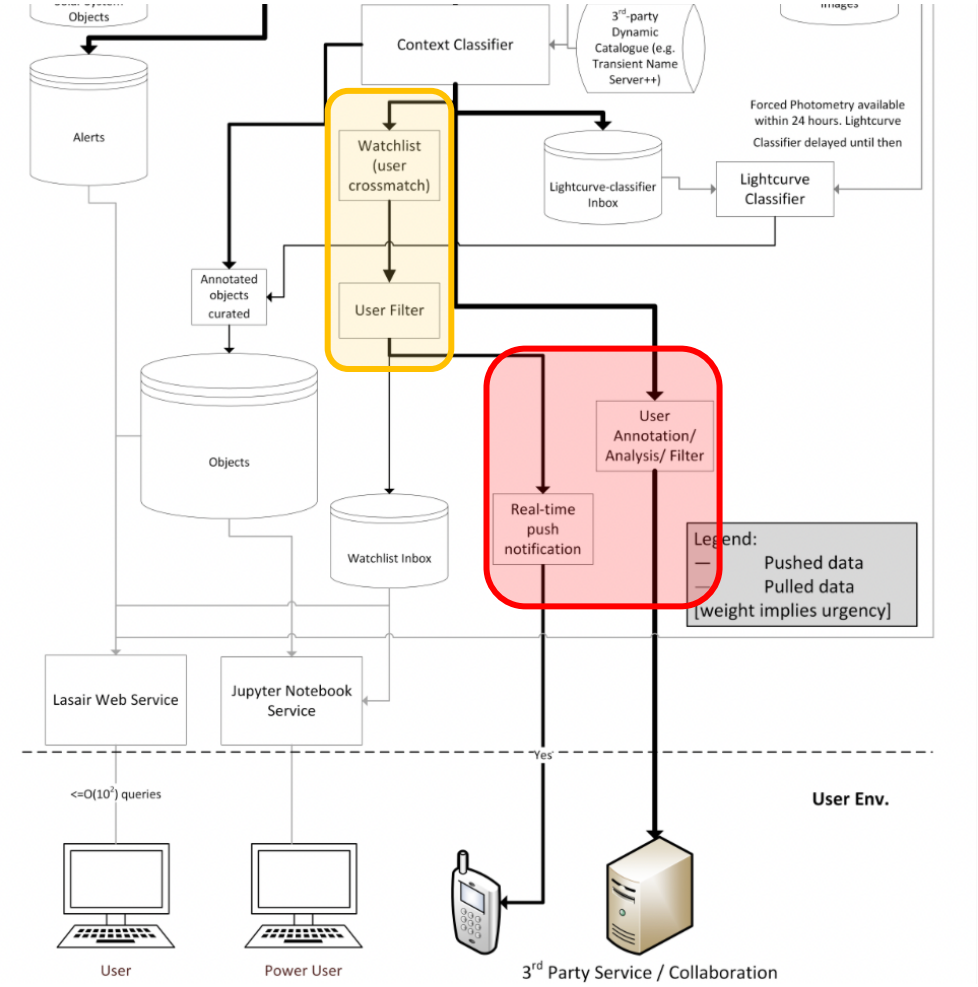
SQL Queries on Data Streams

Lasair Cycle-1 Technology Review

18 March 2020

Where does this fit in?



Needed for:

- Real-time push notification
- Custom event streams

Also a way of implementing:

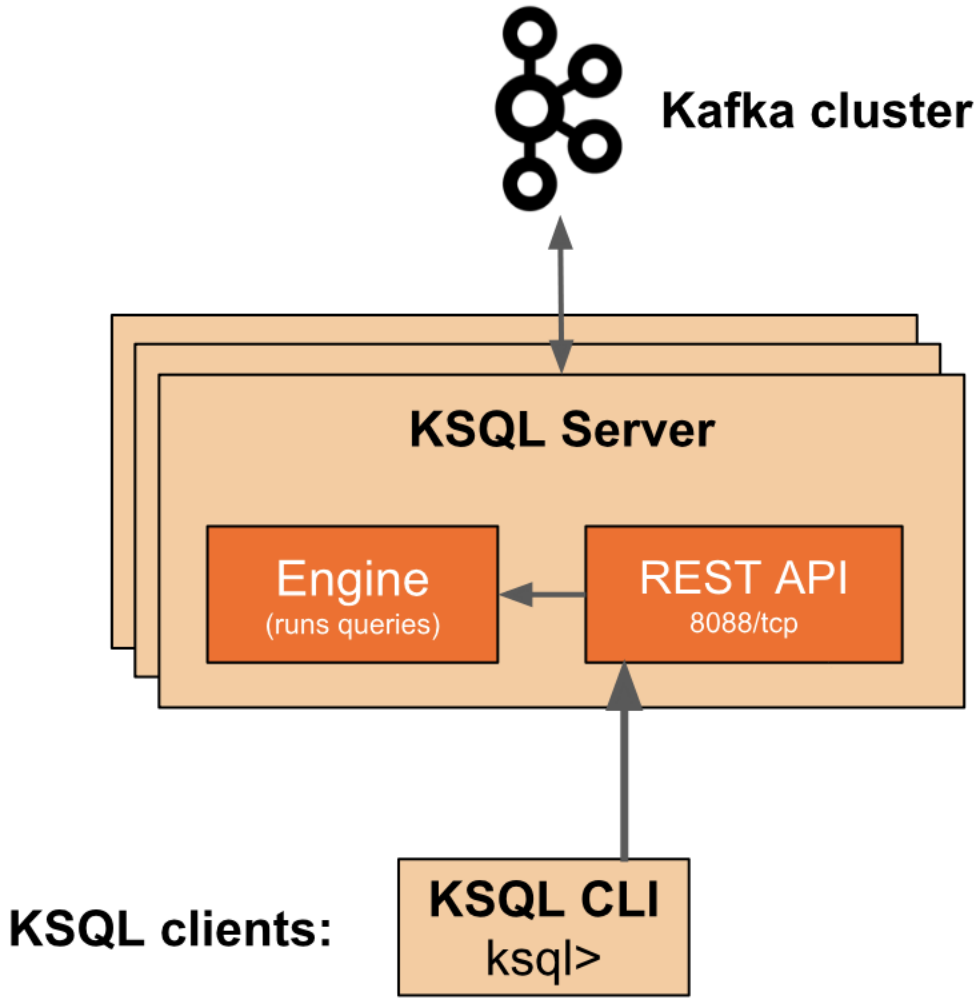
- Watch lists
- User queries

Technology Options

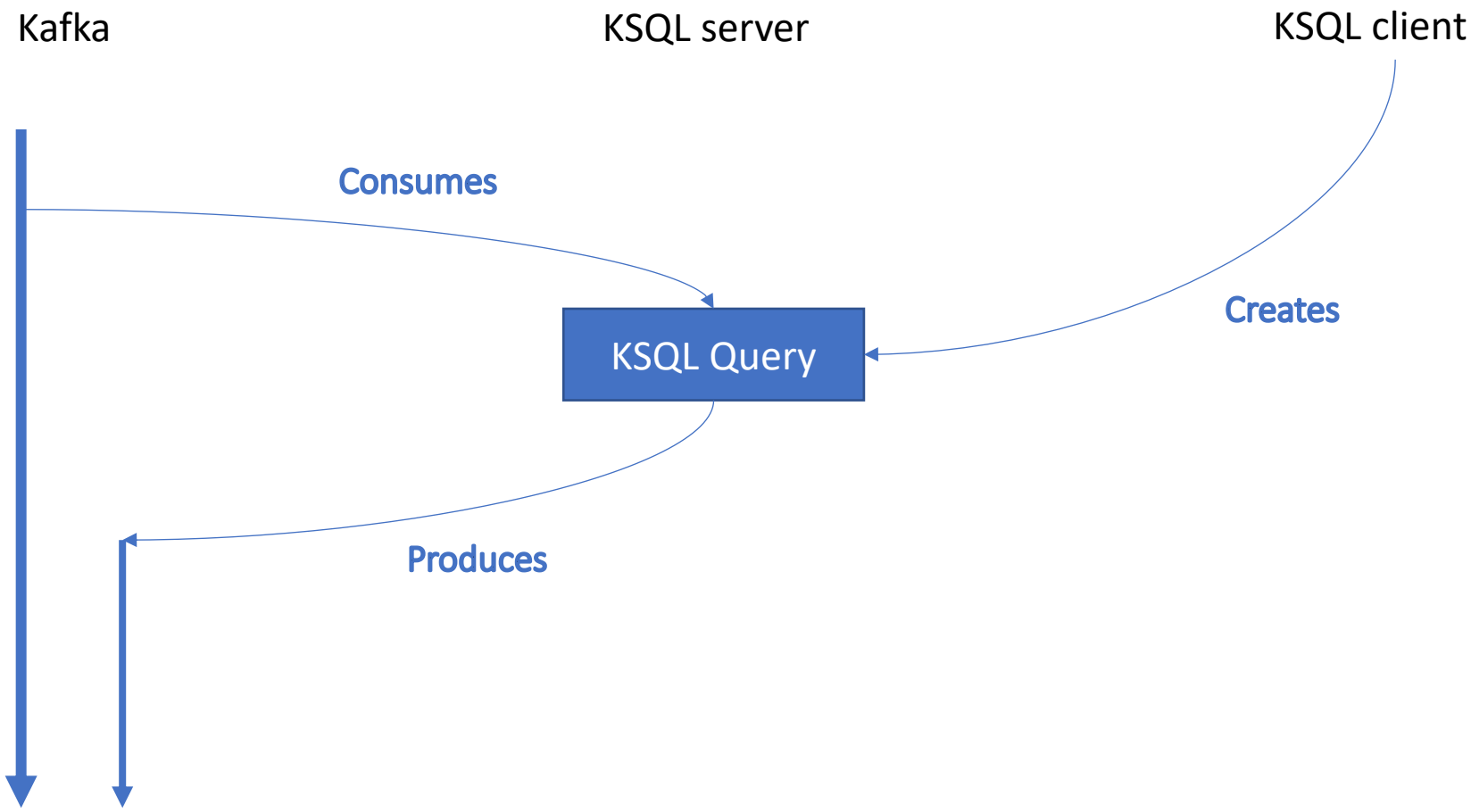
- KSQL
- Temporary MySQL database
- Anything else we should be thinking about?

- Probably not:
  - Apache Drill (not really focused on stream processing)
  - Spark Structured Streaming (too different, too much work for too little gain)

KSQL architecture and components



Typical operation – filtering a stream



Example – describe the data source

```
CREATE STREAM teststream (
     objectId VARCHAR,
     ramean DOUBLE,
     decmean DOUBLE,
     mjdmin DOUBLE,
     mjdmax DOUBLE,
     magrmin DOUBLE,
     latestrmag DOUBLE,
     classification VARCHAR,
     score VARCHAR)
WITH (
     kafka_topic='2SN-likecandidates’,
     value_format='JSON');
```

Example – create a new stream/topic

```
CREATE STREAM sn_test_stream
AS SELECT * FROM teststream
WHERE classification='SN’
      AND ramean > 215 AND ramean < 220
      AND decmean > 75 AND decmean < 80
EMIT CHANGES;
```

Example – prove it did something

```
ksql> SELECT * FROM sn_test_stream;
+------------------+------------------+------------------+------------------+------------------+--------
----------+------------------+------------------+------------------+------------------+----------------
-+
|ROWTIME           |ROWKEY            |OBJECTID          |RAMEAN            |DECMEAN           |MJDMIN
          |MJDMAX            |MAGRMIN           |LATESTRMAG        |CLASSIFICATION    |SCORE
|
+------------------+------------------+------------------+------------------+------------------+--------
----------+------------------+------------------+------------------+------------------+----------------
-+
|1579434266420     |null              |ZTF20aadcdzj      |219.283945995     |79.94889376       |58850.53
703700006 |58867.45336809987 |18.892            |18.9792           |SN                |Not Near PS1 star
|
|1579435142352     |null              |ZTF20aadcdzj      |219.28394712380955|79.9488929952381  |58850.53
703700006 |58867.48990740022 |18.892            |19.0336           |SN                |Not Near PS1 star
|
|1579803073043     |null              |ZTF20aadcdzj      |219.28392438260872|79.94889672173913
|58850.53703700006 |58871.5284258998  |18.892            |19.1958           |SN                |Not Near
PS1 star |
|1579874718652     |null              |ZTF20aadcdzj      |219.28391969166668|79.94889701666666
|58850.53703700006 |58872.56950230012 |18.892            |19.1625           |SN                |Not Near
PS1 star |
^CQuery terminated
```

Example – from Kafka

```
~$ kafkacat -C -b lasair-dev.roe.ac.uk:9092 -t SN_TEST_STREAM -o beginning -c 1
```

{"OBJECTID":"ZTF18acbwaxk","RAMEAN":186.55033410652817,"DECMEAN":58.3141267943620 1,"MJDMIN":58586.18103010021,"MJDMAX":58867.43031249987,"MAGRMIN":15.6157,"LATESTRMAG":18.9523,"CLASSIFICATION":"SN","SCORE":"Not Near PS1 star"}

MySQL – Outline workflow

- Kafka consumer reads a (small) batch of alerts
- Writes them to a temporary (probably in memory) MySQL table
- Run the SQL query on the temporary table
- Kafka producer writes the query result (to a different topic)

MySQL vs KSQL

- MySQL syntax virtually identical to querying the object DB whereas KSQL requires minor additions/changes
  - But semantics still differ so some translation likely still required in any case
- KSQL avoids writing custom code (but not a lot anyway)
- KSQL deployment is slightly more complex (but not very)
- MySQL approach could potentially integrate with other components of Lasair if it makes sense
  - -> less load on the Kafka server and less network traffic
- MySQL requires some thought as to how to avoid data loss on errors

Concluding Thoughts

- We have two possible approaches to solving the problem, either of which should work
- Which one we choose may depend on whether we (want to) end up with a microservice architecture or not