AthenaX: Streaming Processing Platform @Uber

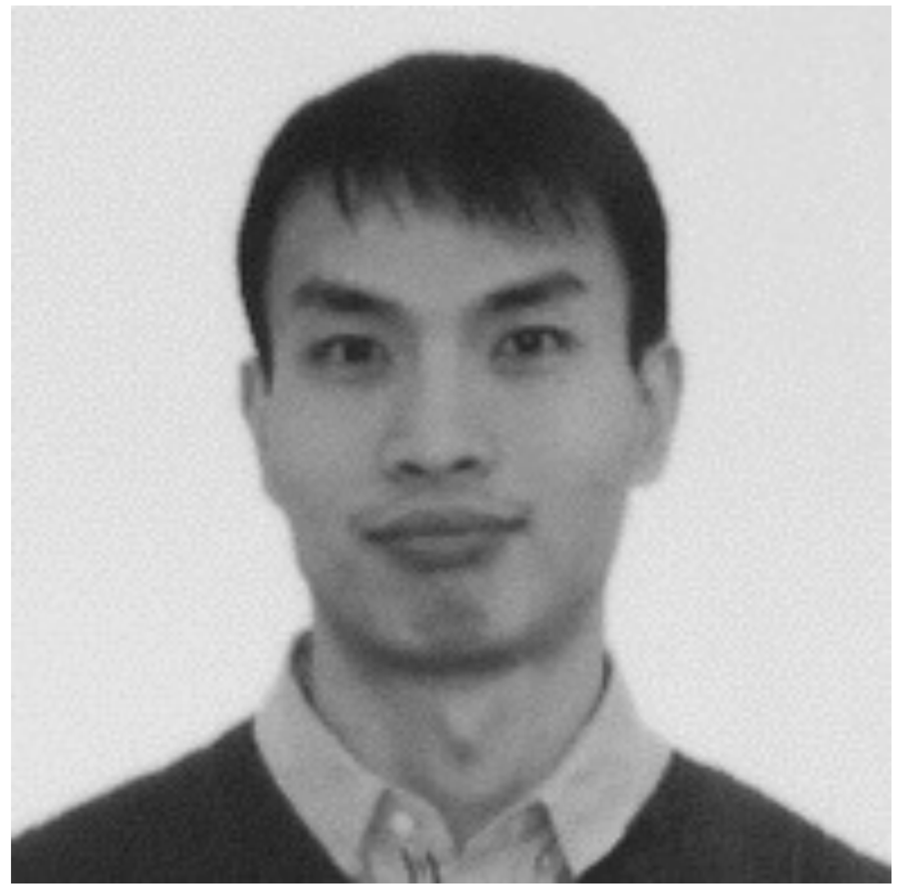
Bill Liu, Haohui Mai

APRIL 11, 2017

Speakers

Bill Liu

- Senior Software Engineer @ Uber

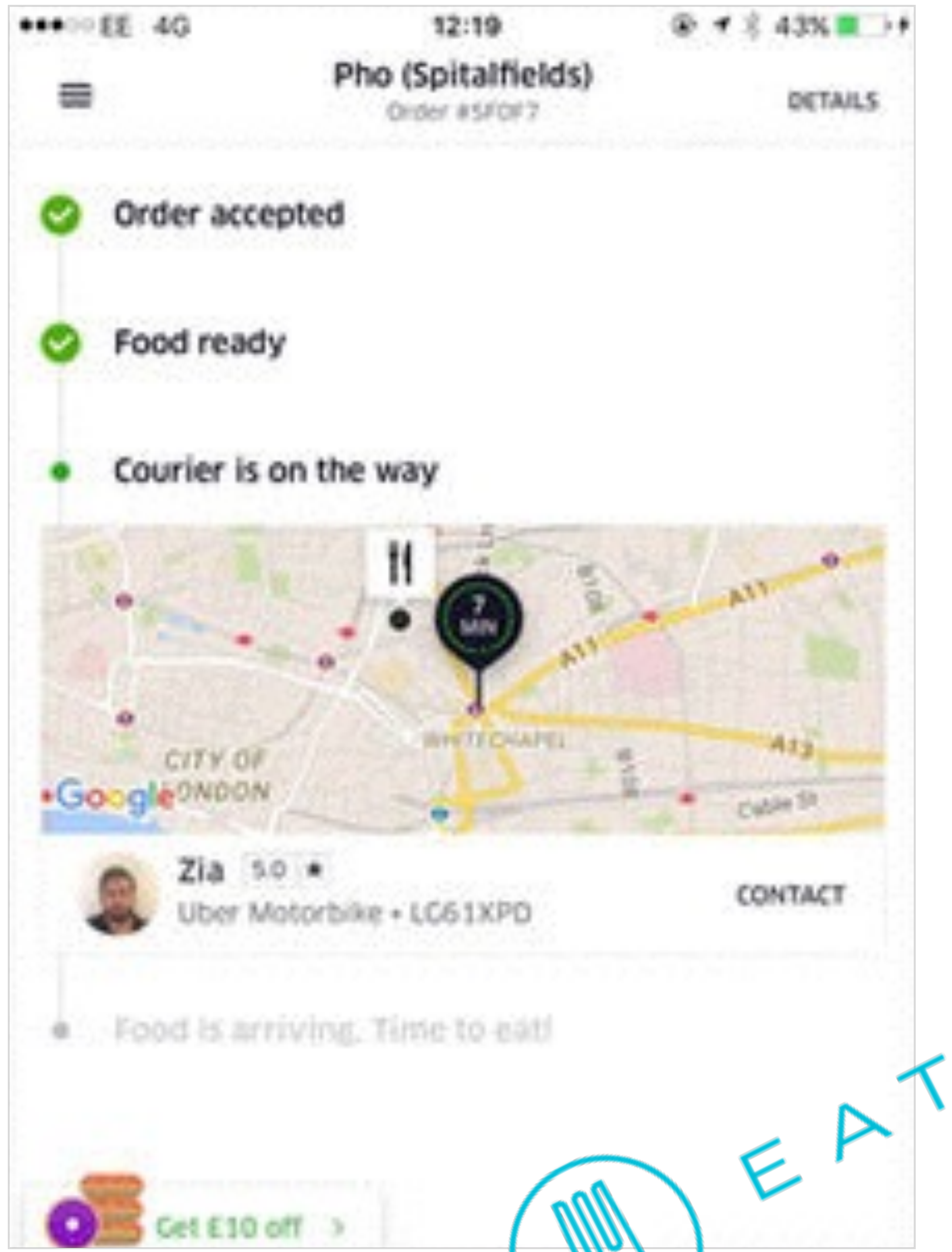


Haohui Mai, @wheat9

- Senior Software Engineer @ Uber
- PMC, Apache Hadoop & Storm

Uber business is real-time

- Uber: Transport A → B on demand reliably
- Dynamic marketplace
- Example: UberEATS



●●●○○ EE 4G 12:19 43%

Pho (Spitalfields)
Order #5F0F7
DETAILS

Order accepted

Food ready

Courier is on the way

Zia 5.0 ★
Uber Motorbike • LC61XPD
CONTACT

Food is arriving. Time to eat!

Get £10 off ›

EATS

Challenges

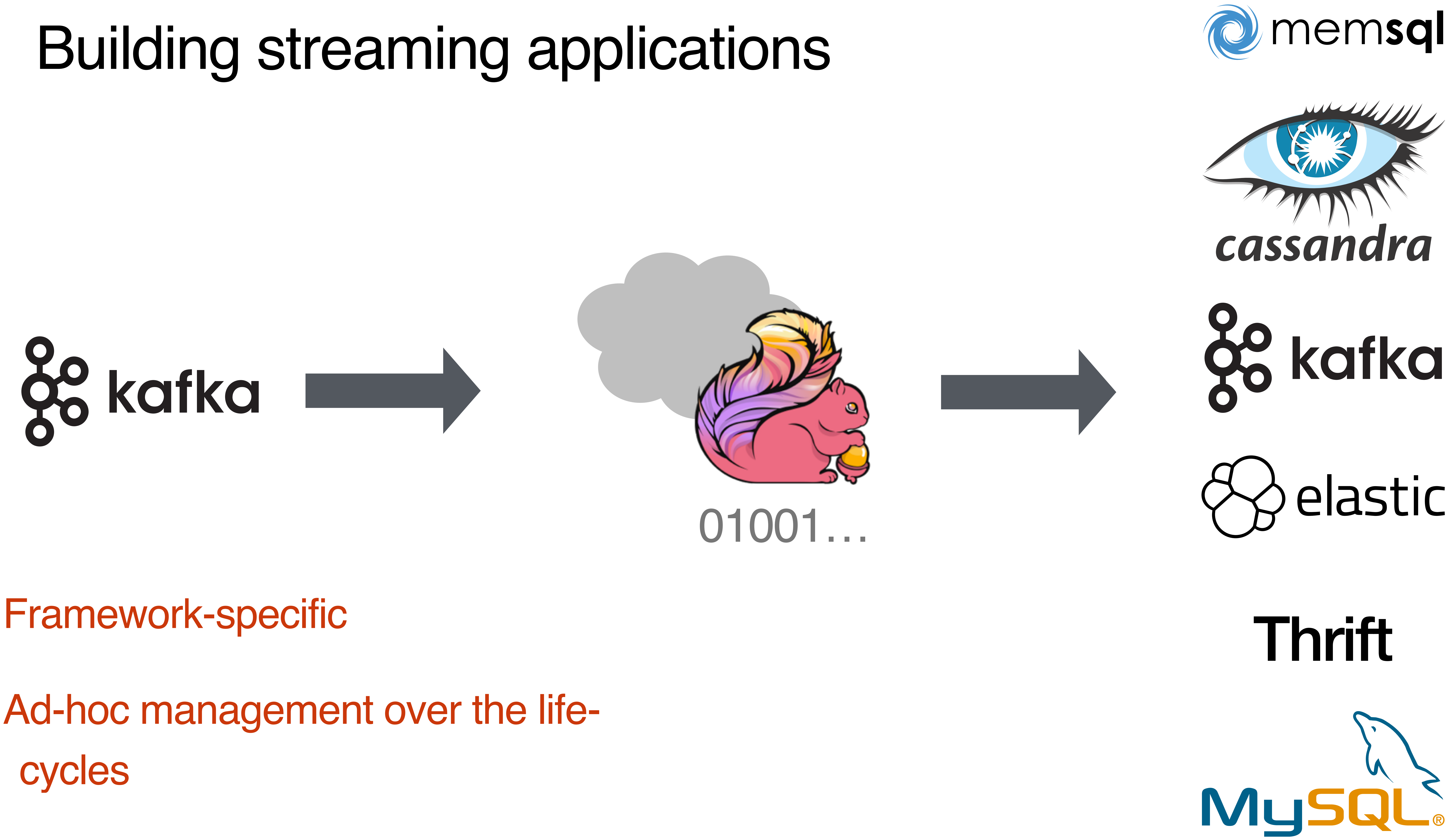
Infra.: Reliability & scalability

- 99.99% SLA on latency
- At-least-once processing
- Billions of messages
- Multiple PB / day

Solutions: Productivity

- Audiences: majority of employees use SQL actively
- Abstractions: Flink / DSL?
- Integrations: data management, monitoring, reporting, etc.

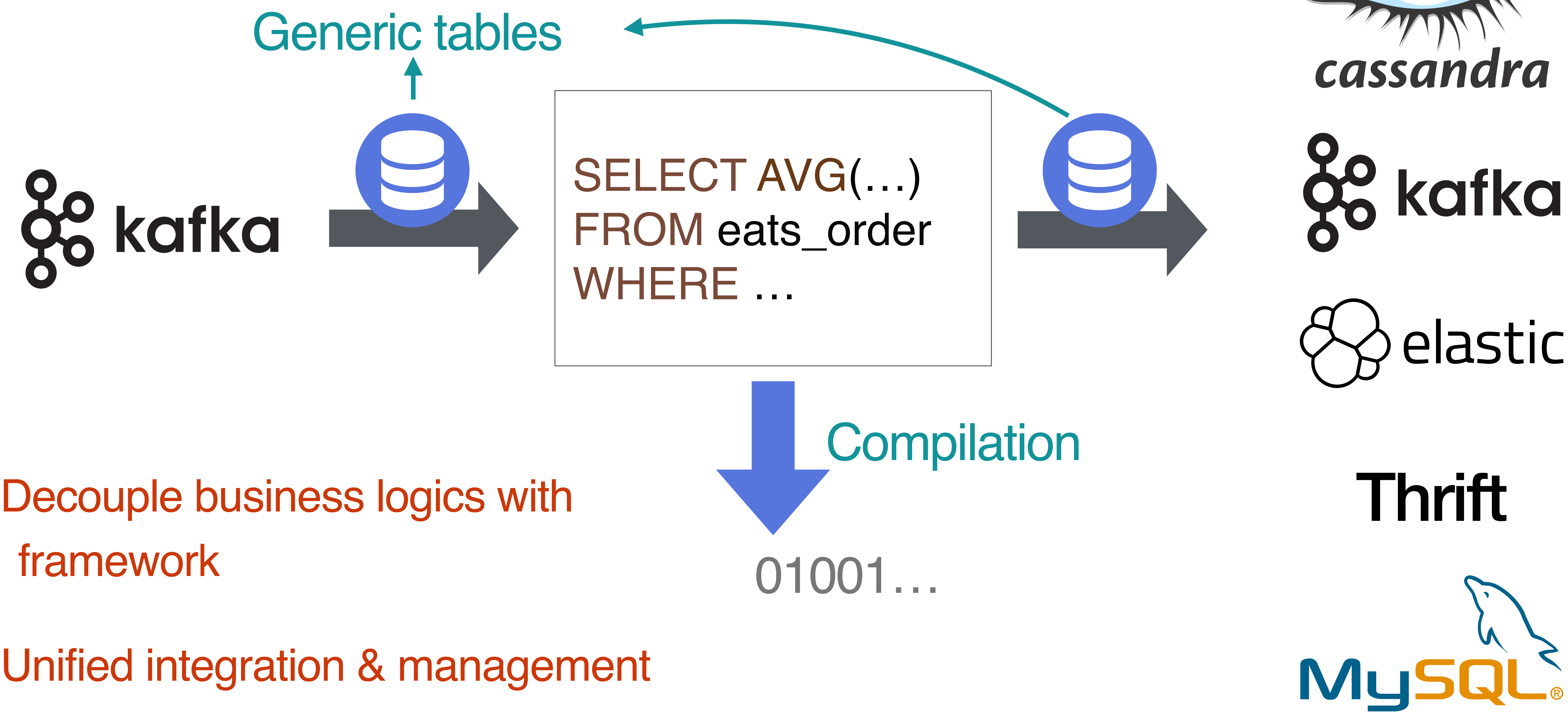
Building streaming applications



- Framework-specific
- Ad-hoc management over the life-cycles

The AthenaX approach

Write SQLs to build streaming applications



```
SELECT AVG(…)
FROM eats_order
WHERE …
```

- Decouple business logics with framework
- Unified integration & management

AthenaX: Streaming processing platform @ Uber

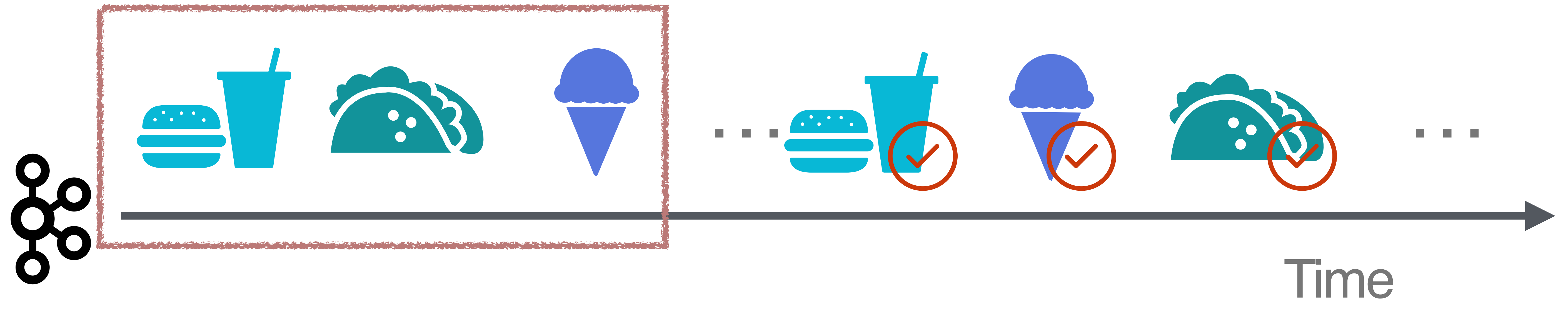
- Write SQLs to build streaming applications
  - Insight: generic table
- Reliable, scalable processing based on Apache Flink
- Develop & deploy streaming applications in production in hours instead of weeks

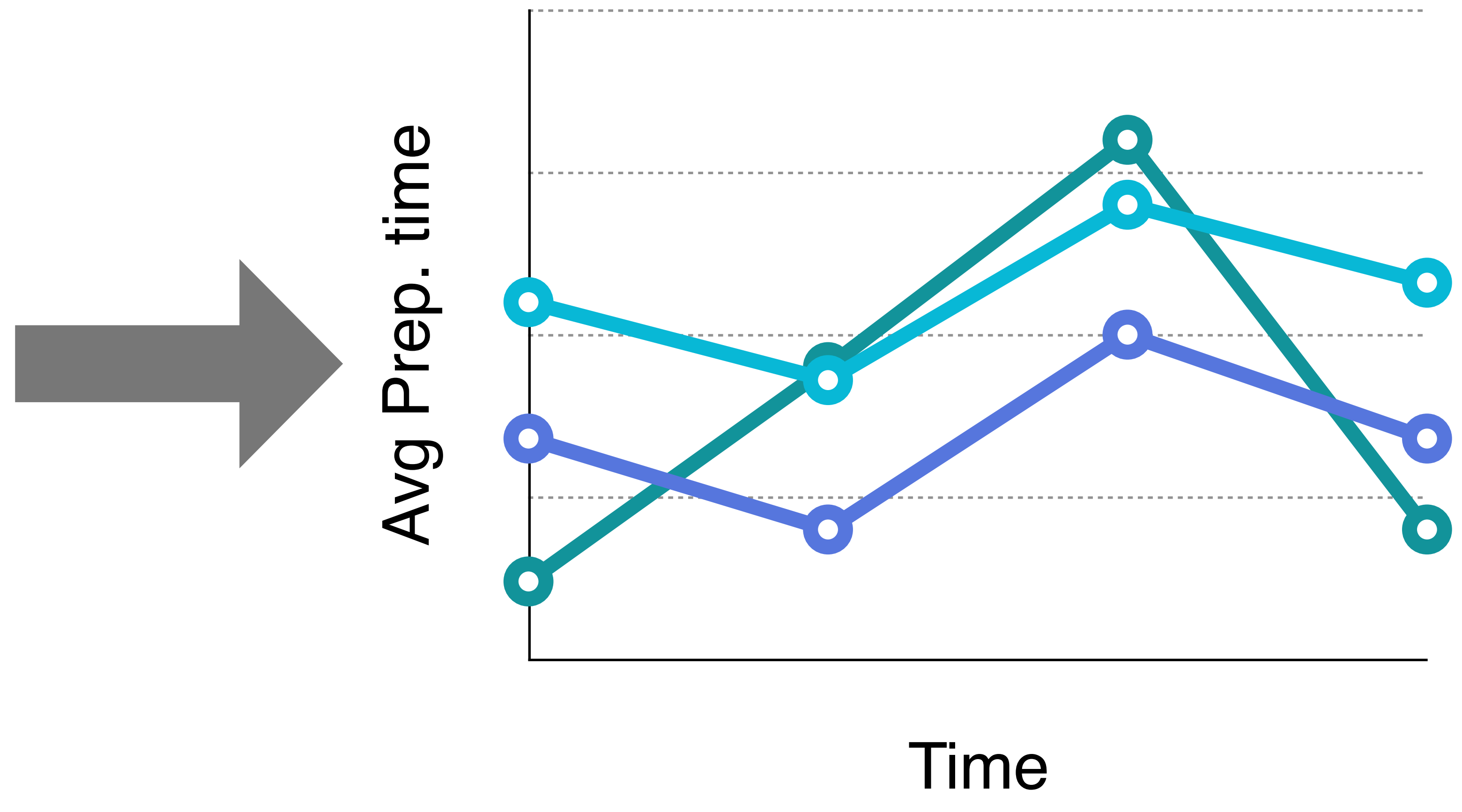
Agenda

- Motivating example
- Case study: ETD in UberEATS
- Implementation
- Current status
- Conclusion

Example

Real-time dashboard for restaurants





```
SELECT meal_id, AVG(meal_prep_time)
FROM eats_order

GROUP BY meal_id, HOP(proctime(),
  INTERVAL ‘1’ MINUTE,
  INTERVAL ‘15’ MINUTE)
```

Example (cont.)

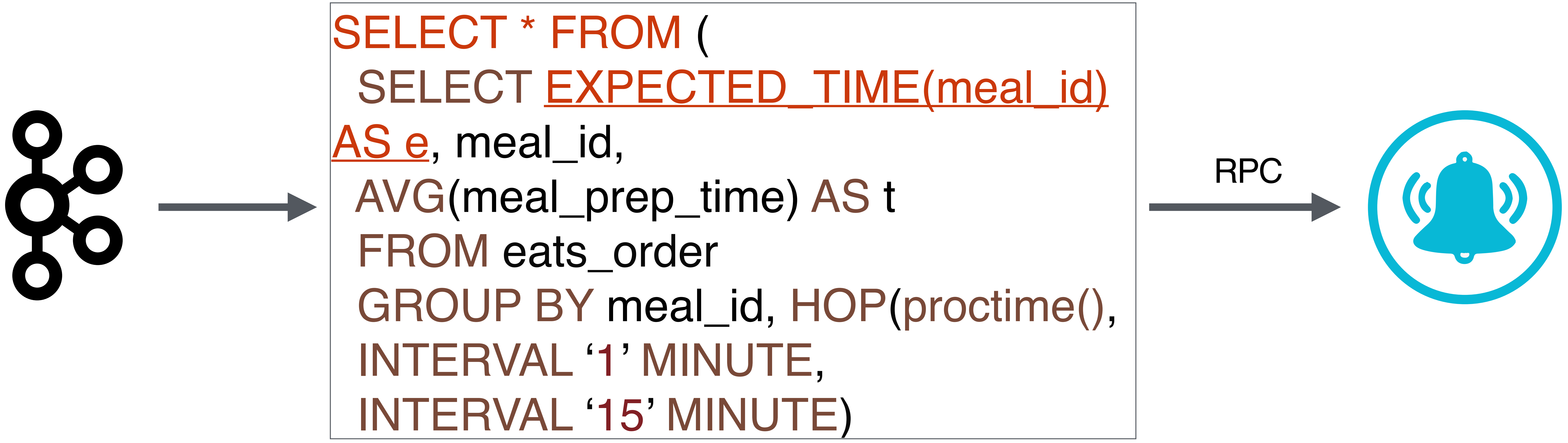
Building streaming processing applications with SQL

```
SELECT AVG(meal_prep_time) FROM
eats_order

GROUP BY meal_id, HOP(proctime(),
  INTERVAL ‘1’ MINUTE,
  INTERVAL ‘15’ MINUTE)
```

Example (cont.)

Building streaming processing applications with SQL



```
SELECT * FROM (
  SELECT EXPECTED_TIME(meal_id)
AS e, meal_id,
  AVG(meal_prep_time) AS t
  FROM eats_order
  GROUP BY meal_id, HOP(proctime(),
  INTERVAL '1' MINUTE,
  INTERVAL '15' MINUTE)
```

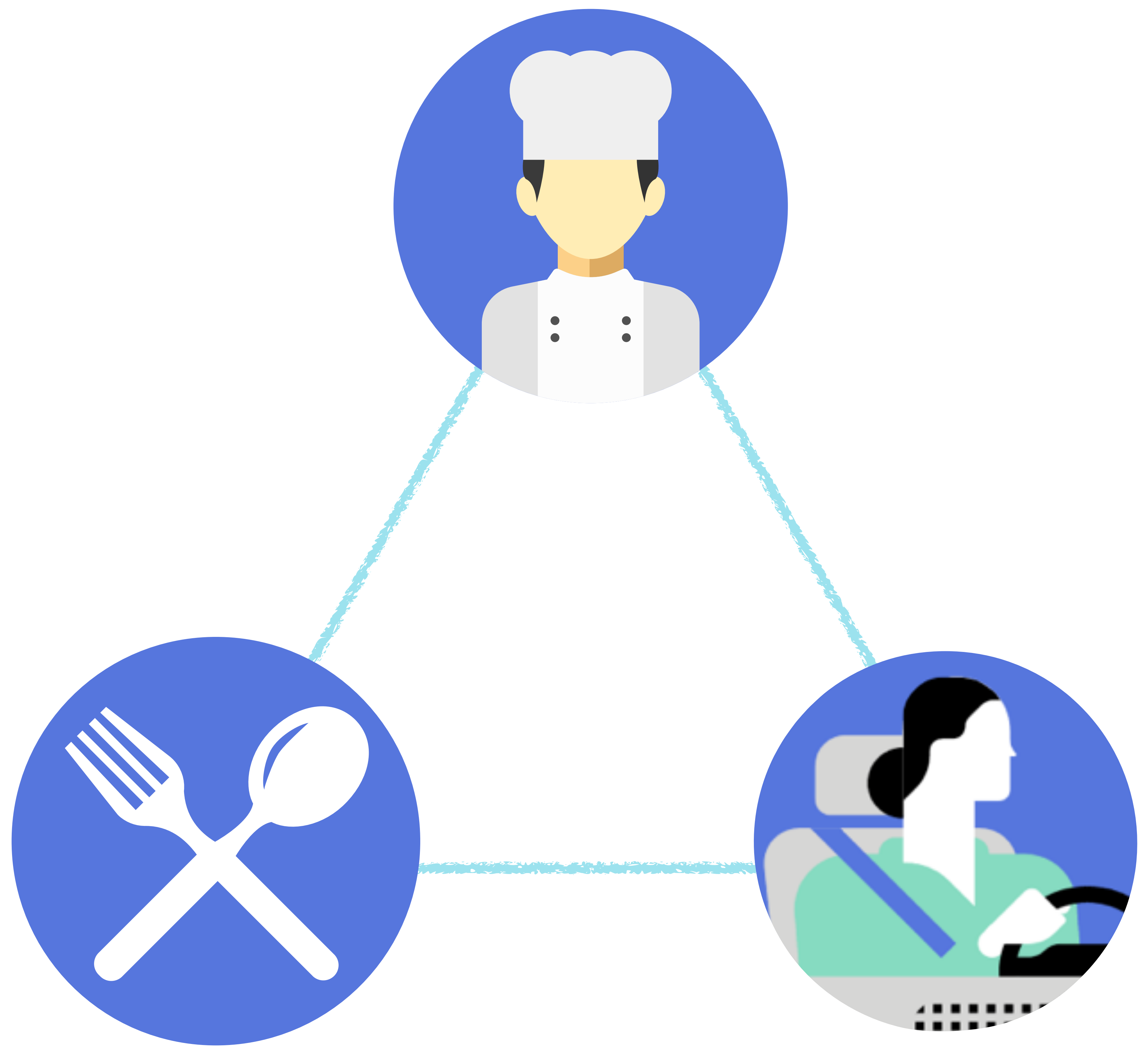
Tables are more generic than analytical stores

Agenda

- Motivating example
- Case study: ETD in UberEATS
- Implementation
- Current status
- Conclusion

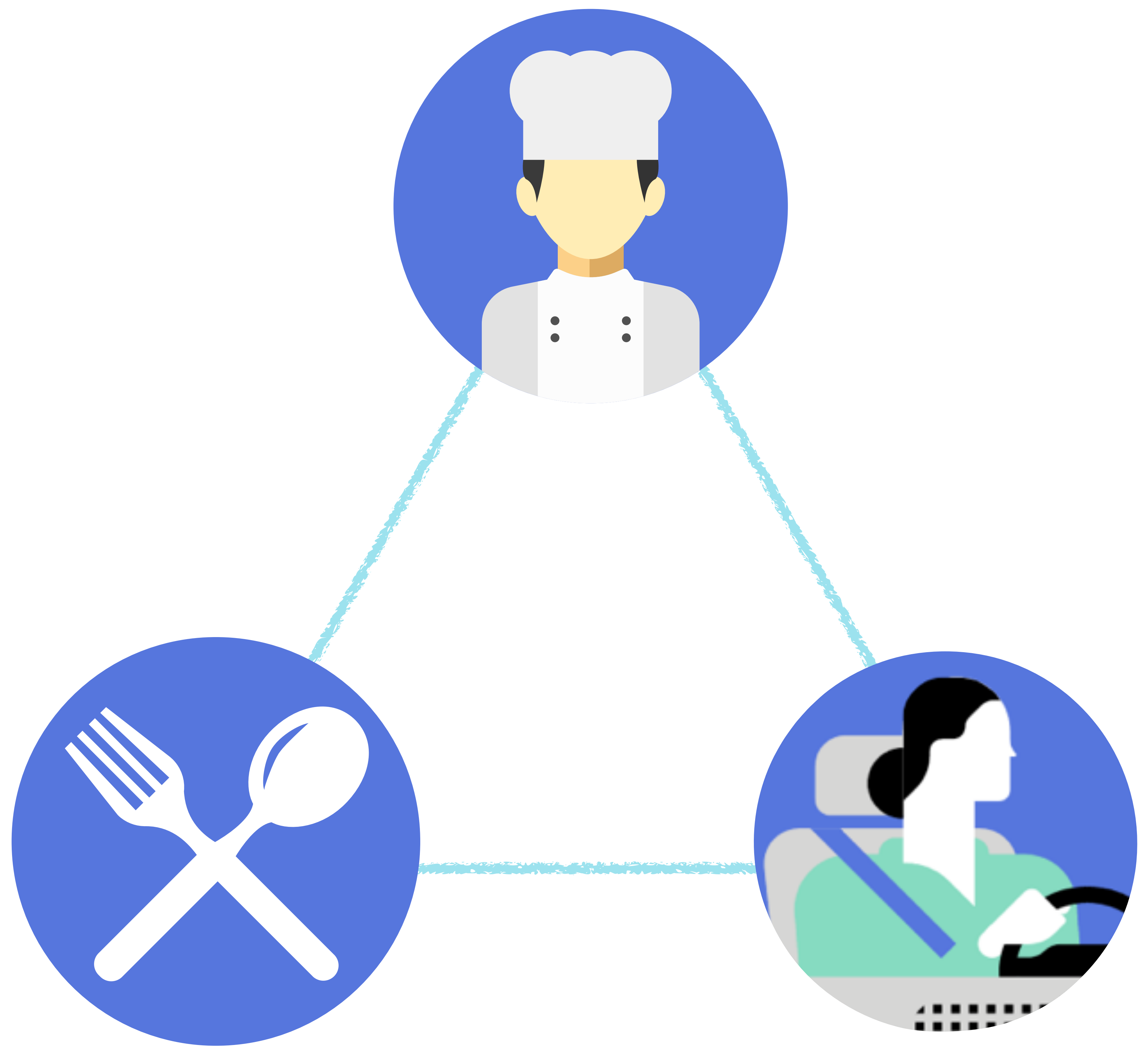
The case of UberEATS

- Three-way marketplace
- Real-time metrics
    - Estimated Time to Delivery (ETD)
    - Transactions
    - Demand forecasts



The case of UberEATS

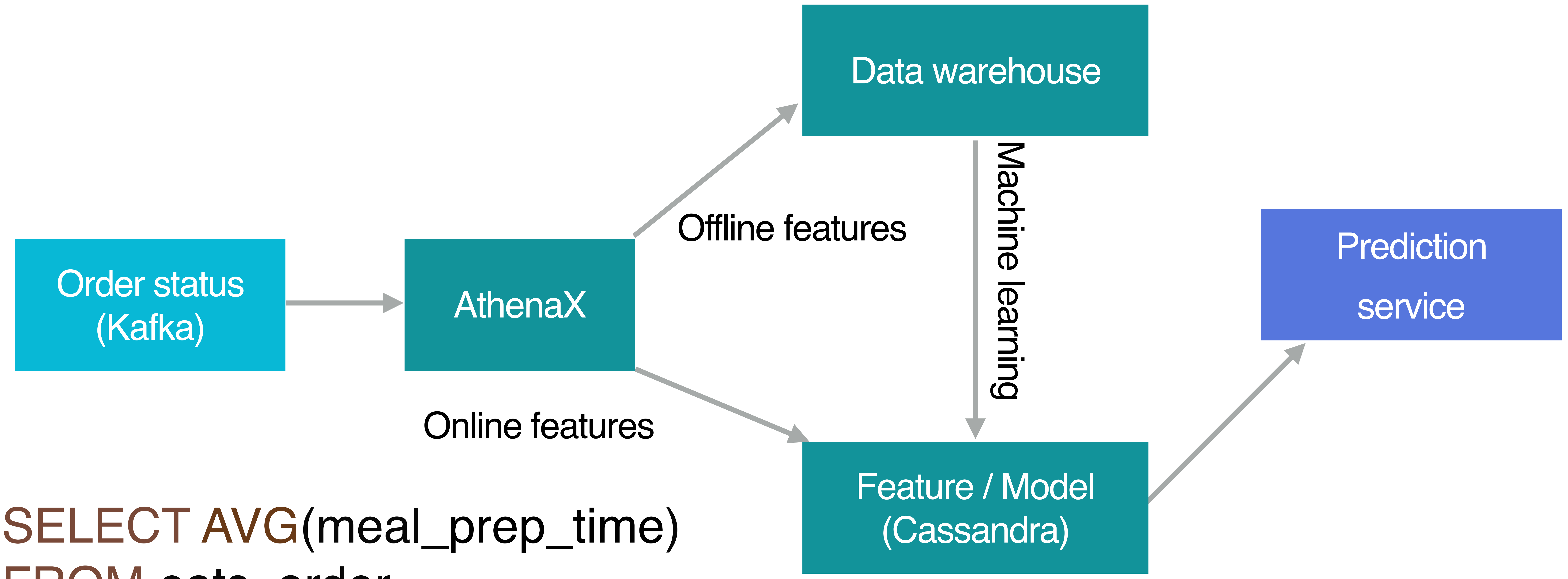
- Three-way marketplace
- Real-time metrics
  - Estimated Time to Delivery (ETD)
  - Transactions
  - Demand forecasts



Predicting the ETD

- Key metric: time to prepare a meal(t_{prep})
- Learn a function f: *(order status)* $\rightarrow t_{prep}$ *periodically*
- Predict the ETD for current orders using f
- AthenaX extracts features for both learnings and predictions

Architecture of the ETD service

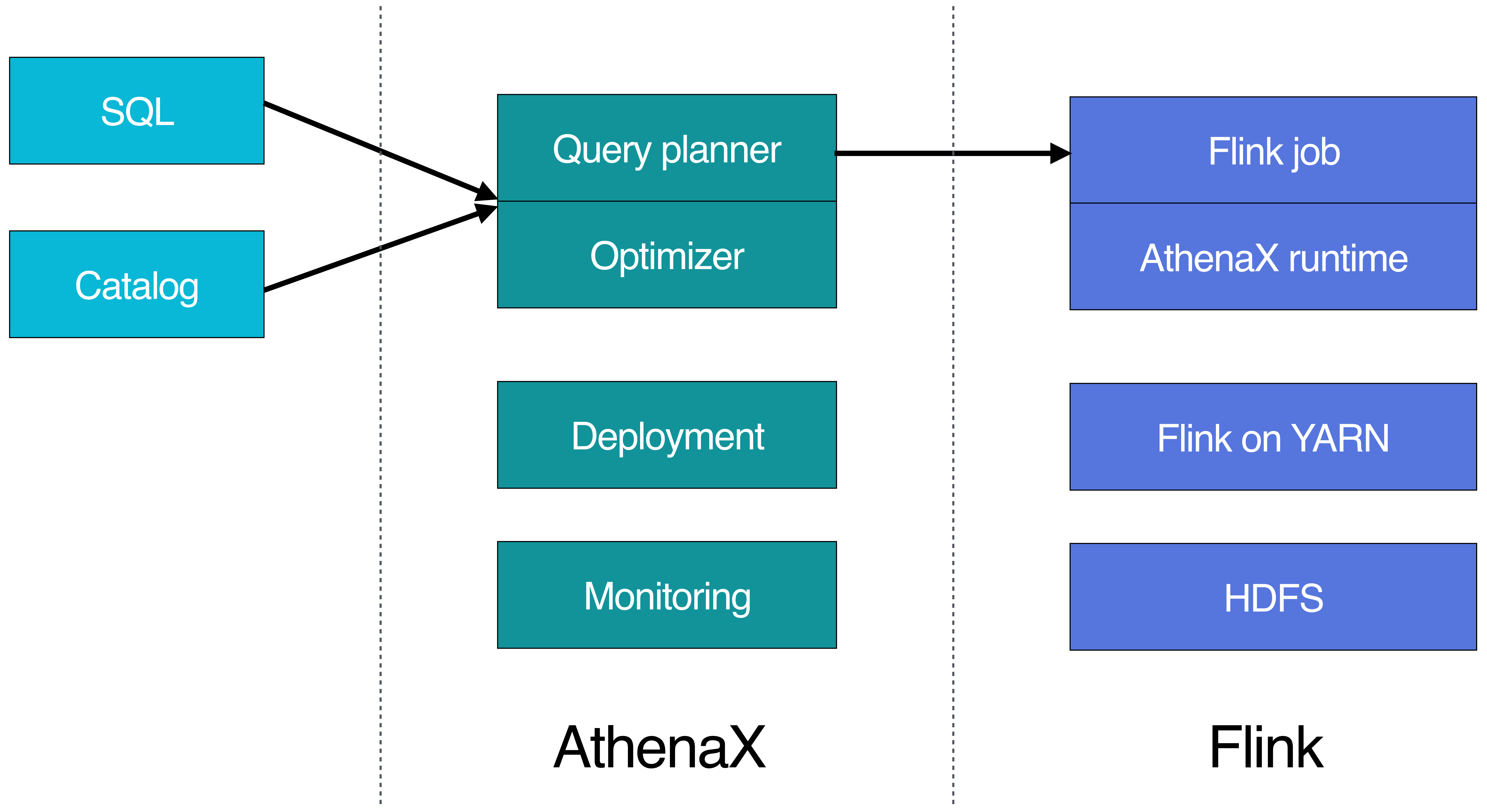


```
SELECT AVG(meal_prep_time)
FROM eats_order
GROUP BY meal_id,
HOP(proctime(),
  INTERVAL ‘1’ MINUTE,
```

Agenda

- Motivating example
- Case study: ETD in UberEATS
- Implementation
- Current status
- Conclusion

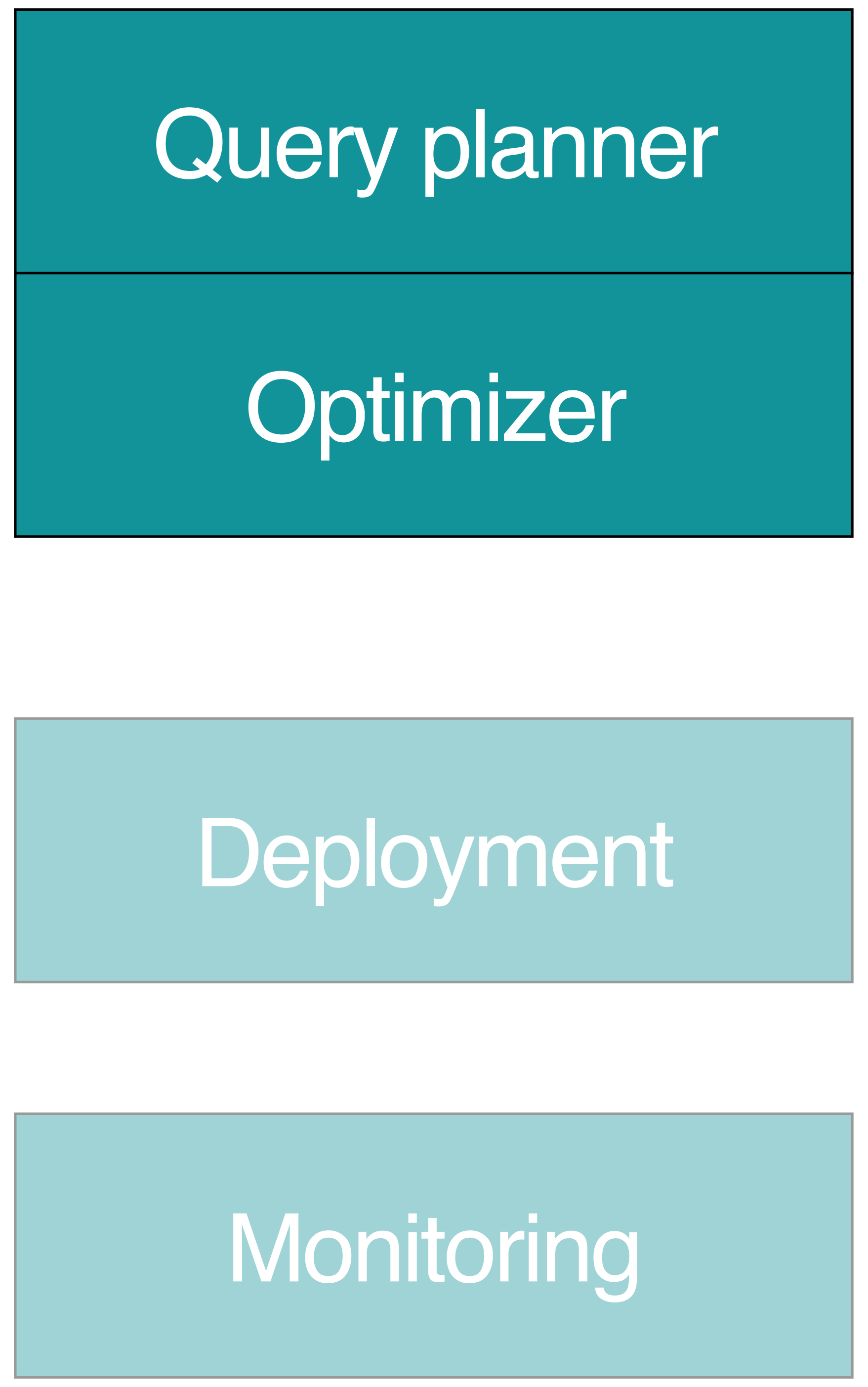
Architecture



Executing AthenaX applications

Compile SQLs to Flink applications

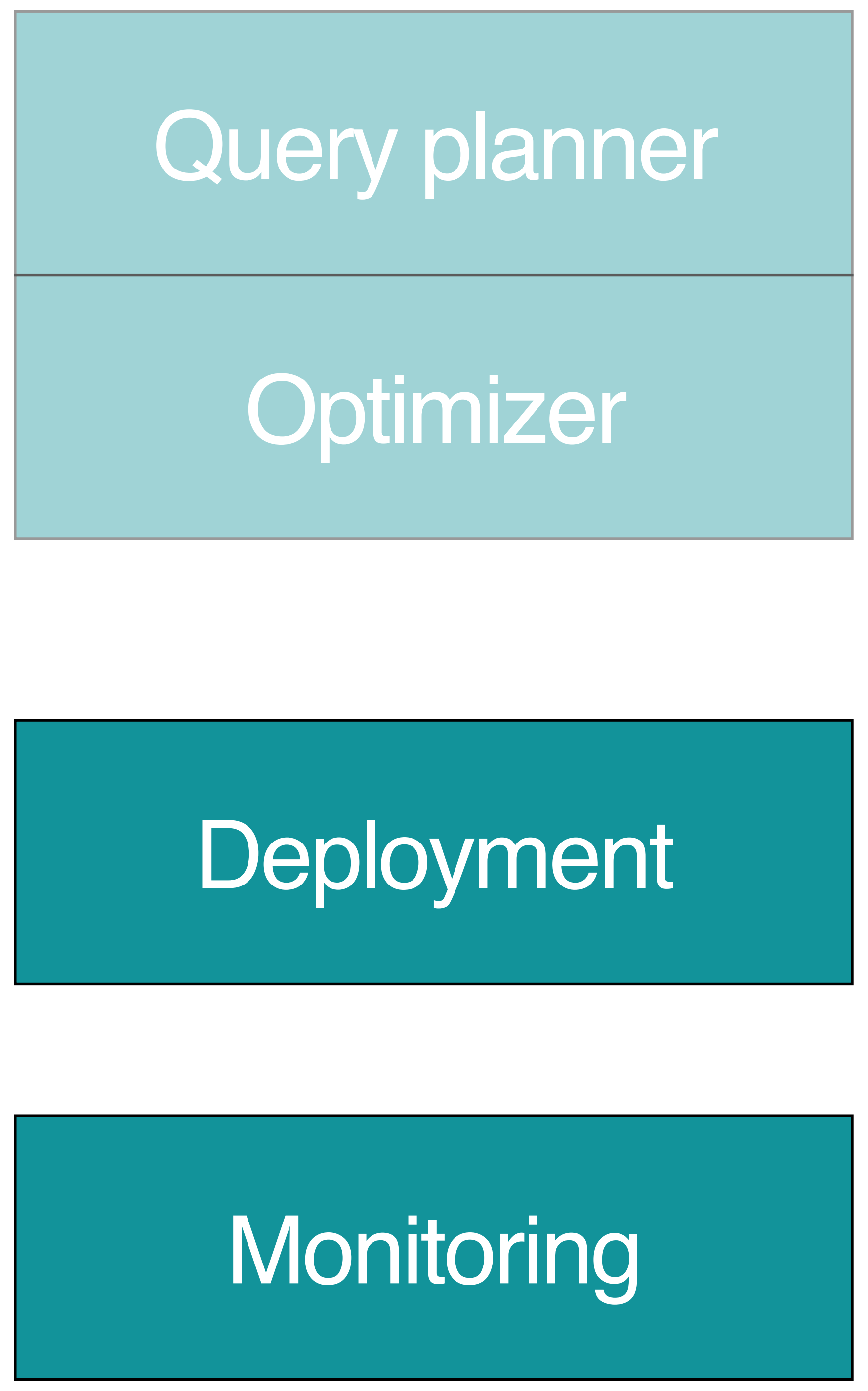
- Compilation + Code generation
  - Flink SQL APIs: SQL → Logical plans → Flink applications
  - Leverage the Volcano optimizer in Apache Calcite
- Challenges: exposing streaming semantics



AthenaX as a self-serving platform

Self-serving production support end-to-end

- Metadata / catalog management
- Job management
- Monitoring
- Resource management and elastic scaling
- Failure recovery



Agenda

- Motivating example
- Case study: ETD in UberEATS
- Implementation
- Current status
- Conclusion

Current status

- Pilot jobs in production
  - In the process of full-scale roll outs
- Based on Apache Flink 1.3-SNAPSHOT
  - Projection, filtering, group windows, UDF
  - Streaming joins not yet supported

Embrace the community

Contributions to the upstream

- Group window support for streaming SQL
  - CALCITE-1603, CALCITE-1615
  - FLINK-5624, FLINK-5710, FLINK-6011, FLINK-6012
- Stability fixes
  - FLINK-3679, FLINK-5631
- Table abstractions for Cassandra / JDBC (WIP)
- Available in the upcoming 1.3 release

Agenda

- Motivating example
- Case study: ETD in UberEATS
- Implementation
- Current status
- Conclusion

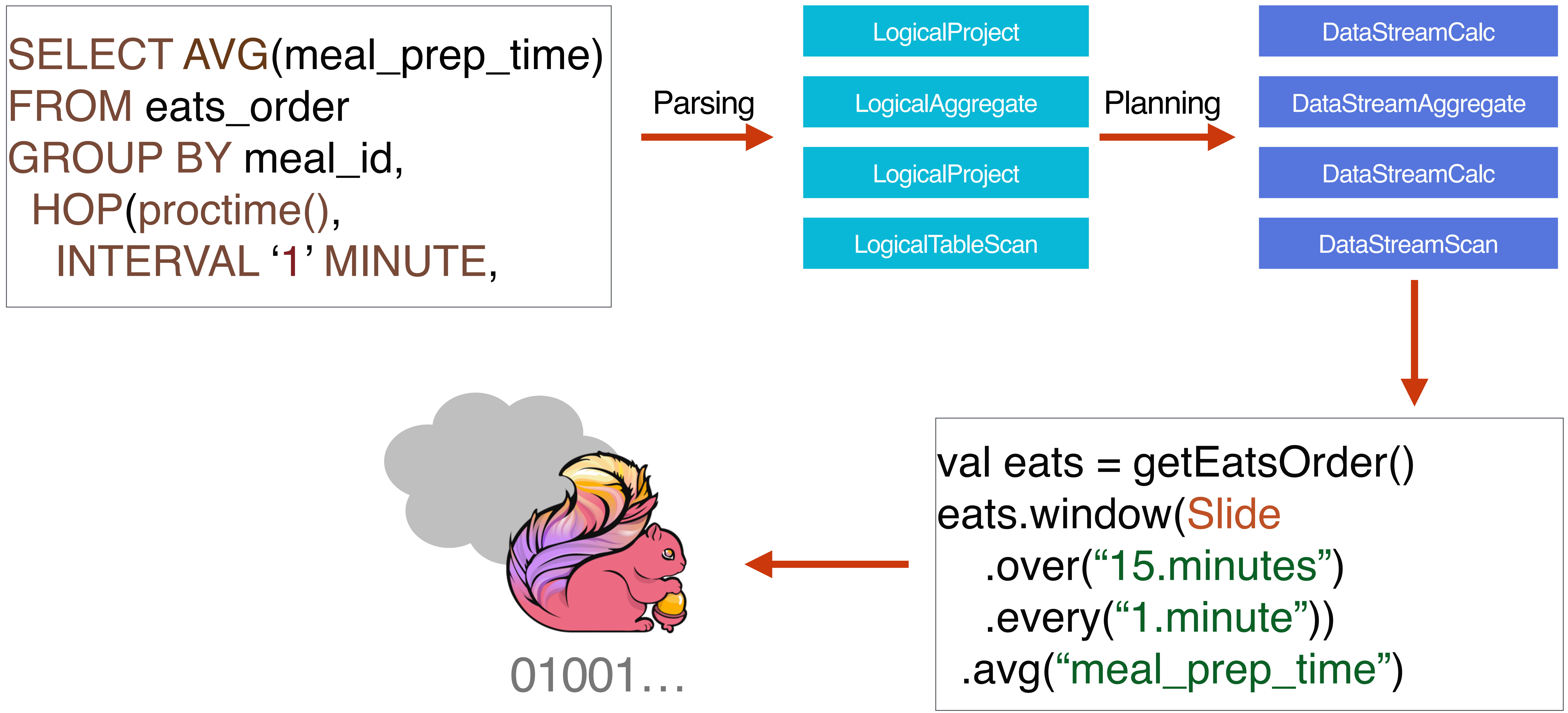
Conclusion

- AthenaX: write SQLs to build streaming applications
  - Treat table as a generic concept
  - Productivity: development → production in hours
- The AthenaX approach
  - SQL on streams as a platform
  - Self-serving production support end-to-end

Thank you

UBER

Compiling SQL

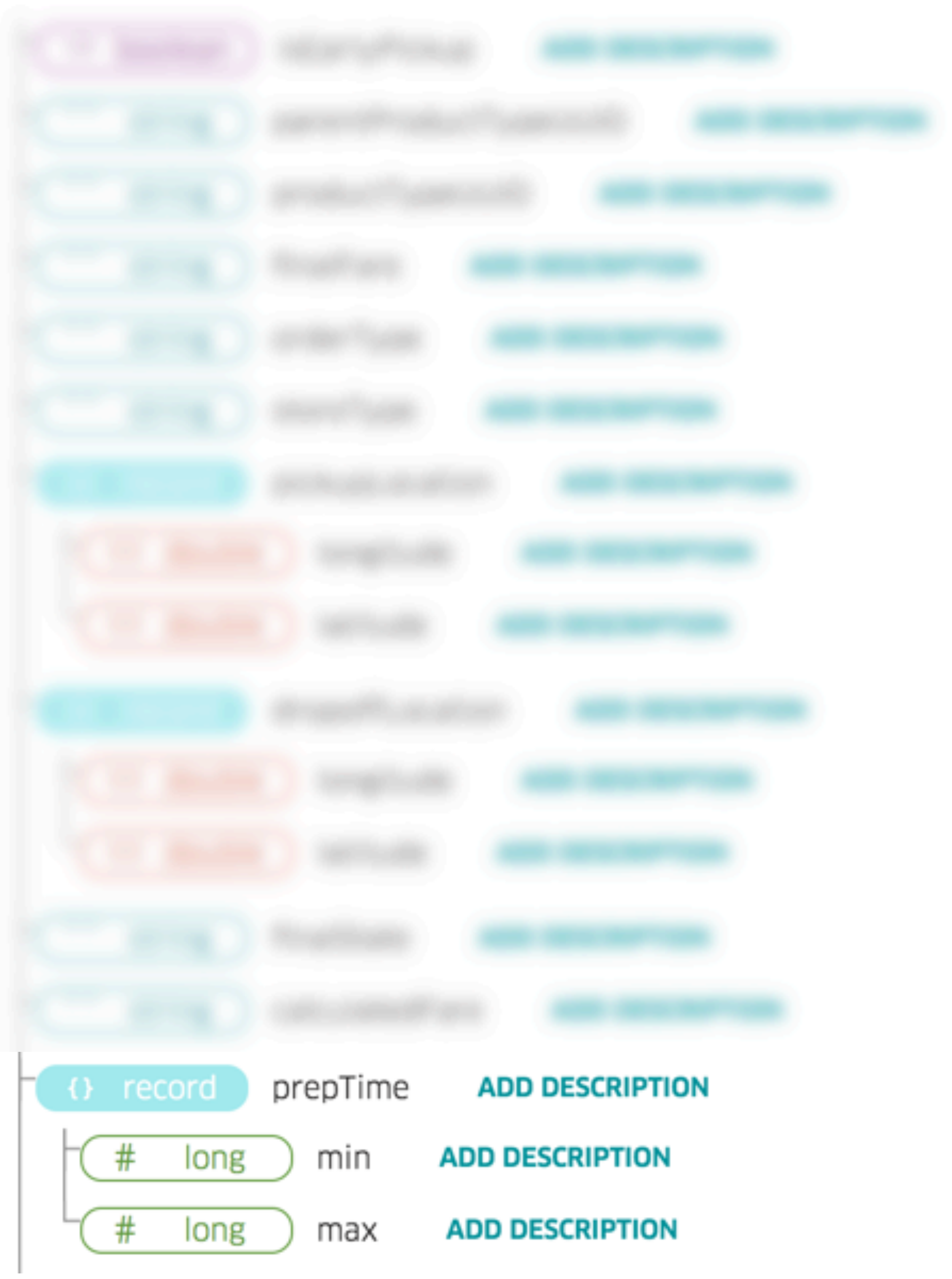


```
SELECT AVG(meal_prep_time)
FROM eats_order
GROUP BY meal_id,
 HOP(proctime(),
   INTERVAL '1' MINUTE,
```

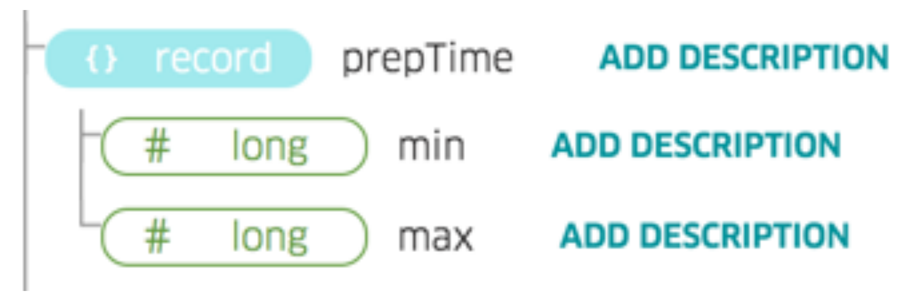
```
val eats = getEatsOrder()
eats.window(Slide
    .over("15.minutes")
    .every("1.minute"))
  .avg("meal_prep_time")
```

Lazy deserialization

Example of SQL optimization


{} record prepTime ADD DESCRIPTION
# long min ADD DESCRIPTION
# long max ADD DESCRIPTION

```
SELECT
AVG(meal_prep_time)
FROM eats_order
```


{} record prepTime ADD DESCRIPTION
# long min ADD DESCRIPTION
# long max ADD DESCRIPTION