Vision processing / with Kafka & ksqlDB

Can I arrange my sock drawer using machine learning?

Simon Aubury

/thoughtworks

@Simon Aubury

Simon Aubury

Principal Data Engineer

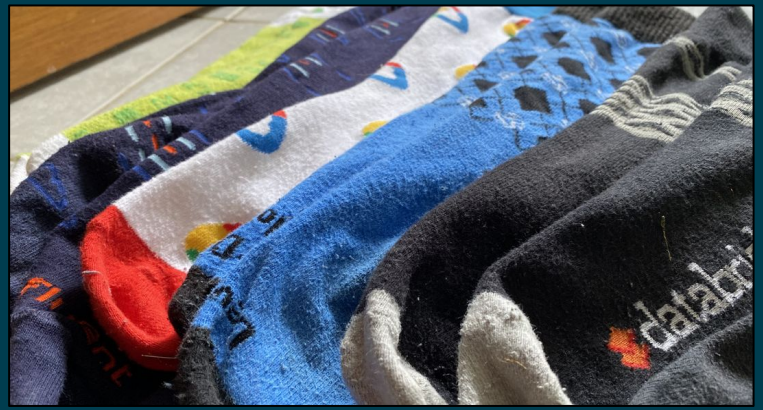
I am here because I love streaming

/thoughtworks

@Simon Aubury

Pairing socks with ML

Can I arrange my sock drawer with transfer learning to build a custom sock image classification model?



github.com/saubury/socksort


Why?

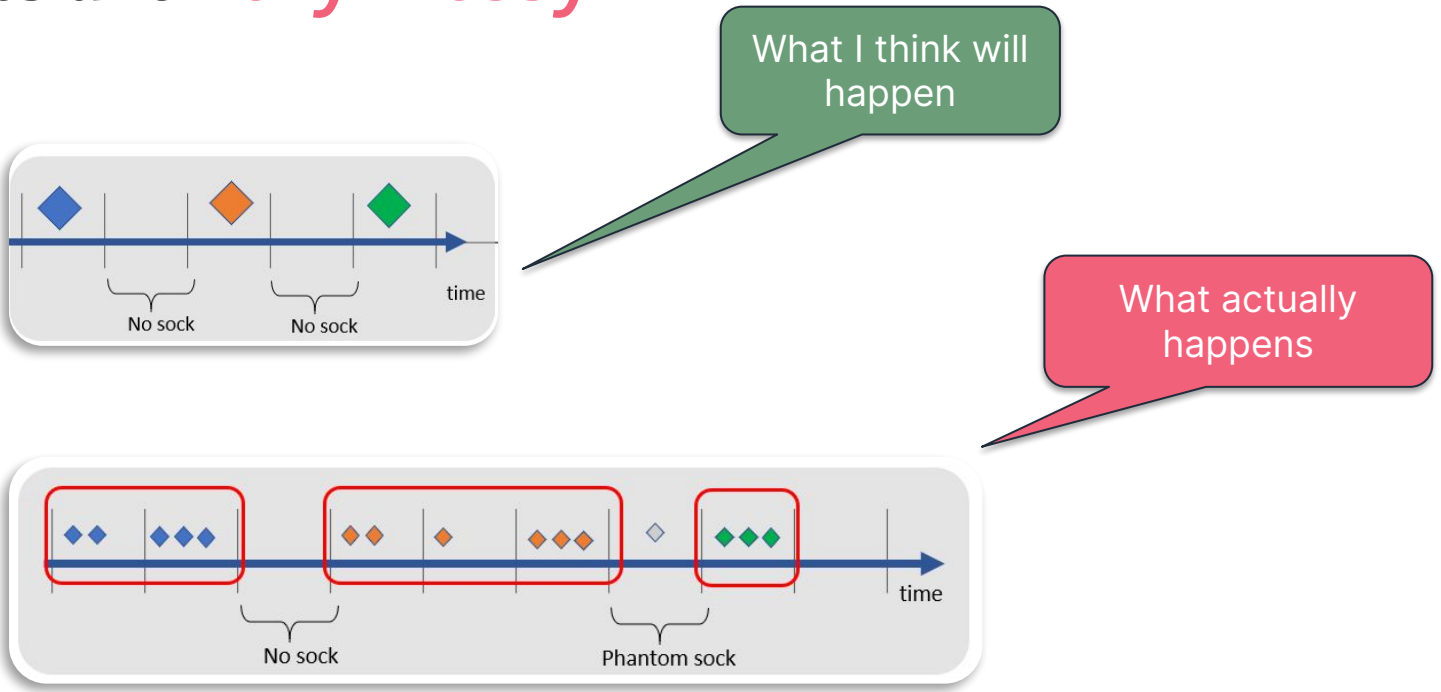
Events are everywhere

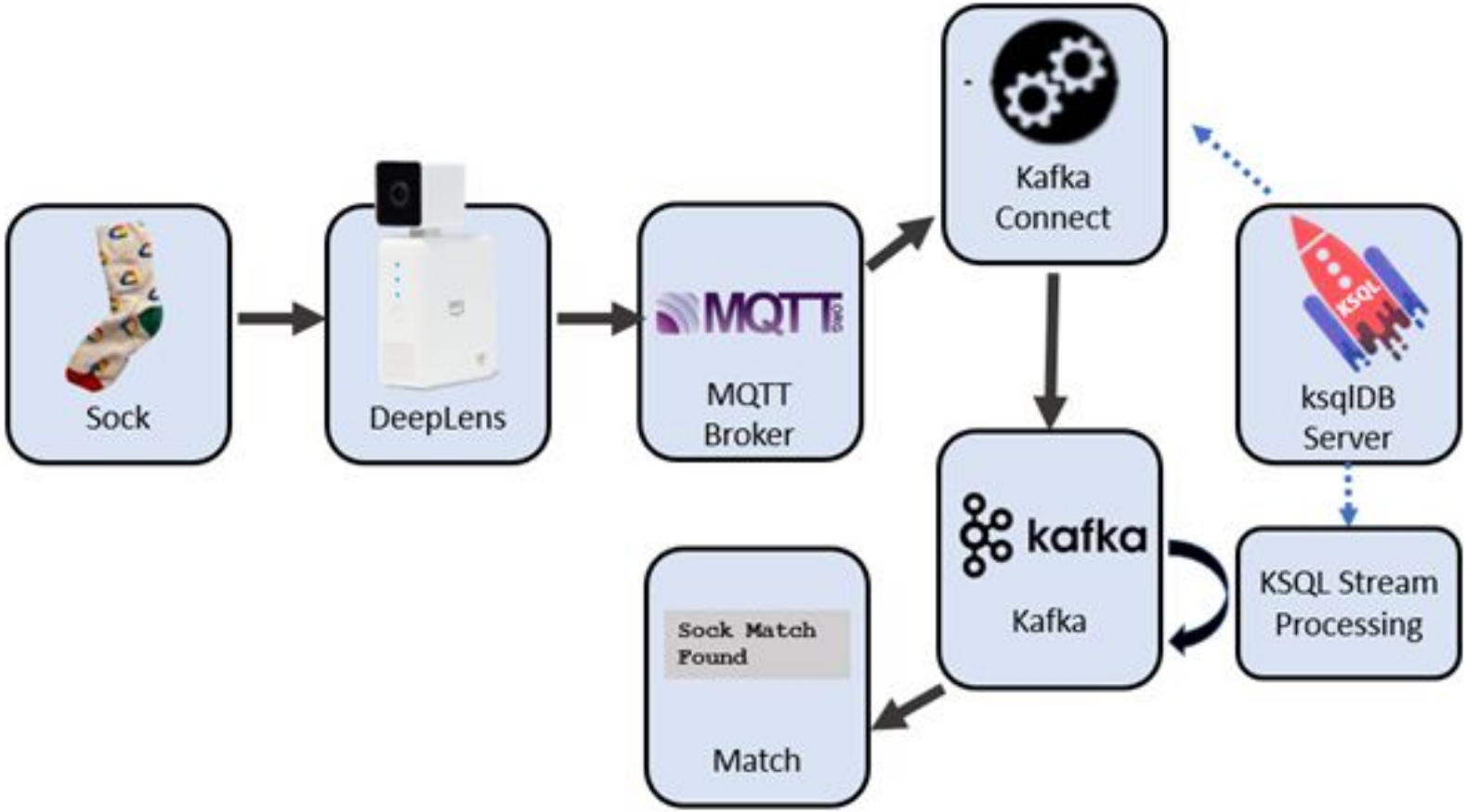
Event-driven architecture is an architecture paradigm promoting the production, detection, consumption of and reaction to events.

Object detection is a stream of events

@Simon Aubury

Events are very messy

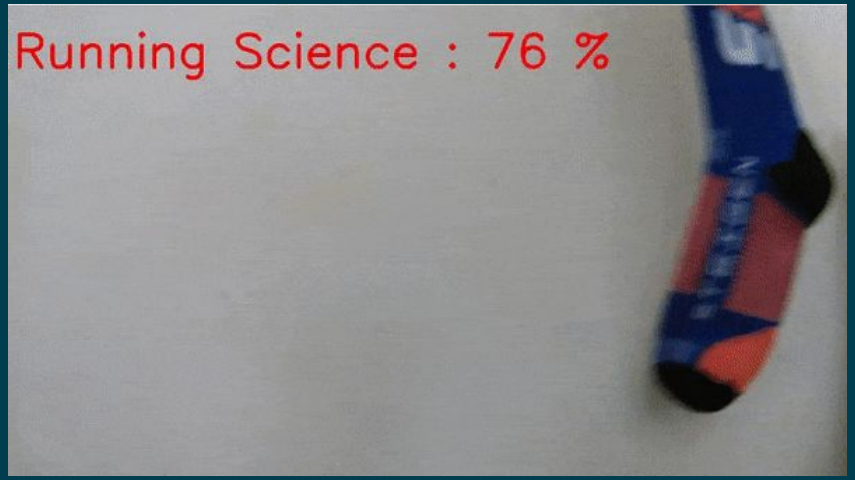




@Simon Aubury

Let's start at the finish

- Hold a sock in front of camera
- Classification for each frame
- Messages written to MQTT
- Messages transported to Kafka
- Stream processing on Kafka
- Socks are matched



Deep learning video camera AWS Deeplens

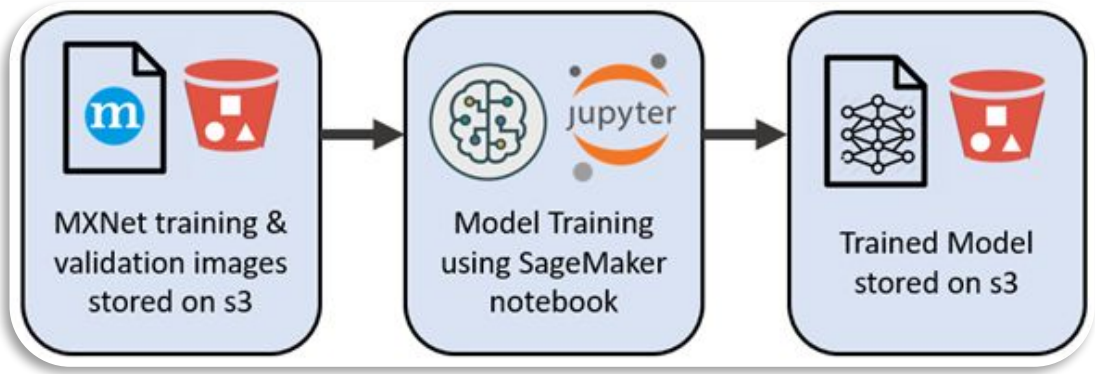
AWS Deeplens Hardware

- 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

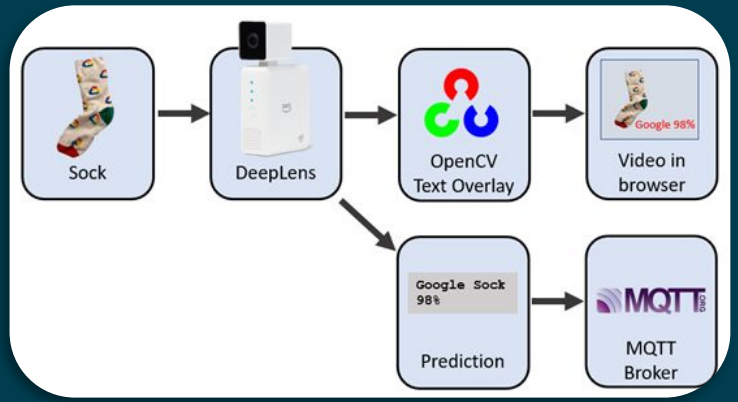
- Requires training data
- Prepare a set of training images
- I need to take a lot of photos of socks



Model Training

Image classification of socks using transfer learning mode.

- Use AWS 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

On the DeepLens

- 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

Inference Lambda Function

```
while doInfer:
    # Get a frame from the video stream
    ret, frame = awscam.getLastFrame()
    # Raise an exception if failing to get a frame
    if ret == False:
        raise Exception("Failed to get frame from the stream")

    # Resize frame to fit model input requirement
    frameResize = cv2.resize(frame, (input_width, input_height))

    # Run model inference on the resized frame
    inferOutput = model.doInference(frameResize)

    # Output inference result to the fifo file so it can be viewed with mplayer
    parsed_results = model.parseResult(model_type, inferOutput)
    top_k = parsed_results[model_type][0:topk]

    sock_label = labels[top_k[0]["label"]]
    sock_prob = top_k[0]["prob"]*100

    # Write to MQTT
    json_payload = {"image" : sock_label, "probability" : sock_prob}
    client.publish(topic=iot_topic, payload=json.dumps(json_payload))

    # Write to image buffer;  screen display
    msg_screen = '{} {:.0f}%'.format(sock_label, sock_prob)
    cv2.putText(frame, msg_screen, (20,200), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 0, 255), 12)
    local_display.set_frame_data(frame)
```


https://github.com/saubury/socksort/blob/master/30-deeplens-greengrass-lambda/deeplens_inference.py

What is MQTT?

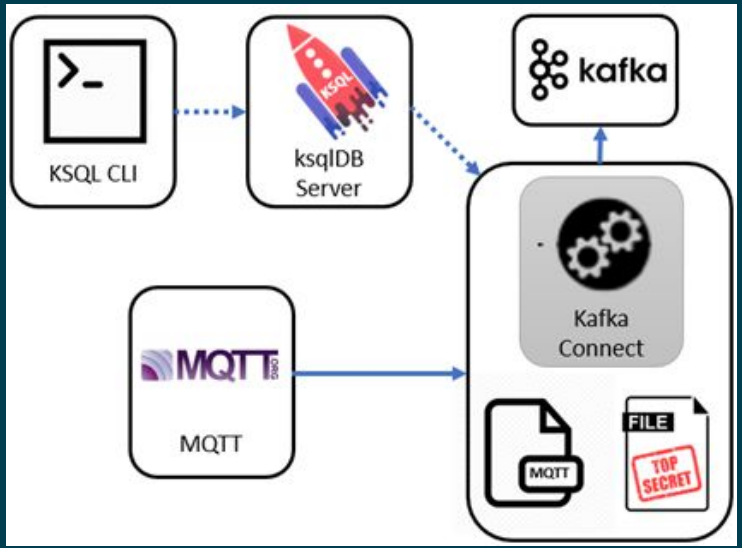
- MQTT is lightweight TCP/IP protocol
  - Small footprint
  - Low power
- MQTT acts more like a key/value store

```
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}
```

Kafka, Kafka Connect & ksqlDB

Kafka is a distributed streaming platform

- Kafka - platform for handling real-time data
- Kafka Connect - framework for streaming data between Kafka and other data systems
- ksqlDB - build real-time systems with SQL statements



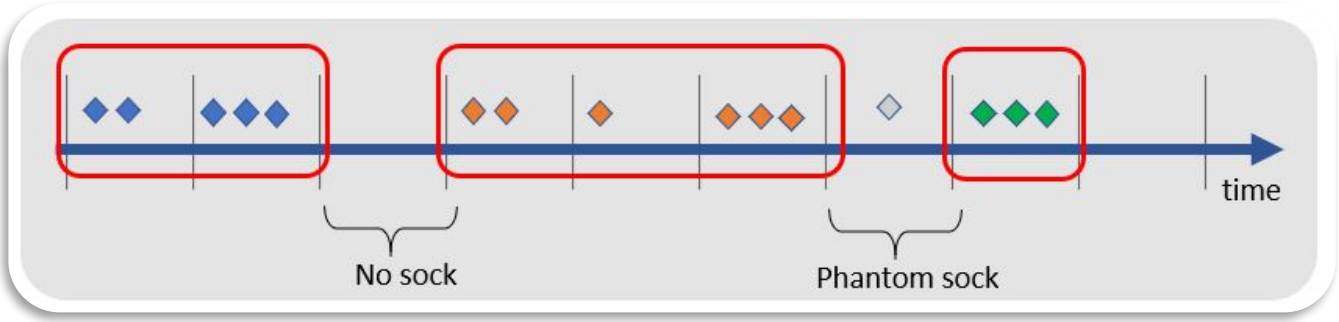

```
CREATE SOURCE CONNECTOR `mqtt-source` WITH(
"connector.class"='io.confluent.connect.mqtt.MqttSourceConnector',
"mqtt.server.uri"='tcp://something.example.com:14437',
"mqtt.username"='some-user',
"mqtt.password"='my-password',
"mqtt.topics"='sockfound',
"kafka.topic"='data_mqtt',
);
```

👻 Ghost socks

Image classifier identifies 3-4 images each second.

```
18:10:48        |Blank
18:10:48        |Blank
18:10:49        1 second
18:10:49
18:10:49        |Blank
18:10:50        |Blank
18:10:50        |Blank
18:10:50        |Blank
18:10:51        |Blank
18:10:51        |Blank
18:10:52        |Mongo    Wrong
18:10:52        |Google
18:10:52        |Google
18:10:53        |Google
18:10:53        |Google   Correct
18:10:53        |Google
18:10:53        |Google
18:10:54        |Google
18:10:54        |Google
18:10:54        |Google
18:10:55        |Mongo    Wrong
18:10:55        |Blank
18:10:56        |Blank
```

@Simon Aubury



Event Stream processing

Goal: find similar messages within a windows of time

```
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;
```

Window Hopping with ksqlDB

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

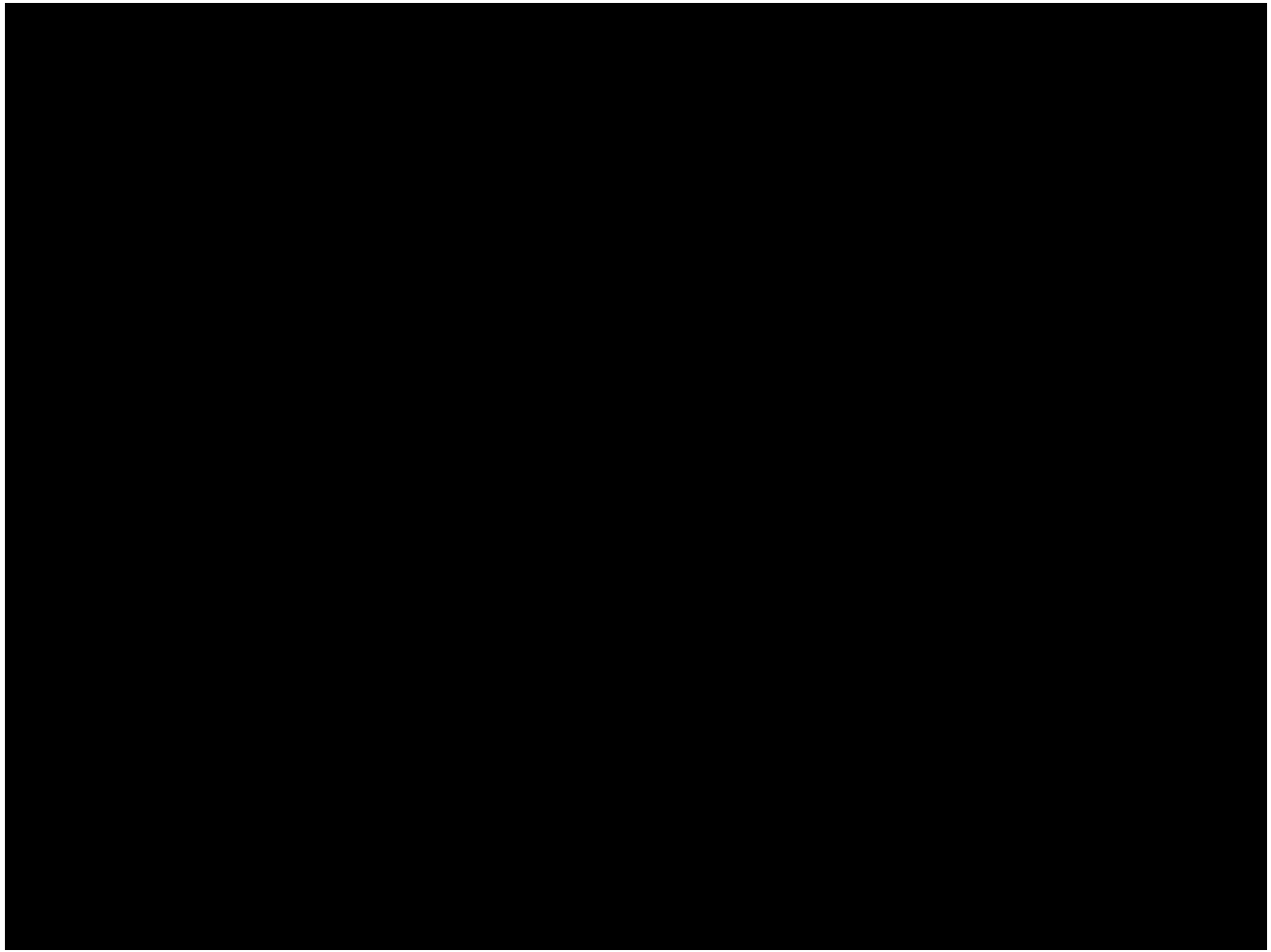
Goal: find pairs of identical socks

```
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                   |
```

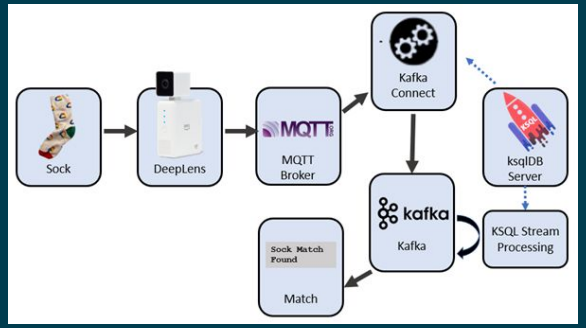
Demo

```
-- Creata stream for the MQTT topic
create stream sock_stream(image varchar, probability double)
with (kafka_topic='data_mqtt',  value_format='json');

-- Bucket sock images into windows of 5 seconds
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)
where image != 'blank'
group by image having count(*) > 3
emit changes;

-- Find pairs of socks (socks appearing in even numbers)
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;
```



What did I learn?

- AWS Deeplens is super cool
- Object detection is a stream of events
- Events are very messy
- IoT architecture with Kafka is really scalable
- Stream processing with ksqlDB is very convenient
- Sock sorting **does not** make you interesting at parties



Q & A

https://github.com/saubury/socksort

linkedin.com/in/**simonaubury**

/thoughtworks

@Simon Aubury