Feeding ML models with the data from the database

Vojtěch Juránek Red Hat

geecon 2023
19-20 Oct PRAGUE

Q&A

http://sli.do

#geecon

ML ops

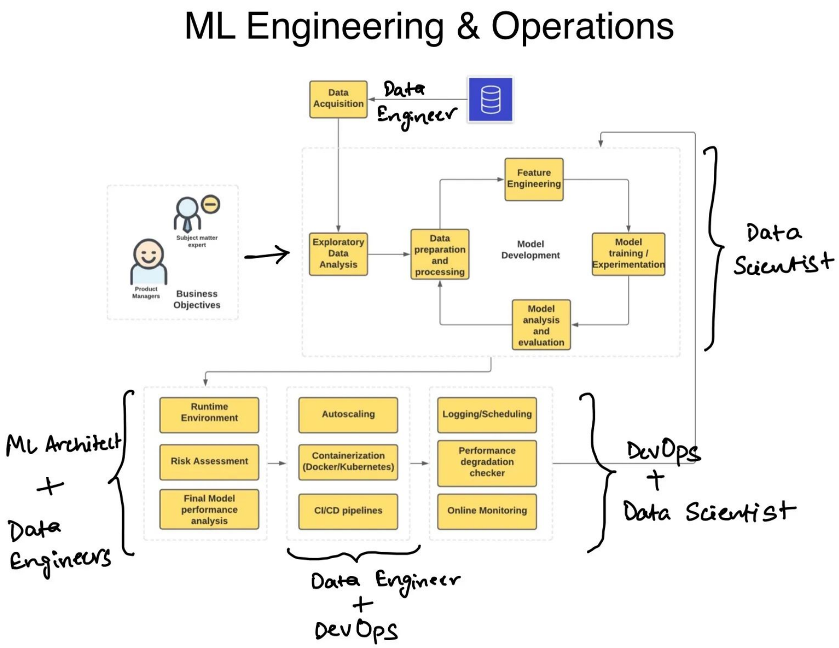
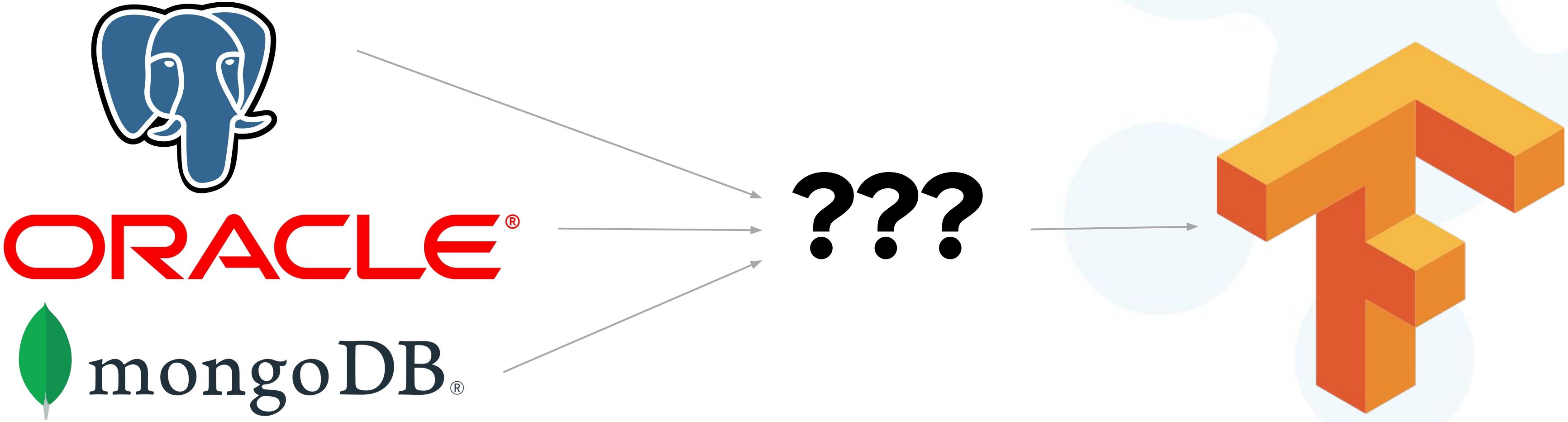


Image taken from https://towardsdatascience.com/what-is-mlops-everything-you-must-know-to-get-started-523f2d0b8bd8

ML ops

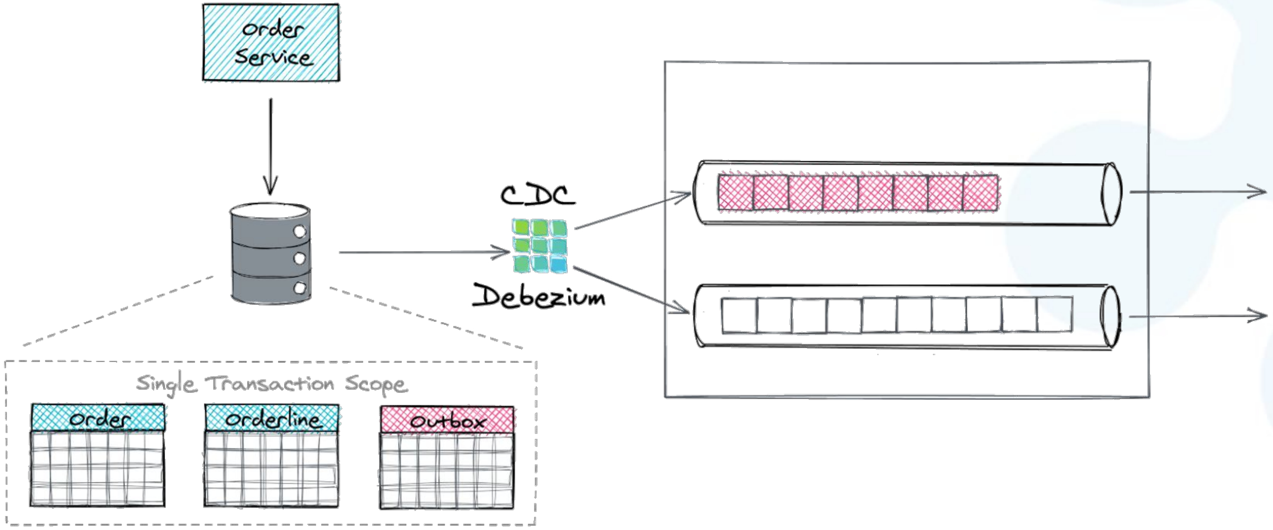


- consistent data, no data losses, no dual writes
- get all the changes without any delay in the real-time
- not overload the DB with the queries

Agenda

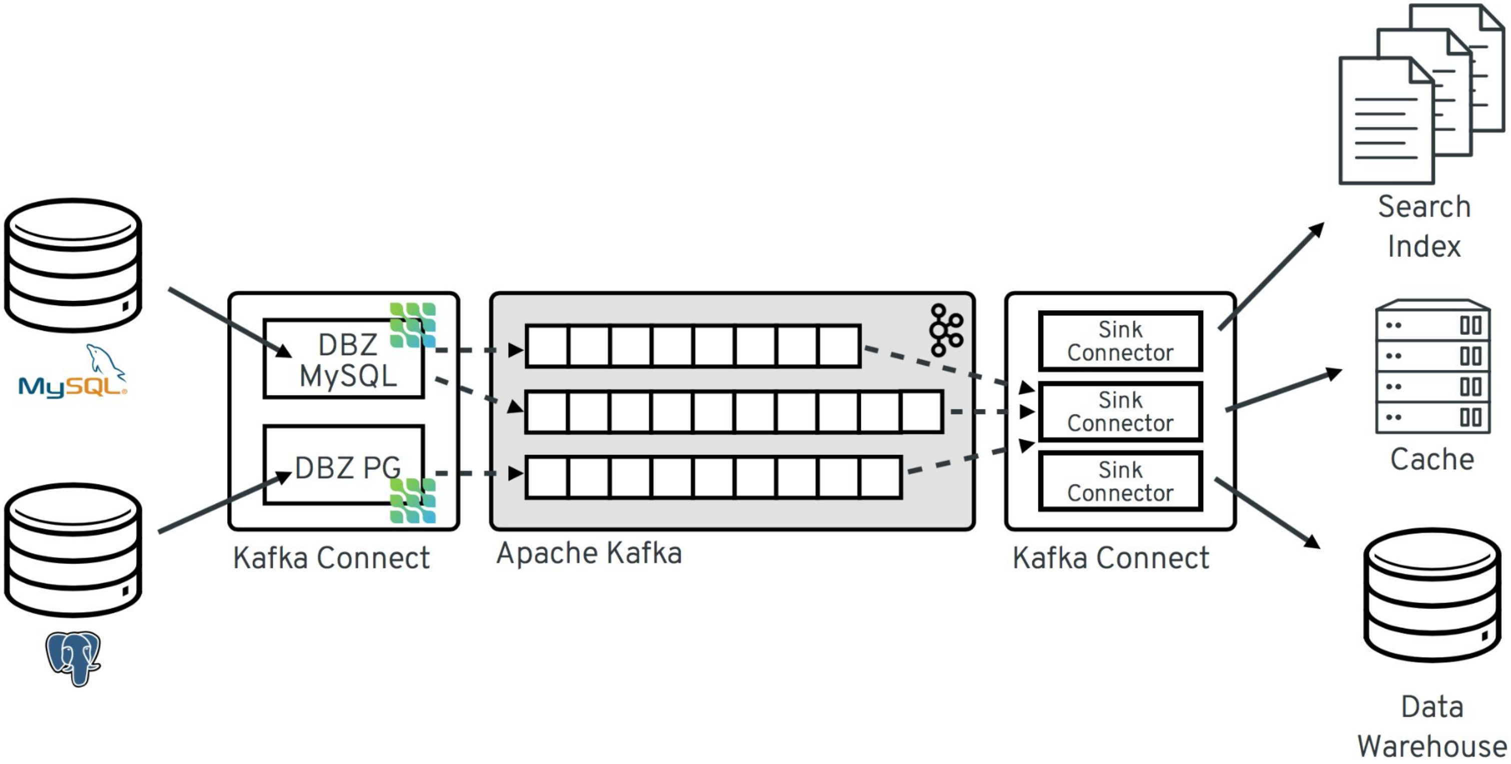
- What is Debezium and how it works.
- How we can leverage Debezium to stream changes from the database to the ML pipeline.
  - Apache Flink
  - TensorFlow

Debezium Change Data Capture

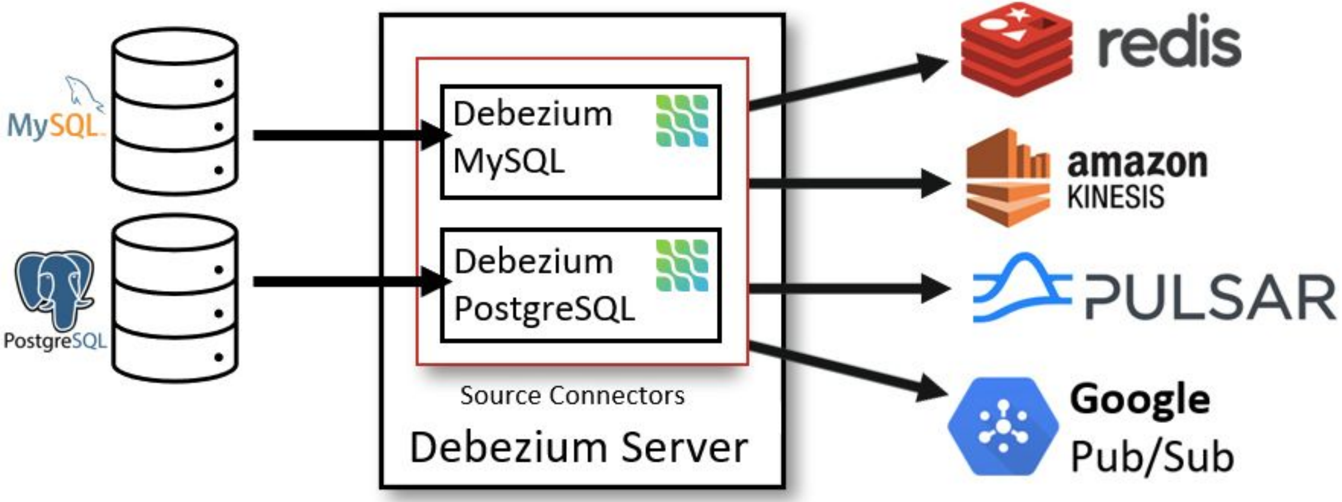


- Reads the database TX log.
- Create stream of events (create, update, delete) from it.
- Captures also schema changes.

Debezium Kafka source connector



Debezium server



Debezium supported databases

- **Stable**
  - MySQL
  - Postgres
  - SQL Server
  - MongoDB
  - Oracle
  - DB2
  - Cassandra
- **Incubating**
  - Vitess
  - Google Spanner

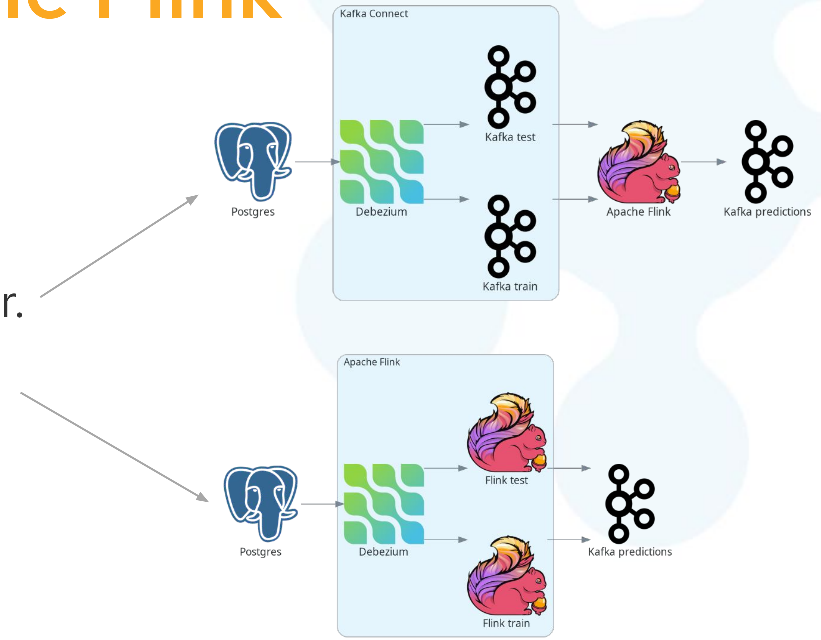
There's more!

- Initial snapshot
- Incremental snapshot

- Embedded engine
- Debezium UI
- Debezium JDBC sink connector
- Kubernetes operator
- Useful single message transformations (SMTs)
- Integrations with other frameworks (e.g. OpenTelemetry, Quarkus)
- Support for Confluent schema registry and Apicurio registry

Integration with Apache Flink

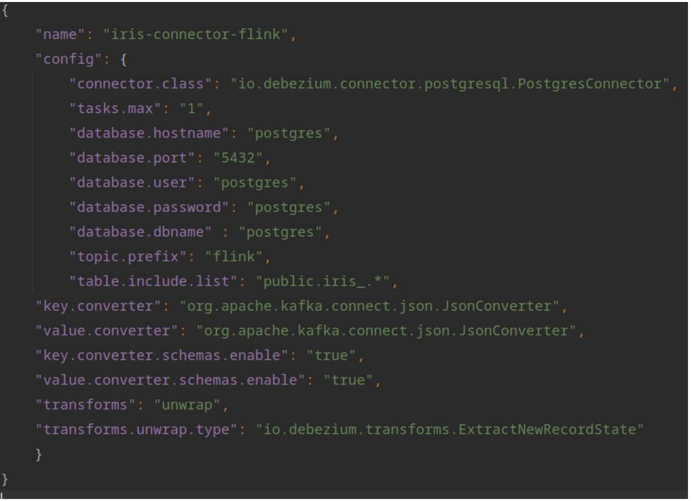
- Seamless integration via Kafka.
- Either loading data from Kafka cluster.
- Or Flink provides direct support for Debezium.



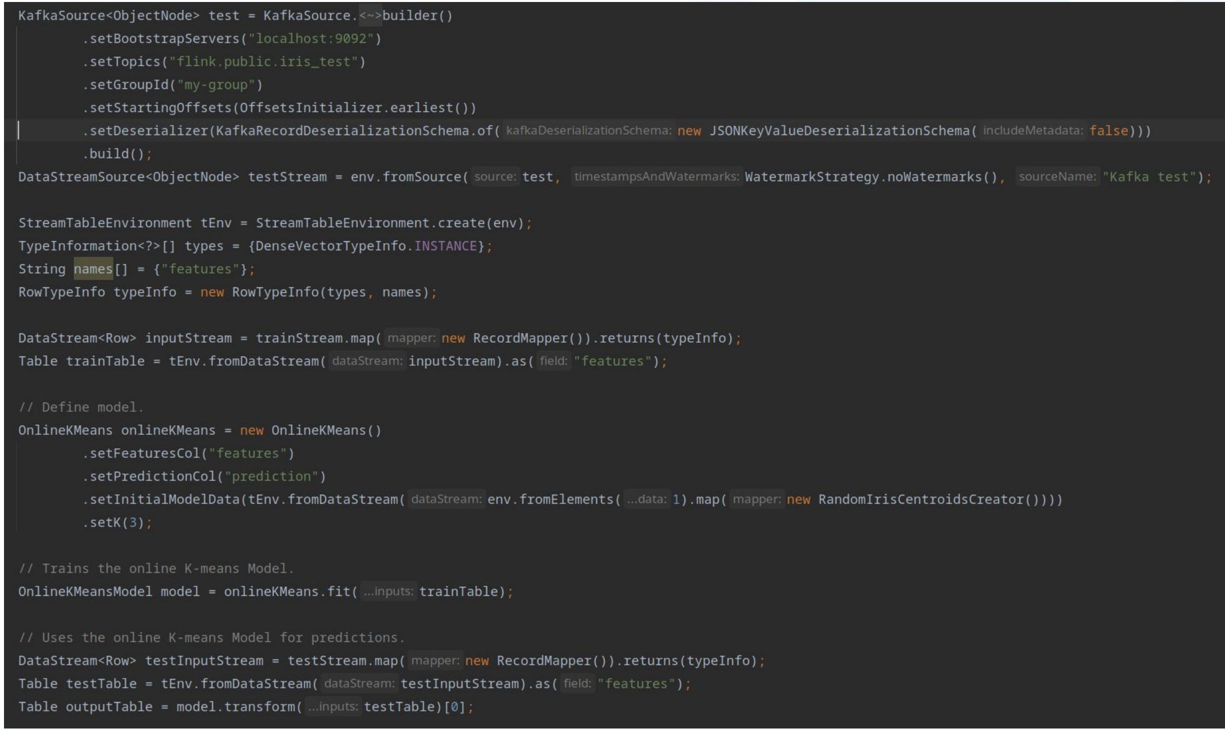
See also

- https://debezium.io/blog/2023/05/02/tensorflow-mnist-classification/
- https://github.com/debezium/debezium-examples/

Integration with Apache Flink

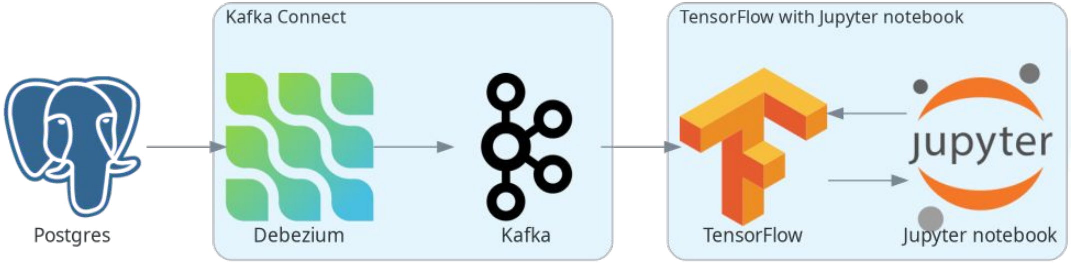


```
{
    "name": "iris-connector-flink",
    "config": {
        "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
        "tasks.max": "1",
        "database.hostname": "postgres",
        "database.port": "5432",
        "database.user": "postgres",
        "database.password": "postgres",
        "database.dbname" : "postgres",
        "topic.prefix": "flink",
        "table.include.list": "public.iris_.*",
    "key.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "key.converter.schemas.enable": "true",
    "value.converter.schemas.enable": "true",
    "transforms": "unwrap",
    "transforms.unwrap.type": "io.debezium.transforms.ExtractNewRecordState"
    }
}
```

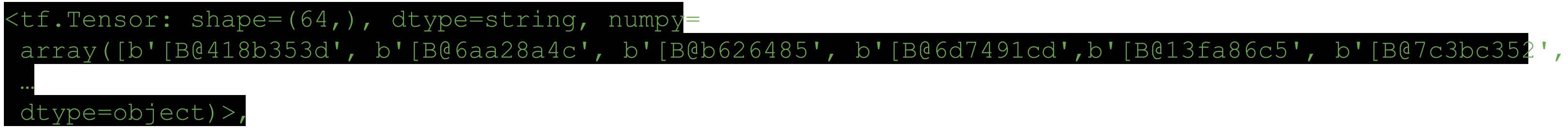


```
KafkaSource<ObjectNode> test = KafkaSource.<~>builder()
        .setBootstrapServers("localhost:9092")
        .setTopics("flink.public.iris_test")
        .setGroupId("my-group")
        .setStartingOffsets(OffsetsInitializer.earliest())
        .setDeserializer(KafkaRecordDeserializationSchema.of( kafkaDeserializationSchema: new JSONKeyValueDeserializationSchema( includeMetadata: false)))
        .build();
DataStreamSource<ObjectNode> testStream = env.fromSource( source: test,  timestampsAndWatermarks: WatermarkStrategy.noWatermarks(),  sourceName: "Kafka test");

StreamTableEnvironment tEnv = StreamTableEnvironment.create(env);
TypeInformation<?>[] types = {DenseVectorTypeInfo.INSTANCE};
String names[] = {"features"};
RowTypeInfo typeInfo = new RowTypeInfo(types, names);

DataStream<Row> inputStream = trainStream.map( mapper: new RecordMapper()).returns(typeInfo);
Table trainTable = tEnv.fromDataStream( dataStream: inputStream).as( field: "features");

// Define model.
OnlineKMeans onlineKMeans = new OnlineKMeans()
        .setFeaturesCol("features")
        .setPredictionCol("prediction")
        .setInitialModelData(tEnv.fromDataStream( dataStream: env.fromElements( ...data: 1).map( mapper: new RandomIrisCentroidsCreator())))
        .setK(3);

// Trains the online K-means Model.
OnlineKMeansModel model = onlineKMeans.fit( ...inputs: trainTable);

// Uses the online K-means Model for predictions.
DataStream<Row> testInputStream = testStream.map( mapper: new RecordMapper()).returns(typeInfo);
Table testTable = tEnv.fromDataStream( dataStream: testInputStream).as( field: "features");
Table outputTable = model.transform( ...inputs: testTable)[0];
```

Integration with TensorFlow



Integration via Kafka works, but serialized object are interpreted as strings:



```
<tf.Tensor: shape=(64,), dtype=string, numpy=
 array([b'[B@418b353d', b'[B@6aa28a4c', b'[B@b626485', b'[B@6d7491cd',b'[B@13fa86c5', b'[B@7c3bc352',
 …
 dtype=object)>,
```

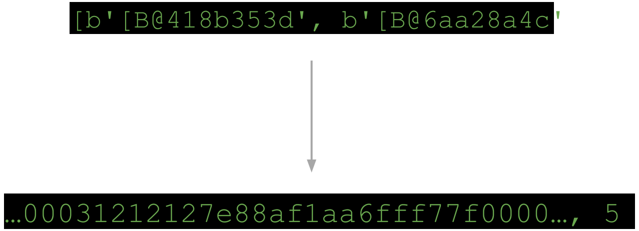
SMT to rescue

Single message transformations (SMTs):

- transform inbound and/or outbound messages
- can be used also e.g. for filtering to save bandwidth on the early stage of the ML pipeline
- many SMTs available out-of-the-box
- very easy to write custom SMT

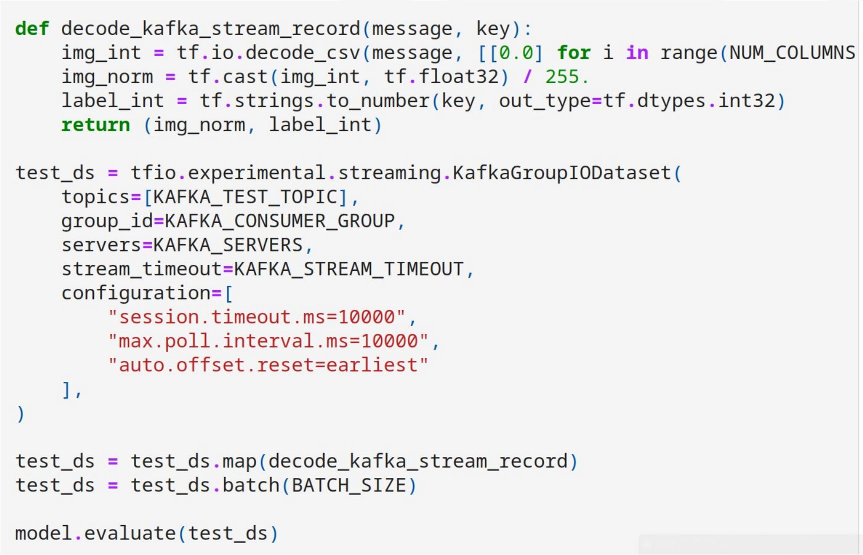
SMT to rescue

TF-Kafka can be fixed by writing SMT which converts Debezium records to CSV string.



```
[b'[B@418b353d', b'[B@6aa28a4c'
```

```
…00031212127e88af1aa6fff77f0000…,  5
```



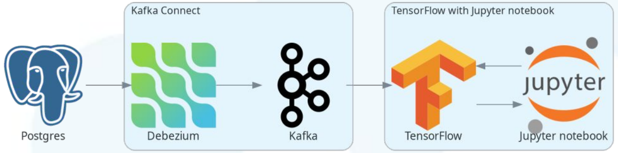
```python
def decode_kafka_stream_record(message, key):
    img_int = tf.io.decode_csv(message, [[0.0] for i in range(NUM_COLUMNS
    img_norm = tf.cast(img_int, tf.float32) / 255.
    label_int = tf.strings.to_number(key, out_type=tf.dtypes.int32)
    return (img_norm, label_int)

test_ds = tfio.experimental.streaming.KafkaGroupIODataset(
    topics=[KAFKA_TEST_TOPIC],
    group_id=KAFKA_CONSUMER_GROUP,
    servers=KAFKA_SERVERS,
    stream_timeout=KAFKA_STREAM_TIMEOUT,
    configuration=[
        "session.timeout.ms=10000",
        "max.poll.interval.ms=10000",
        "auto.offset.reset=earliest"
    ],
)

test_ds = test_ds.map(decode_kafka_stream_record)
test_ds = test_ds.batch(BATCH_SIZE)

model.evaluate(test_ds)
```

See also:

- https://debezium.io/blog/2023/05/02/tensorflow-mnist-classification/
- https://github.com/debezium/debezium-examples/

Takeaways

- CDC is a powerful concept for real-time predictions as well as for ML methods like online machine learning.
- Loading existing data from your database(s) and streaming new data in real time into your ML pipeline is easy with Debezium.



- SMTs are very powerful tool to filter out or modify your data.
- SMTs can fix also possible integration issues.

Thank you!

Q&A

debezium

https://debezium.zulipchat.com/
https://groups.google.com/g/debezium
https://github.com/debezium/