace.ticks_table.mapping": "name=key, symbol=value.symbol,
    datetime=value.datetime, exchange=value.exchange, industry=value.industry, value=value.value"
  }
}
```

MAPPING KAFKA RECORDS TO DSE TABLES

In the example above we see `topic.kafka-stream.stocks_keyspace.ticks_table.mapping`. This tells the connector to read from the Kafka topic named `kafka-stream` and write to the DSE table that is named `ticks_table`, which is in the keyspace named `stocks_keyspace`.

On the right hand side we see key value pairs. Using `symbol=value.symbol` as an example, this tells the connector to write the `symbol` field that is in the value of the Kafka record to the `symbol` column that is in the `stocks_keyspace.ticks_table`.

The connector allows for a single Kafka record to be written to multiple DSE tables, in which case you would provide several mapping configurations for a single connector instance.

UPDATE CONNECTOR CONFIGURATION

```
curl -X PUT -H "Content-Type: application/json" -d @connector.json "http://<worker-ip>:<worker-port>/connectors/<connector-name>/config"
```

DISPLAY CONNECTOR CONFIGURATION

```
curl -X GET -H "http://<worker-ip>:<worker-port>/connectors/<connector-name>/config"
```

DISPLAY CONNECTOR STATUS

```
curl -X GET "http://<worker-ip>:<worker-port>/connectors/<connector-name>/status"
```

PAUSE CONNECTOR

```
curl -X PUT "http://<worker-ip>:<worker-port>/connectors/<connector-name>/pause"
```

RESUME CONNECTOR

```
curl -X PUT "http://<worker-ip>:<worker-port>/connectors/<connector-name>/resume"
```

RESTART CONNECTOR

```
curl -X POST "http://<worker-ip>:<worker-port>/connectors/<connector-name>/restart"
```

RESTART CONNECTOR TASK

```
curl -X POST "http://<worker-ip>:<worker-port>/connectors/<connector-name>/tasks/<task-id>/restart"
```

DELETE CONNECTOR INSTANCE

```
curl -X DELETE "http://<worker-ip>:<worker-port>/connectors/<connector-name>"
```

ABOUT DATASTAX

DataStax delivers the only active everywhere hybrid cloud database built on Apache Cassandra™; DataStax Enterprise and DataStax Distribution of Apache Cassandra, a production-certified, 100% open source compatible distribution of Cassandra with expert support. The foundation for contextual, always-on, real-time, distributed applications at scale, DataStax makes it easy for enterprises to seamlessly build and deploy modern applications in hybrid cloud. DataStax also offers DataStax Managed Services, a fully managed, white-glove service with guaranteed uptime, end-to-end security, and 24x7x365 lights-out management provided by experts at handling enterprise applications at cloud scale. More than 400 of the world's leading brands like Capital One, Cisco, Comcast, Delta Airlines, eBay, Macy's, McDonald's, Safeway, Sony, and Walmart use DataStax to build modern applications that can work across any cloud. For more information, visit www.DataStax.com and follow us on Twitter @DataStax.

© 2019 DataStax, All Rights Reserved. DataStax, Titan, and TitanDB are registered trademarks of DataStax, Inc. and its subsidiaries in the United States and/or other countries. Apache, Apache Cassandra, and Cassandra are either registered trademarks or trademarks of the Apache Software Foundation or its subsidiaries in Canada, the United States, and/or other countries.