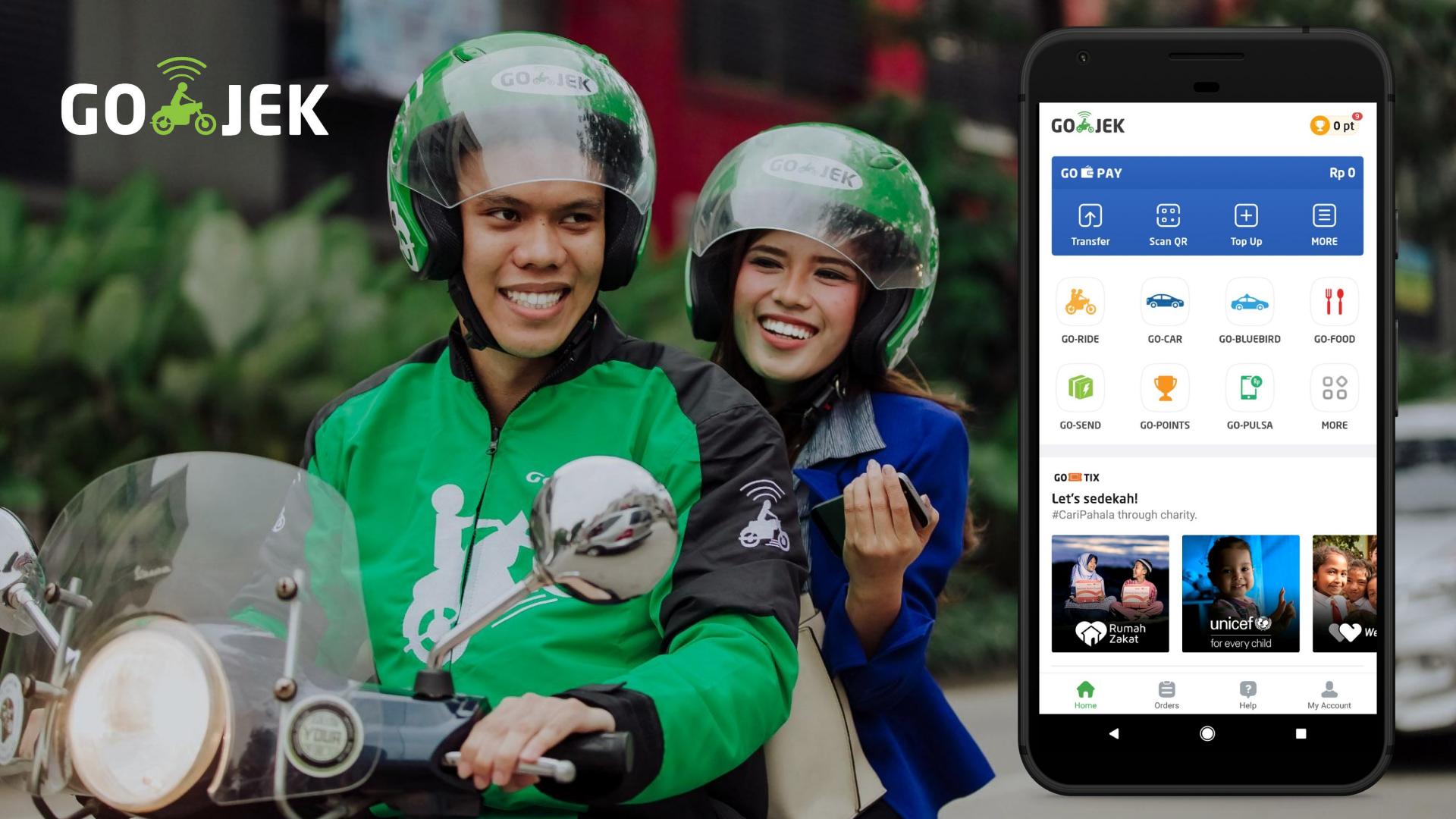
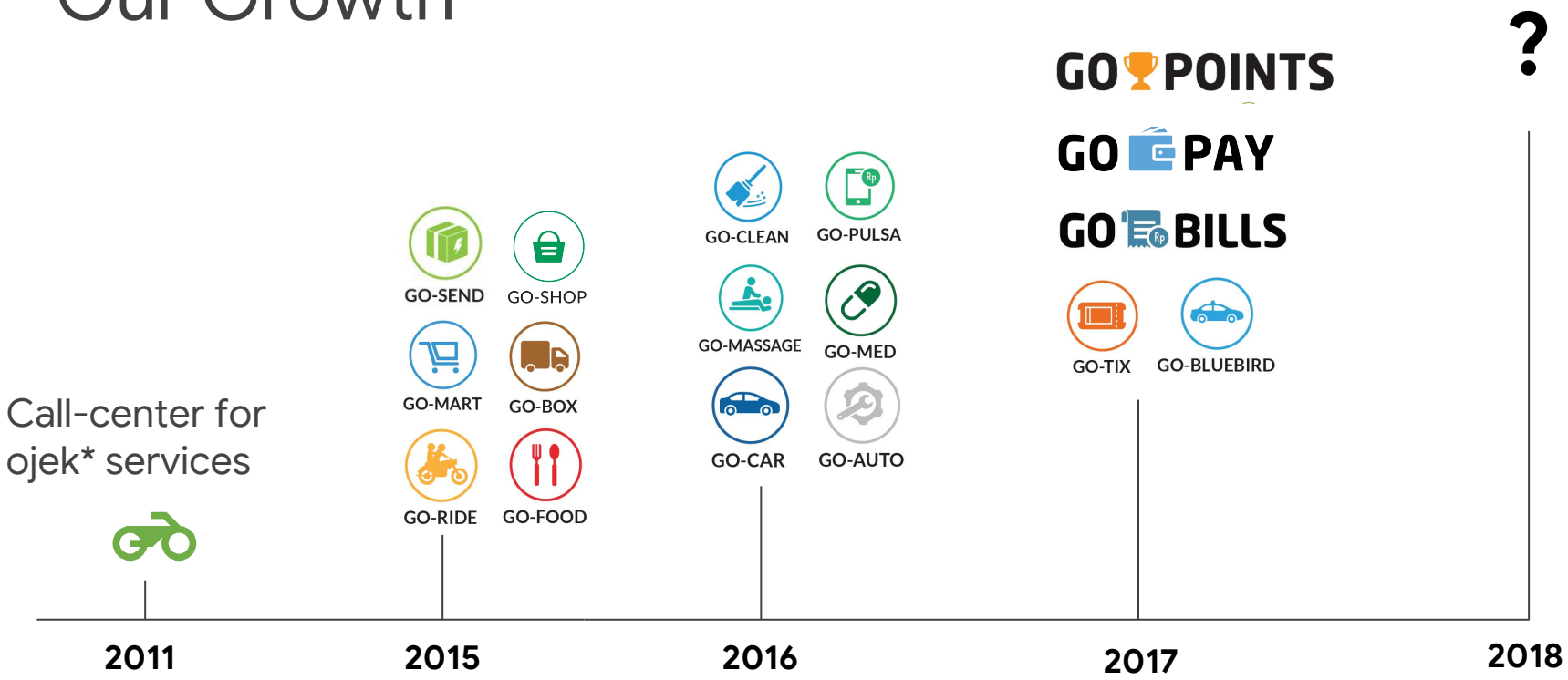
Building a feature platform to scale machine learning

Willem Pienaar
Data Science Platform Lead
GO-JEK

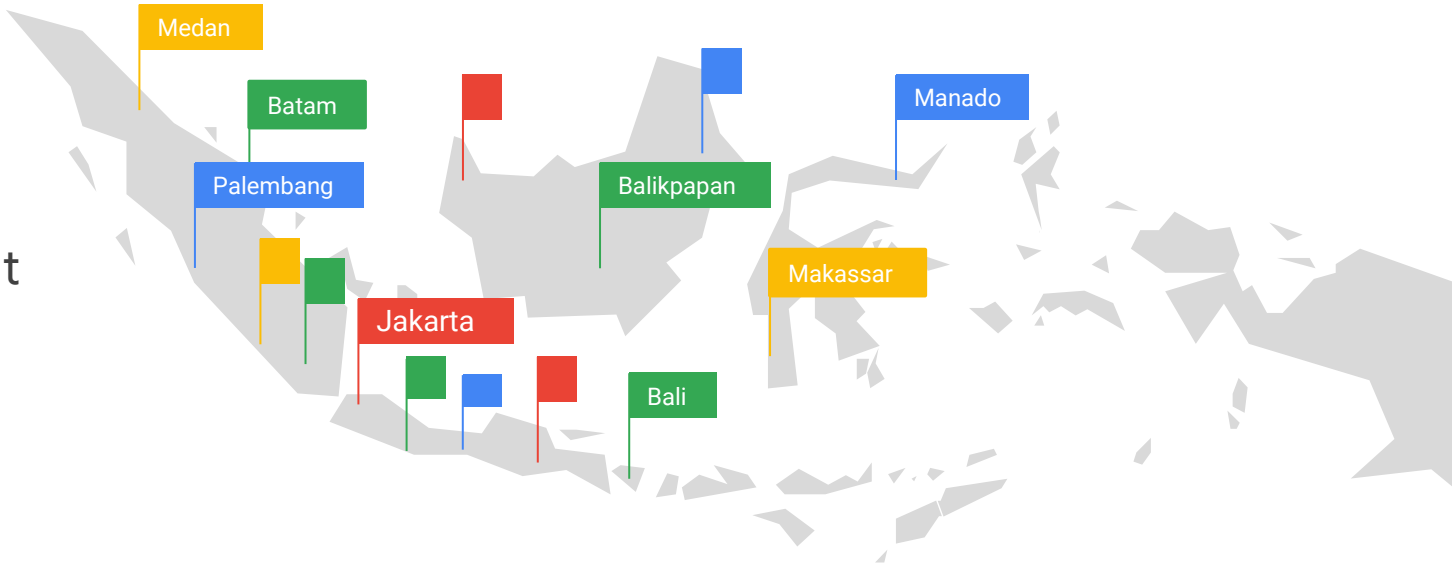

GO JEK
GO JEK
0 pt
GO PAY
Rp 0
Transfer
Scan QR
Top Up
MORE
GO-RIDE
GO-CAR
GO-BLUEBIRD
GO-FOOD
GO-SEND
GO-POINTS
GO-PULSA
MORE
GO TIX
Let's sedekah!
#CariPahala through charity.
Rumah Zakat
unicef
for every child
Home
Orders
Help
My Account

Our Growth


Call-center for ojek* services
GO-SEND
GO-SHOP
GO-MART
GO-BOX
GO-RIDE
GO-FOOD
GO-CLEAN
GO-PULSA
GO-MASSAGE
GO-MED
GO-CAR
GO-AUTO
GO POINTS
GO PAY
GO BILLS
GO-TIX
GO-BLUEBIRD
?
2011
2015
2016
2017
2018

Our Home

Operating in 60 cities throughout Indonesia



80m app downloads

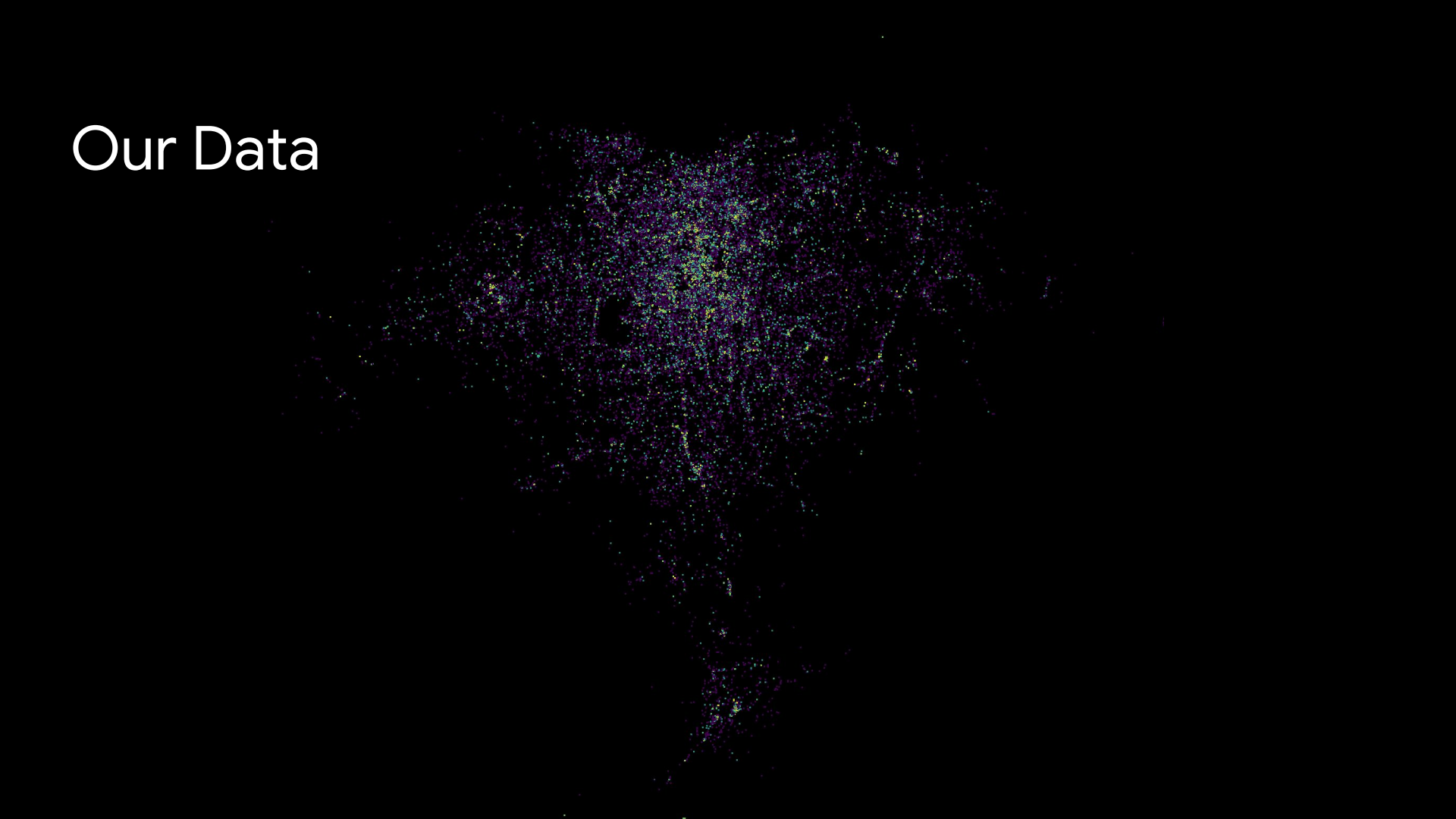
+200k merchants

60 cities

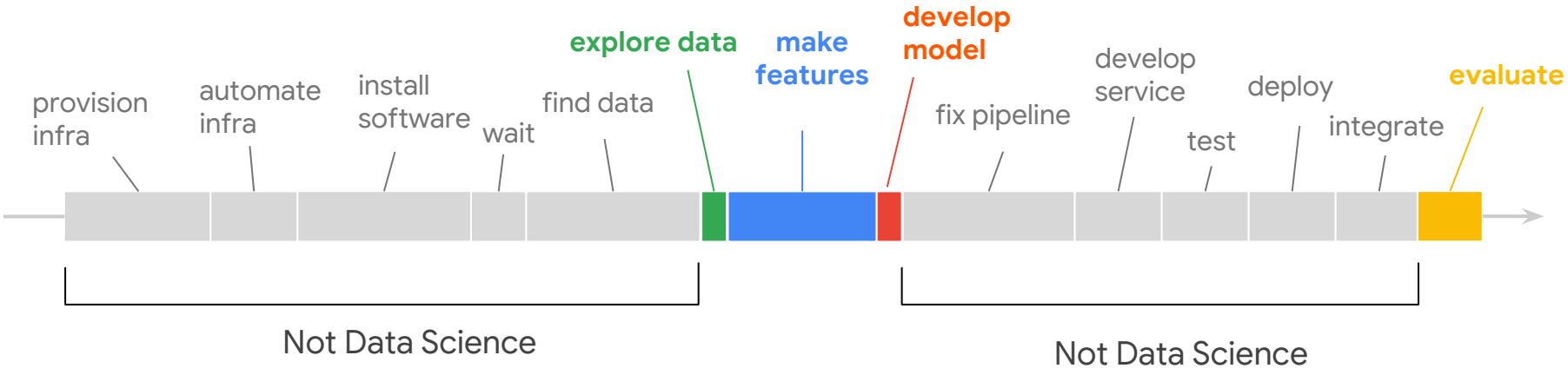
1m+ drivers

100m+ monthly bookings

Our Data



Where do data scientists spend their time?



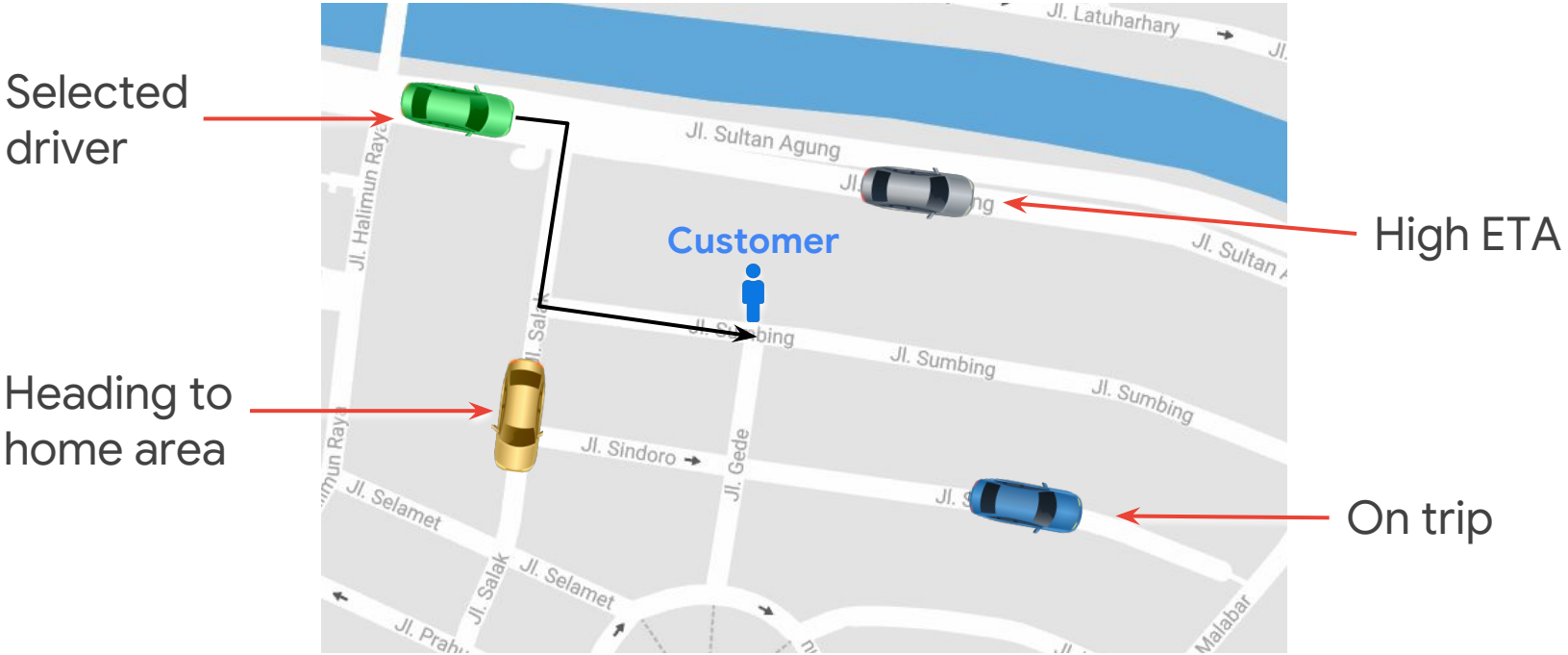
The problems with feature engineering

- Pipeline jungles
- Data processing did not scale
- Real-time features required engineers
- Inconsistency between training and serving
- Lack of discovery
- Lack of standardization

What should a feature platform allow us to do?

- Standardize feature definitions
- Provide a means for creating batch and streaming features
- Allow us to create datasets to train our ML models
- Allow model services to access features in production
- Allow for the discovery of features
- Abstract away engineering

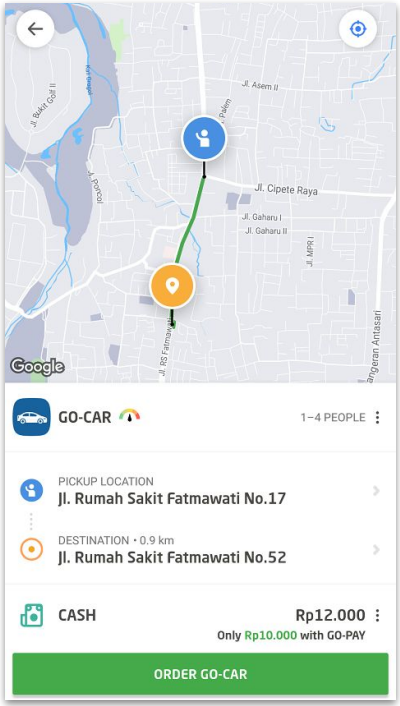
Which driver should we send?



Which features do we need?

Input Features

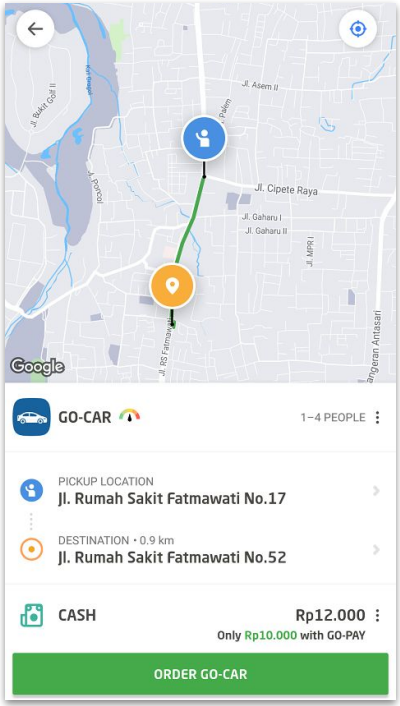
- Driver location, speed, direction, and ETA to customer
- Time of day, day of week
- Demand, supply
- Origin, destination
- Customer profile, actions
- And hundreds of other features



Which features types should we support?

Feature types

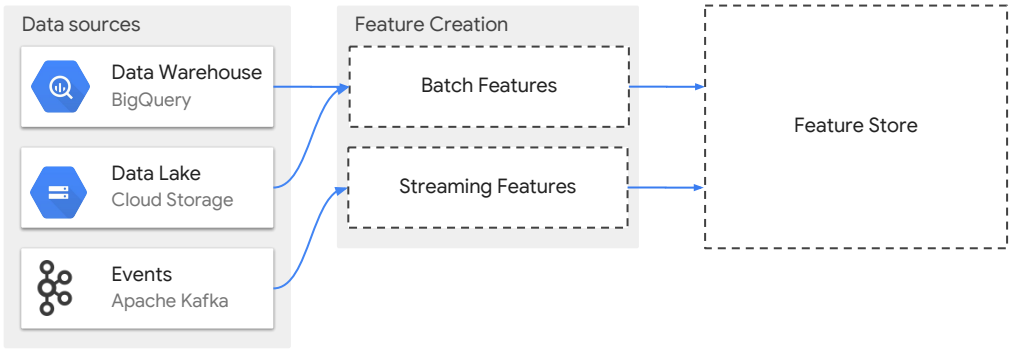
- Entity (driver, customer, area)
- Process (batch, real-time)
- Granularity (day, hour, minute, second)
- Value type (primitive, complex)



Feature specifications

```
name: booking_count
owner: example@go-jek.com
description: Total number of booking created per day per driver
uri: https://yourdomain.com/organization/feature-transforms/driver-feature-pipeline
granularity: DAY
valueType: INT64
entity: driver
tags:
  - driver
  - booking
  - streaming
dataStores:
  serving:
    id: driver_serving
  warehouse:
    id: driver_warehouse
```

Two types of feature creation processes



The two main types of feature creation workflows

1. Batch workflow
2. Stream workflow

Publishing batch features

Workflow

1. Create parameterised SQL query

```
SELECT
 driver_id,
 count(price_of_item) as booking_count
FROM driver_bookings
WHERE
  _PARTITIONTIME >= dateadd(@execution_date, -1, day)
AND
  _PARTITIONTIME < @execution_date
```

Publishing batch features

Workflow

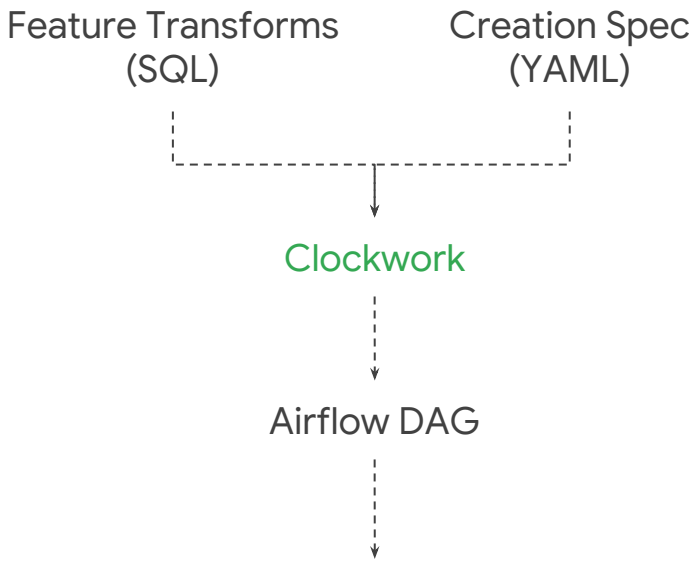
1. Create parameterised SQL query
2. Configure creation specification

```
owner: owner@go-jek.com
startDate: 2018-07-01
catchup: false
interval: "0 0 * * *"
entity: driver
granularity: DAY
path: stable/driver/day/bookings.sql
columns:
  -   name: completed_bookings
      featureId: driver.day.completed_bookings_v1
  -   name: cancelled_bookings
      featureId: driver.day.cancelled_bookings_v1
```

Publishing batch features

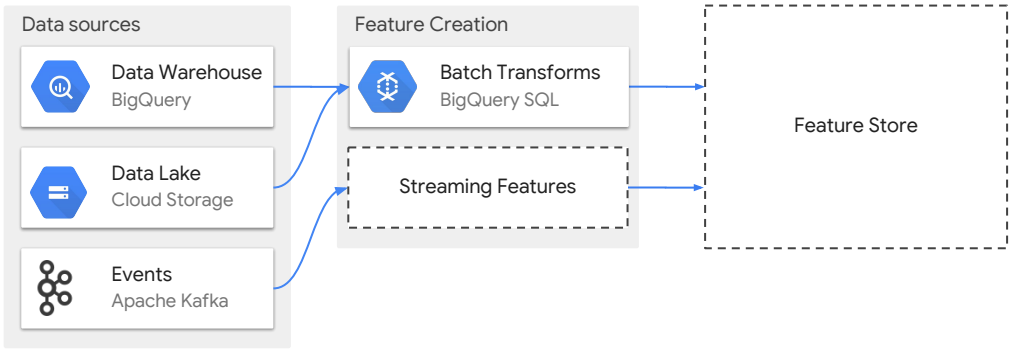
Workflow

1. Create parameterised SQL query
2. Configure creation specification
3. Push to repo, trigger Clockwork

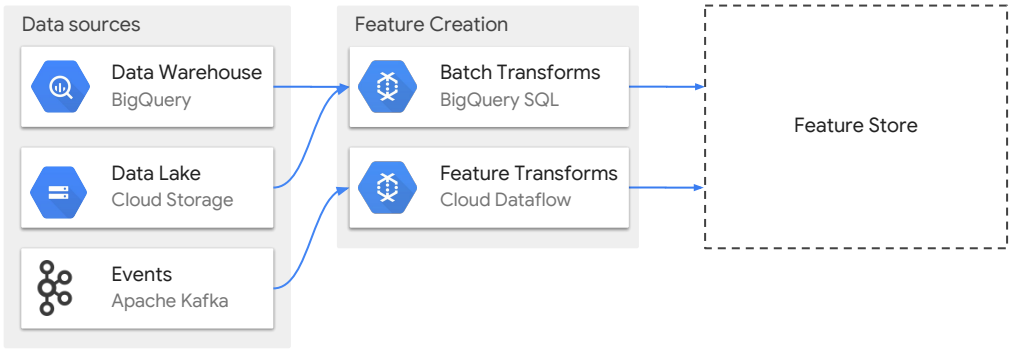


Clockwork
Declarative Airflow workflows through YAML

How do we publish real-time features?



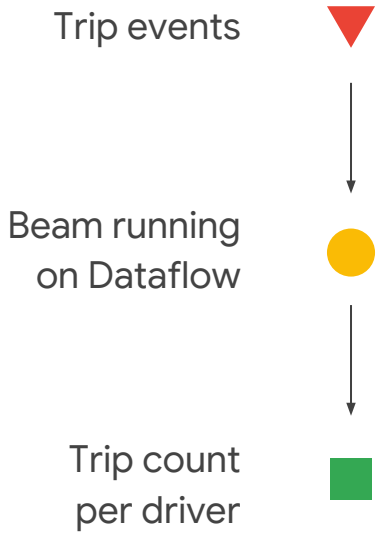
Publishing streaming features



Apache Beam

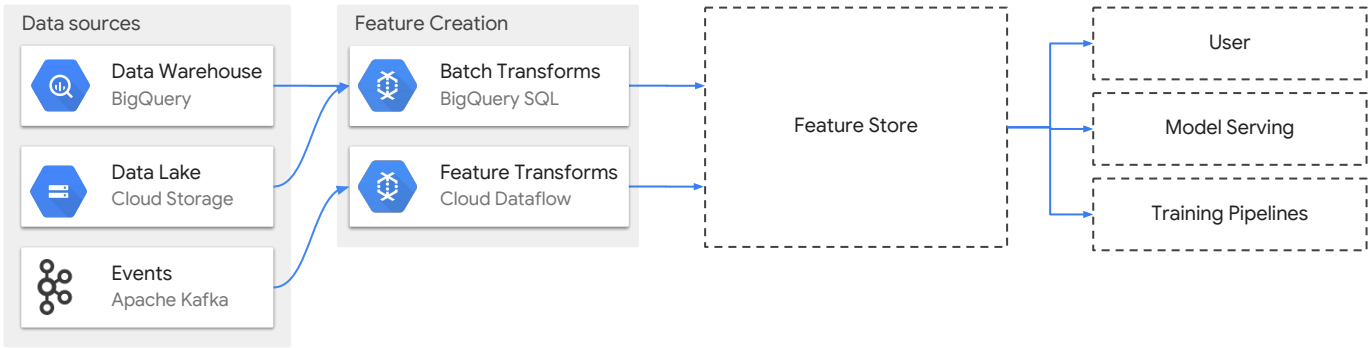
- Unified batch and stream support
- Consistent feature definition between serving and training
- Automatic scaling with Dataflow
- No lock-in

Defining a feature in Beam




```
trip_count_per_driver = (
    trip_events
    | 'filter' >> beam.Filter(lambda trip: trip['status'] == 'success')
    | 'pair' >> beam.Map(lambda driver: (driver, 1))
    | 'sum' >> beam.CombinePerKey(sum))
```

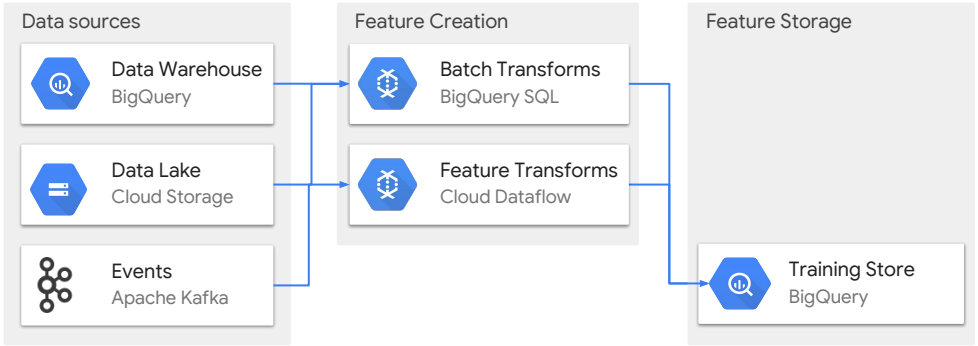
How will our clients use the feature store?



Training store requirements

- Scalable big data processing
- Handle large and sparse key spaces
- Joins
- Easy to use

BigQuery for storing our training data

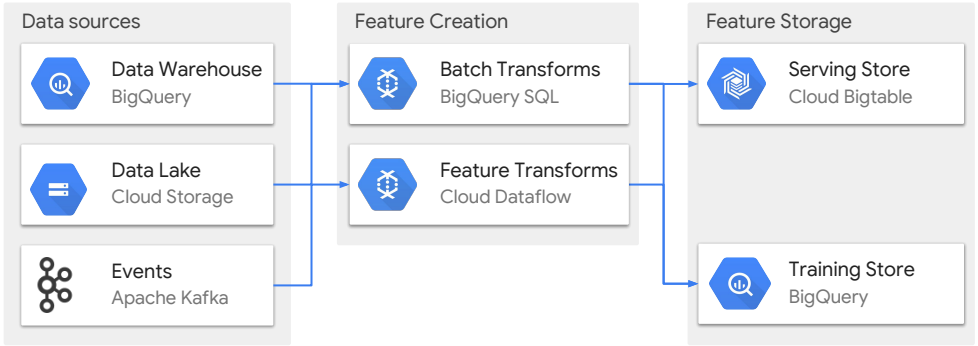


- No infrastructure
- Easy to use (SQL)
- Scales to massive amounts of data
- Integrated with GCP services
- Already contains our labelled data

Serving store requirements

- Low latency reads (<10 ms)
- High throughput reads/writes (150k+ rps)
- Large volume of feature data (TBs)
- Persistence
- Linearly scalable

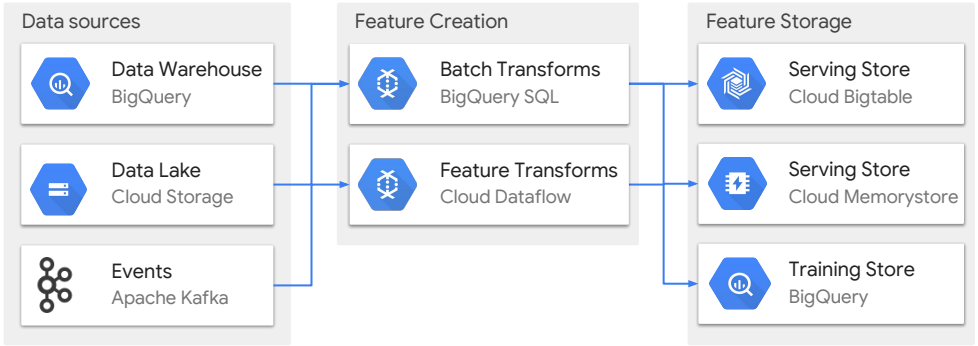
Bigtable as primary store for serving features



Bigtable

- 10k RPS read/write per node
- 10 ms latency read/write
- No infrastructure
- Scalable
- Large capacity per node (TB+)
- Persistent

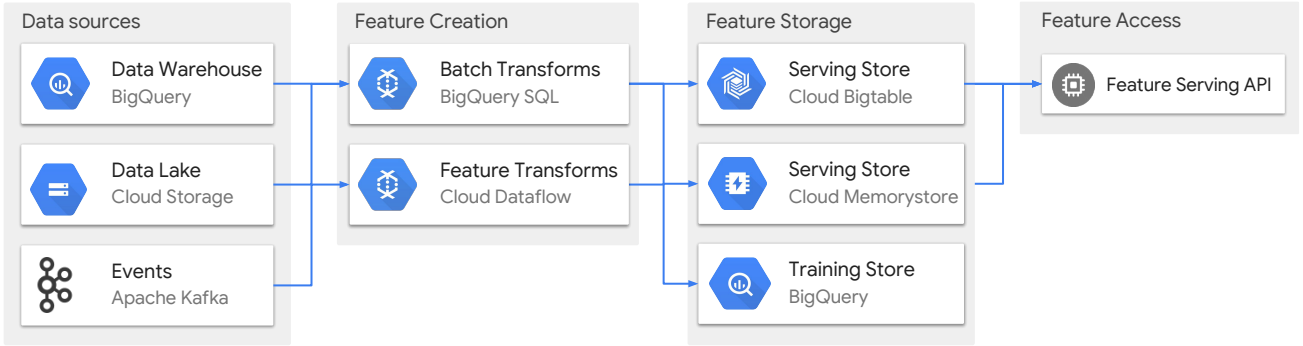
Redis as secondary store for serving features



Redis

- Extremely high throughput
- Very low latency read/write

Build a feature serving API



Serving API

- Load balancing
- Caching
- Fail-over
- QoS
- Authentication

Extending the feature platform

```
name: booking_summary
owner: example@go-jek.com
granularity: DAY
valueType: BYTES
entity: driver
dataStores:
  serving:
    id: driver_serving
  warehouse:
    id: driver_warehouse
```

Extending the feature platform

```
name: booking_summary
owner: example@go-jek.com
granularity: DAY
valueType: BYTES
entity: driver
options:
  protoValueType: com.gojek.ds.driver.BookingSummary
  JSONInWarehouse: True
dataStores:
  serving:
    id: driver_serving
  warehouse:
    id: driver_warehouse
```

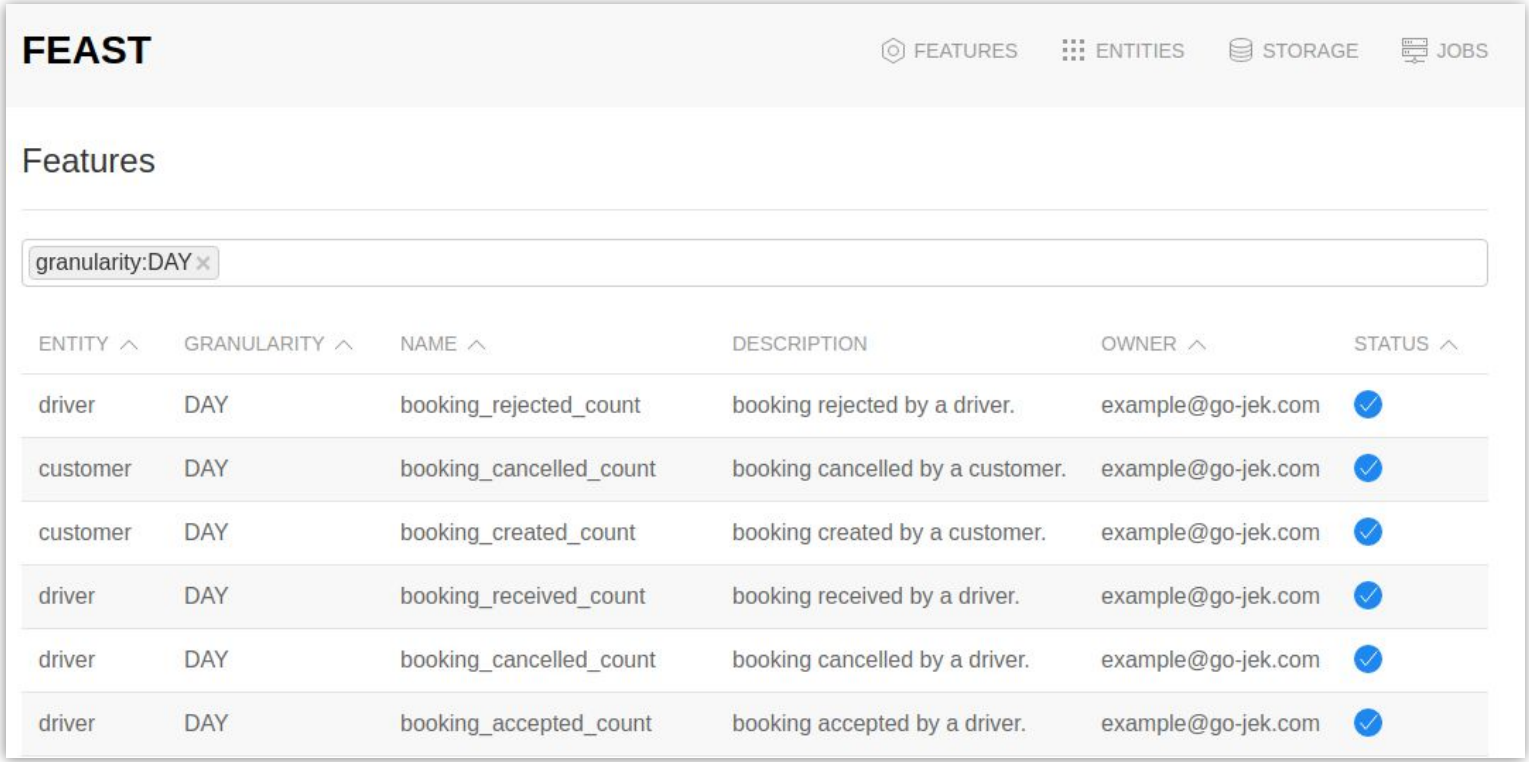
Will decode bytes into JSON before storing in BigQuery

Extending the feature platform

```
name: booking_count
owner: example@go-jek.com
granularity: DAY
valueType: INT64
entity: driver
options:
  minimumDiscreteValue: 0
  maximumDiscreteValue: 1000
  warnOnFailure: True
dataStores:
  serving:
    id: driver_serving
  warehouse:
    id: driver_warehouse
```

Logs a warning if the feature value is outside of a bound

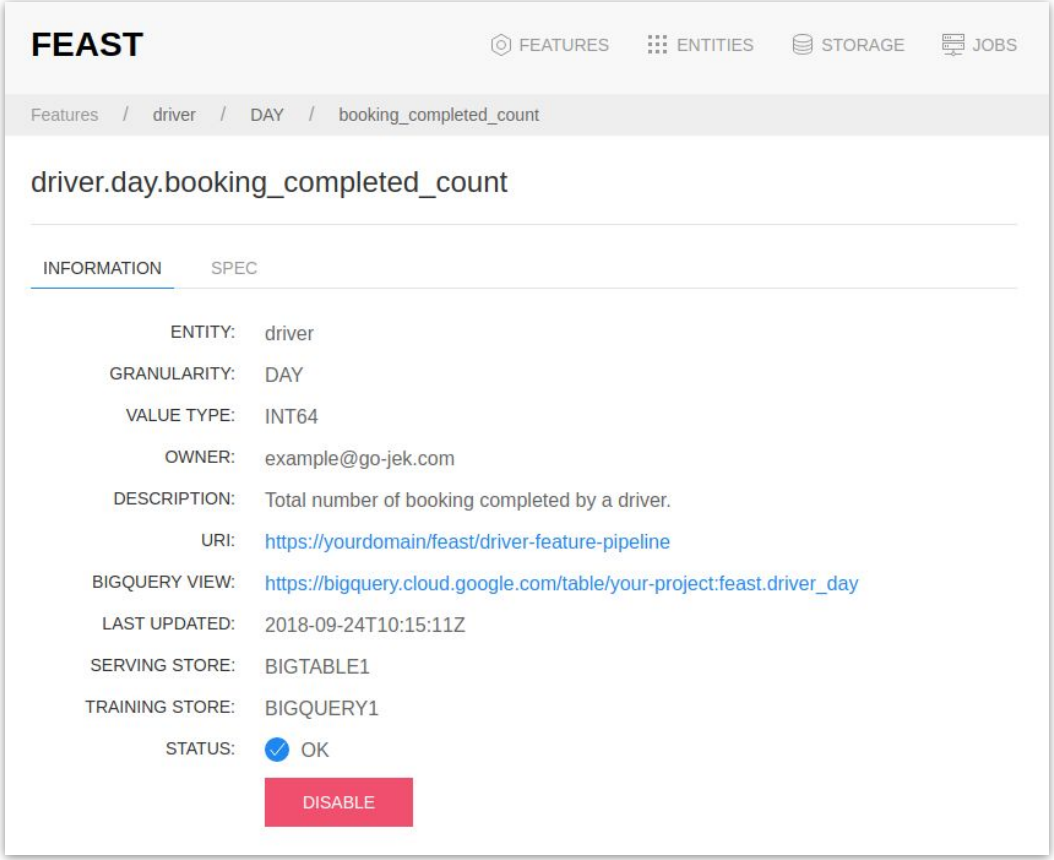
Feature explorer



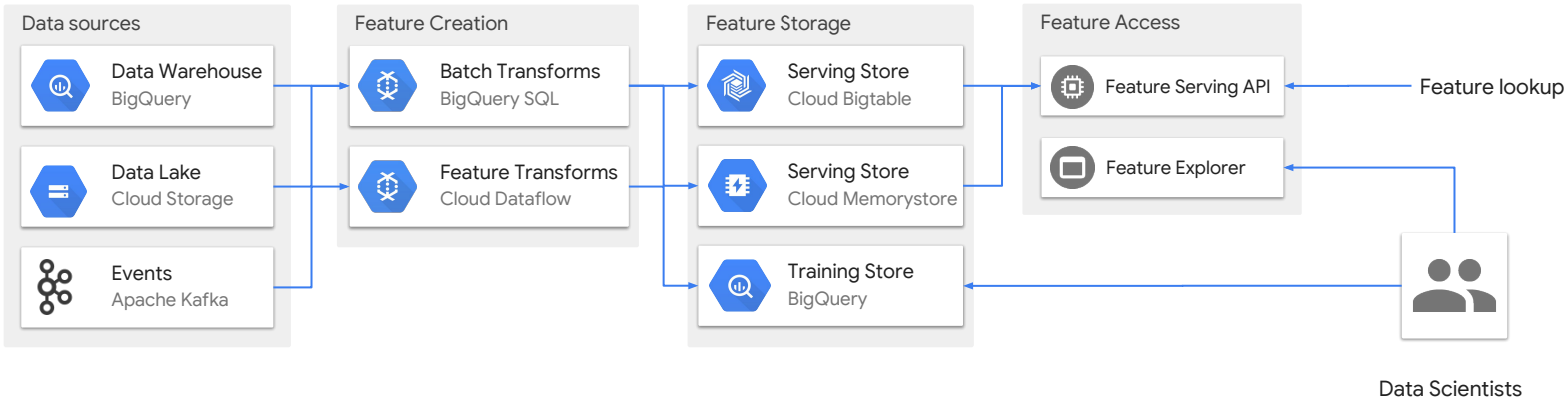
Feature explorer

What is it used for?

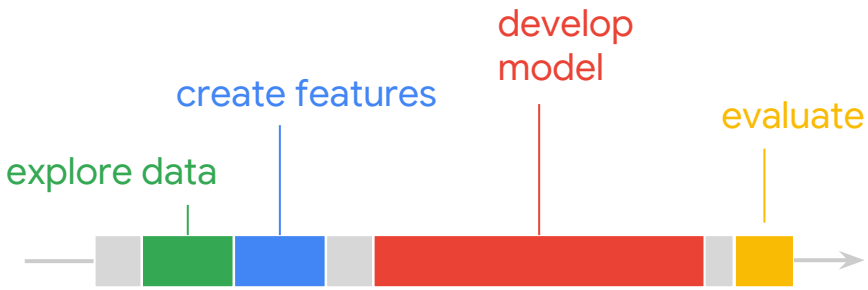
- Feature discovery
- Feature management
- Performance statistics
- Usage statistics
- Traceability
- Job execution
- System health



Feature platform



How are they spending their time now?



Impact on GO-JEK

- Faster time to market for ML projects
- Improved customer experience
- Less data scientists per customer
- Less infrastructure

Thank you!
@will.pienaar