github.com/**saubury/socksort**

Pairing socks with ksqlDB, Kafka and Kafka Connect

linkedin.com/in/**simonaubury**

@SimonAubury

Hello!

I am Simon Aubury

Principal Data Engineer @ ThoughtWorks

I am here because I love streaming

Event Driven sock sorting

Pairing socks with Kafka

Can I arrange my sock drawer with ksqlDB, Kafka and Kafka Connect?

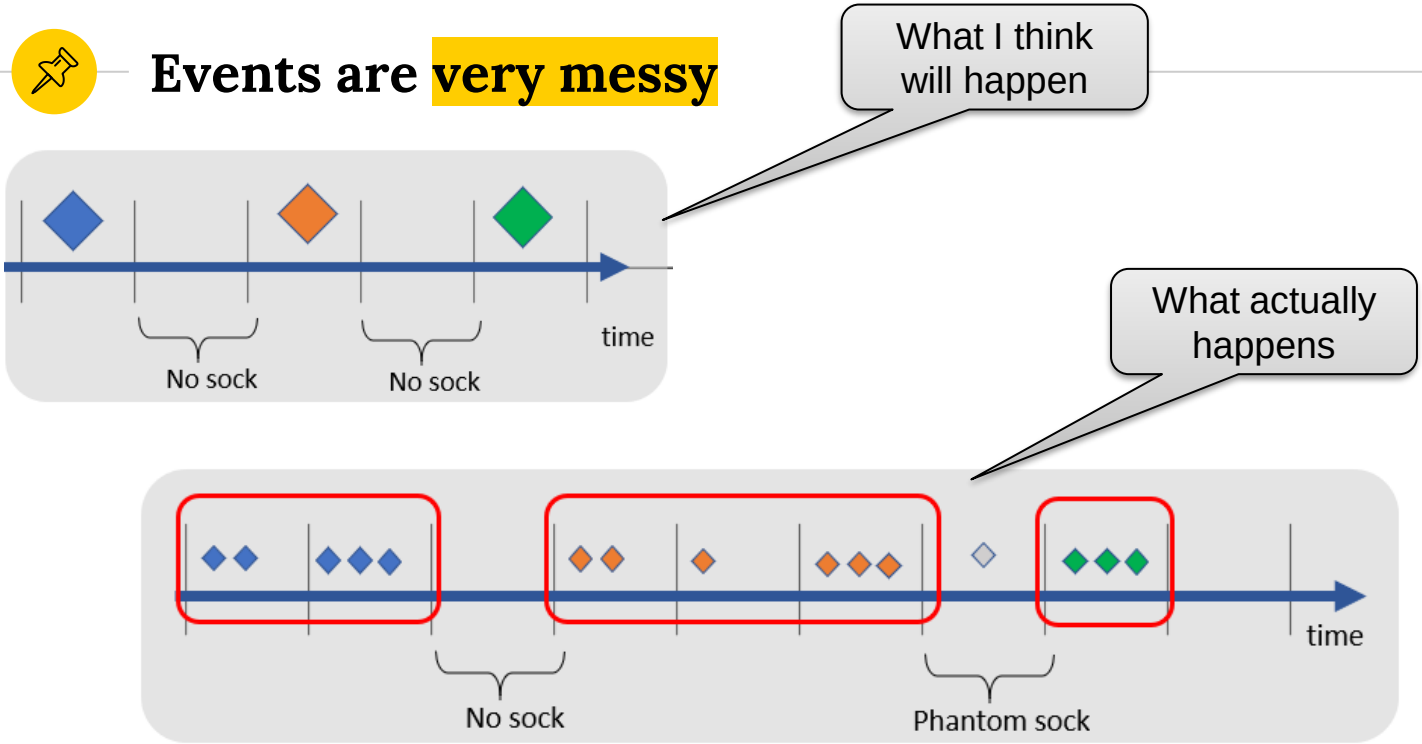

Why?

Events are everywhere

Event-driven architecture is an architecture paradigm promoting the production, detection, consumption of and reaction to events.

- Object detection is a stream of events

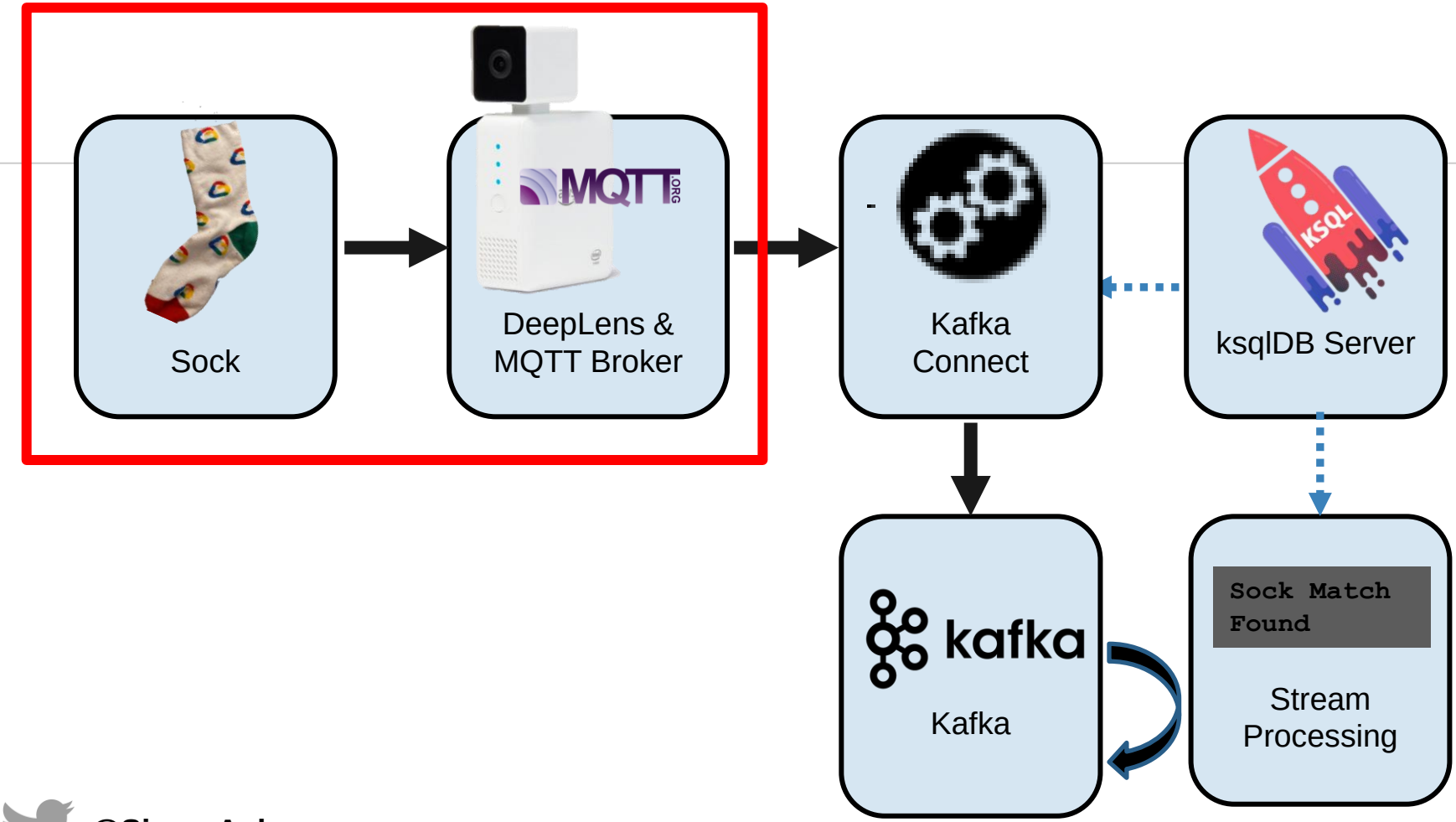
Events are very messy



Let's start at the finish

- Hold a sock in front of camera
- *Stuff happens involving Kafka*
- ksqlDB displays the location of the other sock


Warning!
Some non-Kafka stuff ahead ...



@SimonAubury

Deep learning video camera AWS Deeplens

AWS Deeplens Hardware

- A “deep-learning enabled video camera”.
  - 4MP video camera,
  - Intel Atom Processor
  - 8GB RAM
  - Runs Ubuntu.
- Plenty of hardware to help sort my socks.

Many photos of socks

Supervised learning image classification

- Requires training data
- Prepare a set of training images
- I need to take a lot of photos of socks

Model Training

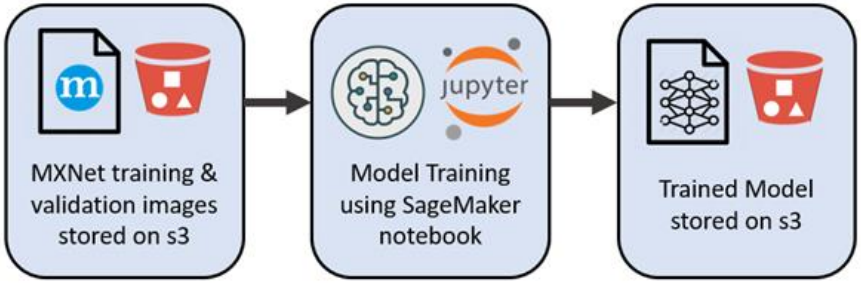
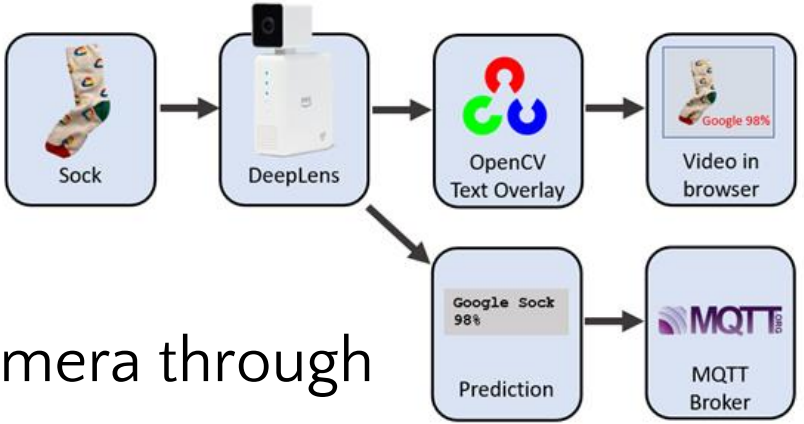


Image classification of socks using transfer learning mode.

- Use Amazon Sagemaker image classification algorithm in transfer learning
- Deploy a temporary classifier to test the inference function
- Test a few demonstration images can be correctly classified

Inference Lambda Function



- Run all images captured by the camera through the classification model.
  - Review a live camera feed within as web-browser
  - OpenCV adds text overlaid on the image.
- Write to a MQTT topic

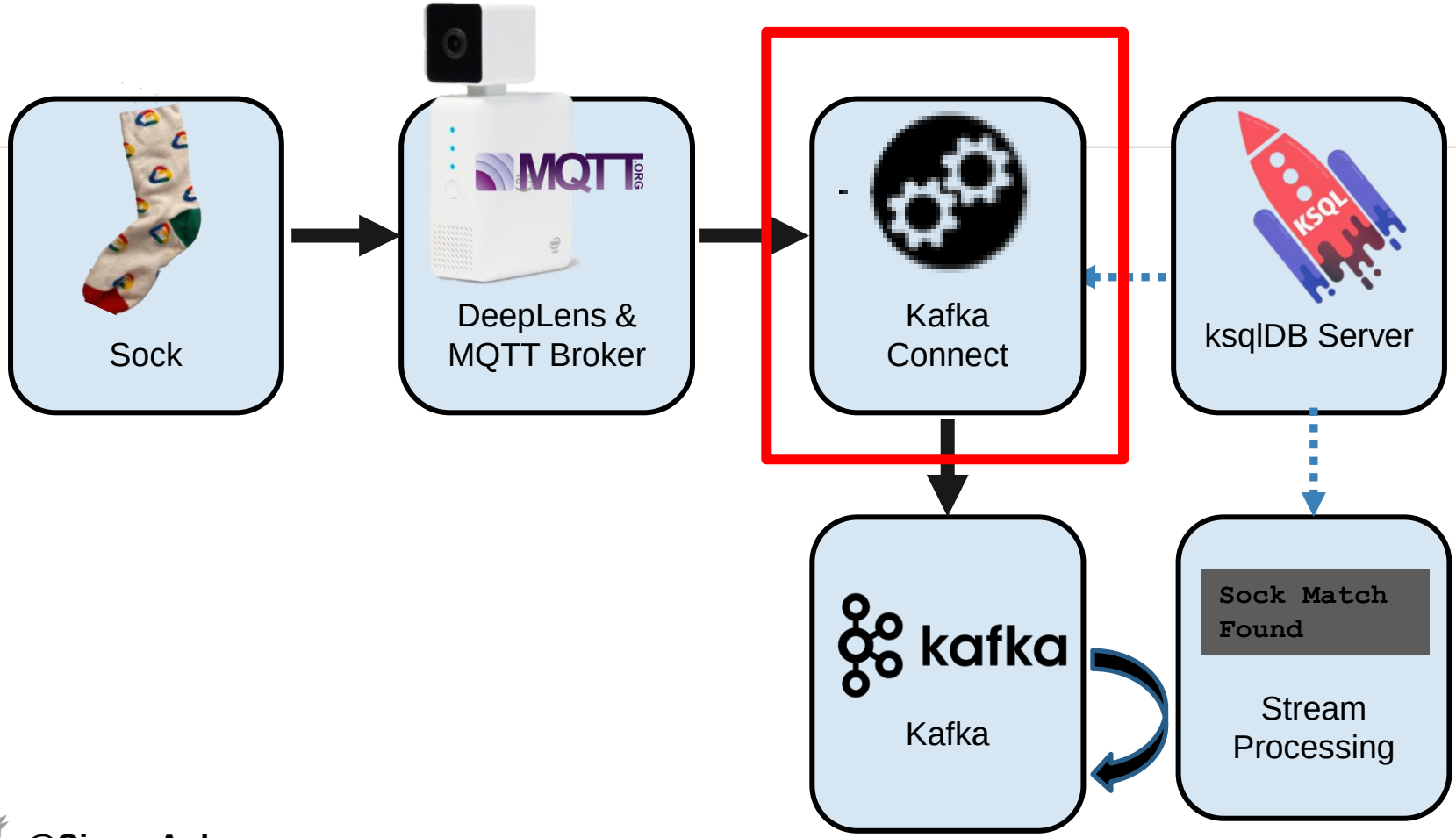
What is MQTT?

```
mosquitto_sub -h ${MQTT_HOST} -p ${MQTT_PORT} -u ${MQTT_USER} -P
${MQTT_PASS} -t sockfound


{"image": "Blank", "probability": 37.59765625}
{"image": "Blank", "probability": 41.162109375}
{"image": "Google", "probability": 97.314453125}
{"image": "Google", "probability": 94.970703125}
{"image": "Google", "probability": 64.6484375}
{"image": "Blank", "probability": 67.3828125}
{"image": "Blank", "probability": 50.634765625}
```

- MQTT is lightweight TCP/IP protocol
  - Small footprint
  - Low power
- MQTT acts more like a key/value store
  - Whereas Kafka is a complete streaming platform.


Back to safety

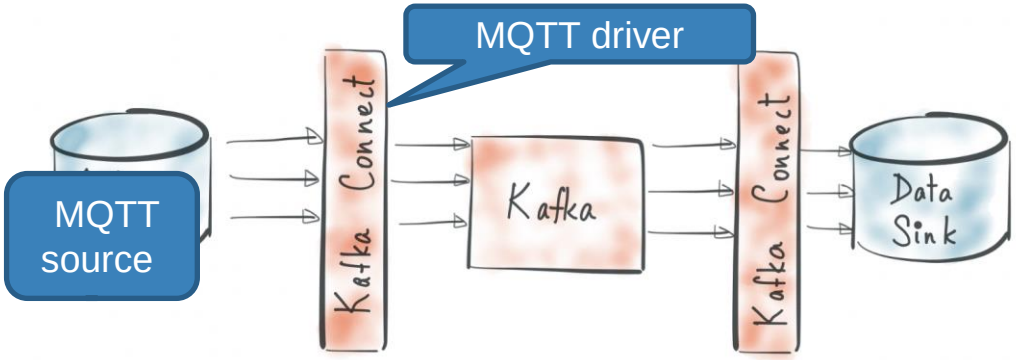


@SimonAubury

Kafka connect

Kafka Connect is a framework for streaming data between Apache Kafka and other data systems

- Kafka connect **sources** – from something into Kafka
- Kafka connect **sinks** – from Kafka into something

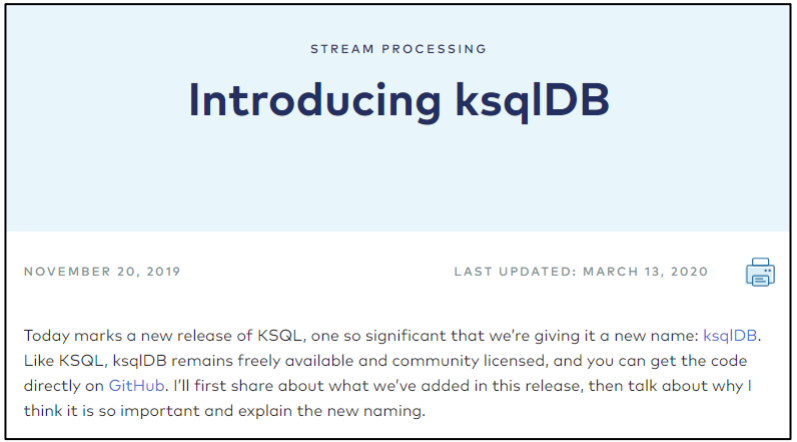


MQTT to Kafka with Kafka Connect

- Save config to file
- Manage the secrets
- Find the end point of Connect server
- Fire up curl / postman to check config
- … do more stuff to check

KSQL to ksqlDB

- KSQL - build event streaming applications “with SQL”
- ksqlDB – KSQL with
  - Push **and Pull** queries
  - Connector management



STREAM PROCESSING

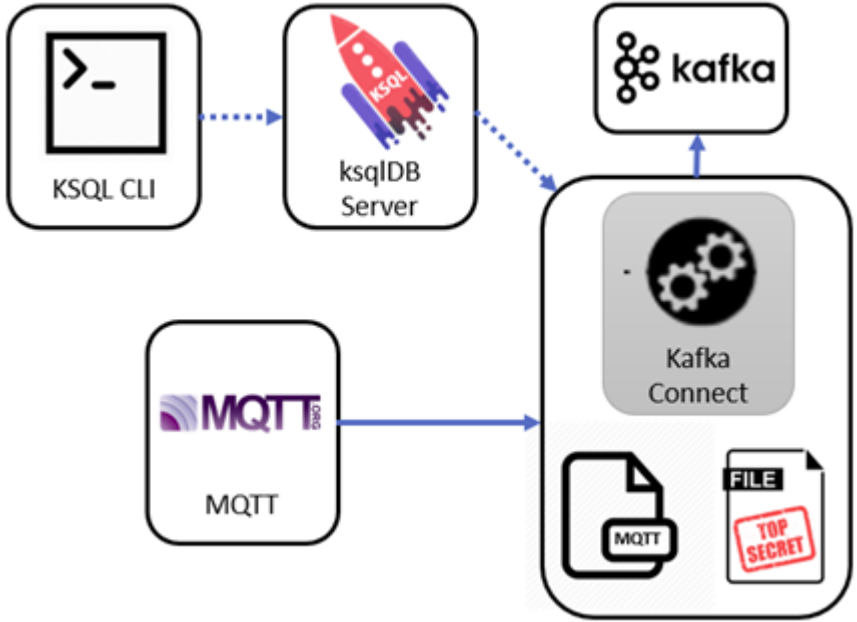
Introducing ksqlDB

NOVEMBER 20, 2019 LAST UPDATED: MARCH 13, 2020

Today marks a new release of KSQL, one so significant that we're giving it a new name: ksqlDB. Like KSQL, ksqlDB remains freely available and community licensed, and you can get the code directly on GitHub. I'll first share about what we've added in this release, then talk about why I think it is so important and explain the new naming.

Kafka Connect with ksqlDB

ksqlDB now has commands so you can directly setup and control Kafka Connect.



MQTT to Kafka with Kafka Connect by using ksqlDB

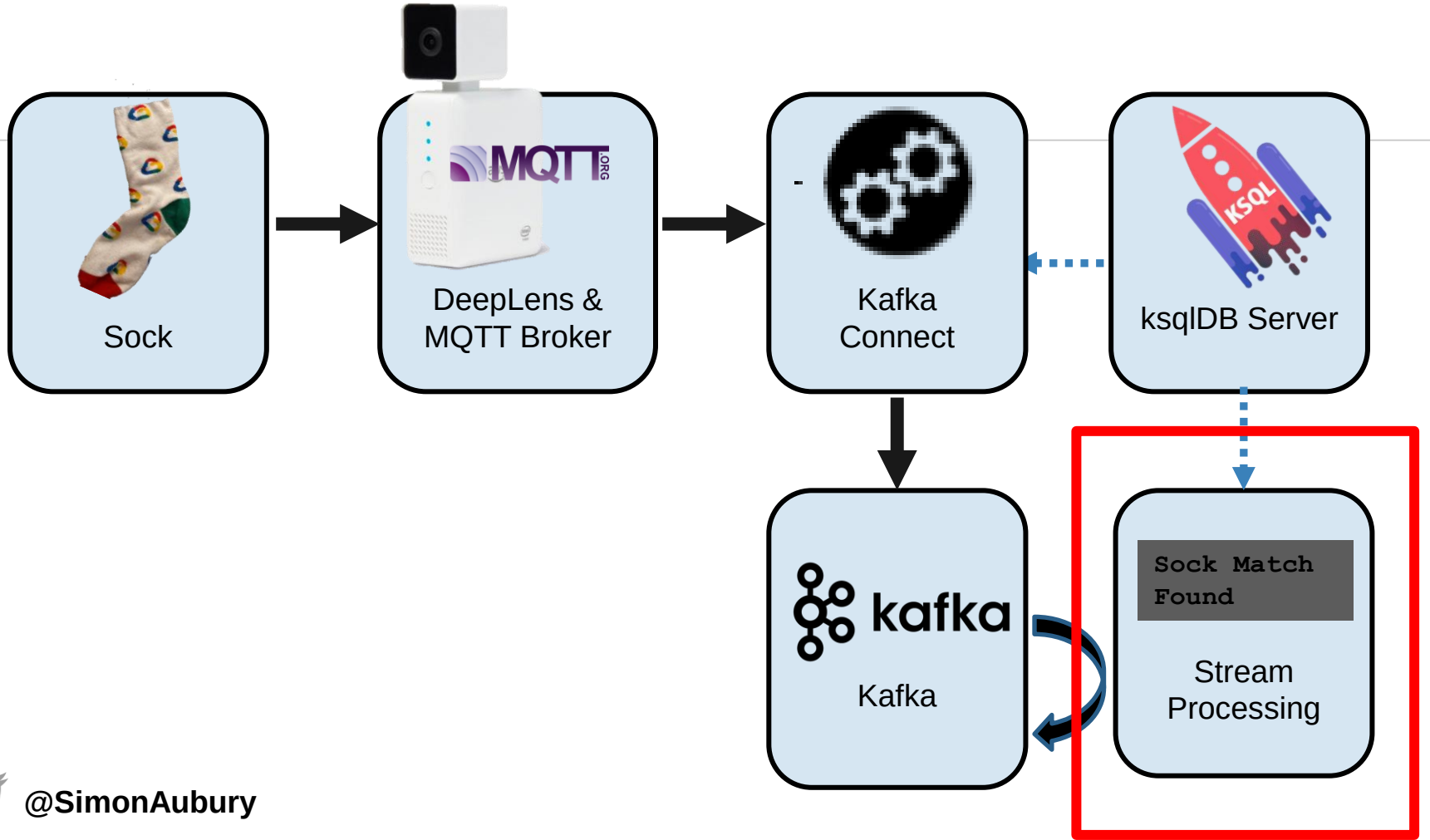
ksqlDB command

```
CREATE SOURCE CONNECTOR `mqtt-source` WITH(
    "connector.class"='io.confluent.connect.mqtt.MqttSourceConnector',
    "mqtt.server.uri"='${file:/scripts/credentials.properties:MQTT_URI}',
    "mqtt.username"='${file:/scripts/credentials.properties:MQTT_USERNAME}',
    "mqtt.password"='${file:/scripts/credentials.properties:MQTT_PASSWORD}',
    "mqtt.topics"='sockfound',
    "kafka.topic"='data_mqtt',
    "key.converter"='org.apache.kafka.connect.storage.StringConverter',
    "value.converter"='org.apache.kafka.connect.converters.ByteArrayConverter',
    "tasks.max"='1',
    "confluent.topic.bootstrap.servers"='kafka:29092',
    "confluent.topic.replication.factor"='1'
);
```

```
MQTT_URI=tcp://something.example.com:14437
MQTT_USERNAME=someuser
MQTT_PASSWORD=somepassword
```

credentials.properties

@SimonAubury



@SimonAubury

Do we have any socks in Kafka?

We can check incoming MQTT messages are landing in Kafka by querying the Kafka topic with KSQL

```
ksql> print 'data_mqtt';
```

```
{"image": "Running Science", "probability": 43.994140625}
{"image": "Mongo", "probability": 50.29296875}
{"image": "Mongo", "probability": 86.279296875}
{"image": "Mongo", "probability": 53.076171875}
```

Create a stream with ksqlDB

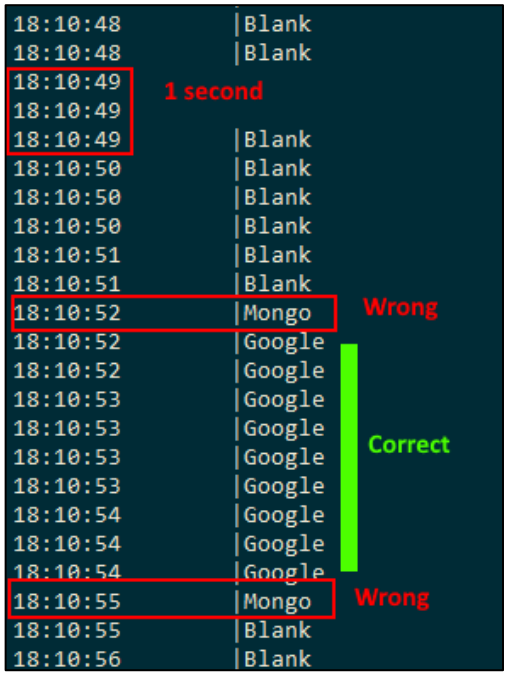
Goal: create a stream for our Kafka topic containing JSON

```
{"image": "Running Science", "probability": 43.994140625}
{"image": "Mongo", "probability": 50.29296875}
{"image": "Mongo", "probability": 86.279296875}
{"image": "Mongo", "probability": 53.076171875}
```

```
create stream sock_stream(image varchar, probability double)
with (kafka_topic='data_mqtt',  value_format='json');
```

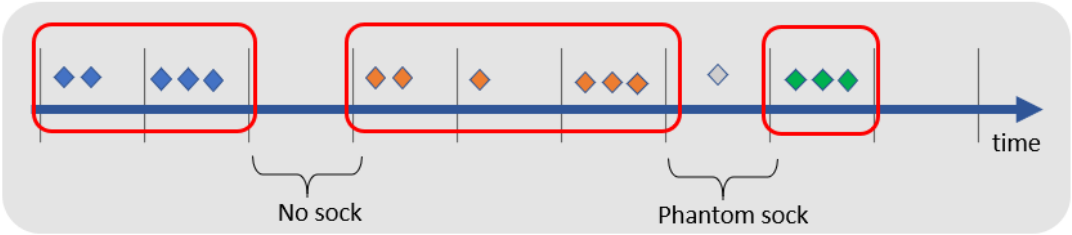
Ghost socks

Image classifier identifies 3–4 images each second.



Stream processing with ksqlDB

Goal: find similar messages within a windows of time



Window Hopping with ksqlDB

Goal: find 4 or more identical socks in a rolling 5 second window

```
create table sock_stream_smoothed as
select image
, timestamptostring(windowstart(), 'hh:mm:ss') as last_seen
, windowstart() as window_start
from sock_stream
window tumbling (size 5 seconds)
group by image having count(*) > 3
emit changes;
```

Staring at the wall with ksqlDB

Goal: eliminate pictures of the wall

```
create stream sock_stream_without_blanks as
select image
from sock_stream
where image != 'blank';
```

Pairing socks with ksqlDB

Goal: find pairs of identical socks

```
select image
, case when (count(*)/2)*2 = count(*) then 'Pair' else 'Un-matched'
end  as pair_seen
, count(*) as number_socks_seen
from sock_stream_smoothed
group by image
emit changes;
```

```
+--------------+-------------+--------------------+
|IMAGE         |PAIR_SEEN    |NUMBER_SOCKS_SEEN   |
+--------------+-------------+--------------------+
|Mongo         |Pair         |2                   |
|Streamset     |Un-matched   |1                   |
|Google        |Pair         |2                   |
|Confluent     |Pair         |2                   |
```

Demo

What did I learn?

What did I learn?

- Object detection is a stream of events
- Events are very messy
  - Stream processing doesn't need to be
- Kafka connect is cool and easy way to integrate systems
  - Even easier to manage with ksqlDB
- Sock sorting **does not** make you interesting at parties

Thanks!

Any *questions* ?

This Presentation

linkedin.com/in/**simonaubury**

github.com/**saubury/socksort**

Presentation template by SlidesCarnival