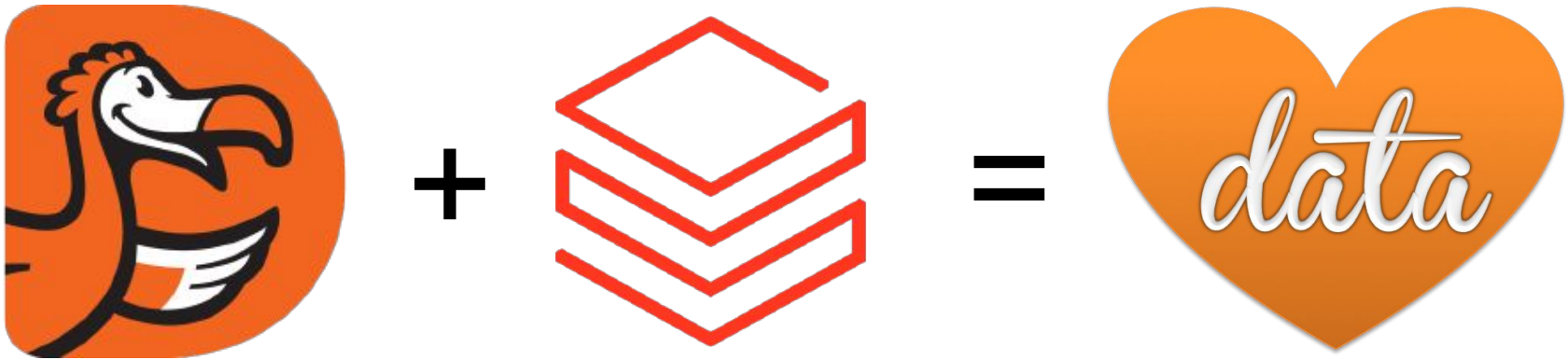
Разбор реального проекта: E2E пайплайн для прогнозирования закупок ингредиентов в пиццериях с помощью Spark Streaming



Спикеры

Михаил Кумачев

Data Engineer

mikhail.kumachev@gmail.com

@ceridan

Ксения Томак

Data Engineer

tomak.ksu@gmail.com

@ktomak

Дарья Буланова

Data Engineer

darya.bulanova@gmail.com

@daryabulanova

Иван Трусов

Very Simple Solutions Architect

ivan.trusov@databricks.com

@renardeinside

План на сегодня

Часть 1:

- Описание проекта и базовой инфраструктуры.
- Архитектура решения.
- Change Data Capture из MySQL в EventHubs, используя Kafka Connect и Debezium.

Часть 2:

- Data modeling с помощью DataVault 2.0. Переливка данных с помощью Spark Streaming.
- Наполнение витрин данных.
- Интеграция с ML-пайплайнами.
- CI/CD для пайплайнов данных.

Dodo Pizza сегодня

- 10 лет на рынке.
- 620 пиццерий.
- 13 стран + 2 на подходе.
- ~150 заказов в минуту.
- > 300 заказов в минуту в пиковые дни.
- Dodo IS

Какую проблему решаем

Прогнозирование закупок ингредиентов в пиццериях Dodo Pizza.

~2,5M событий в день.

Наша инфраструктура

- Все в облаке Azure.
- Источник: данные поступают из двух различных Azure Database for MySQL.
- Хранилище: Delta Lake + Azure Data Lake Storage (ADLS)
- Процессинг: Databricks, Spark, Spark Streaming

Почему Databricks

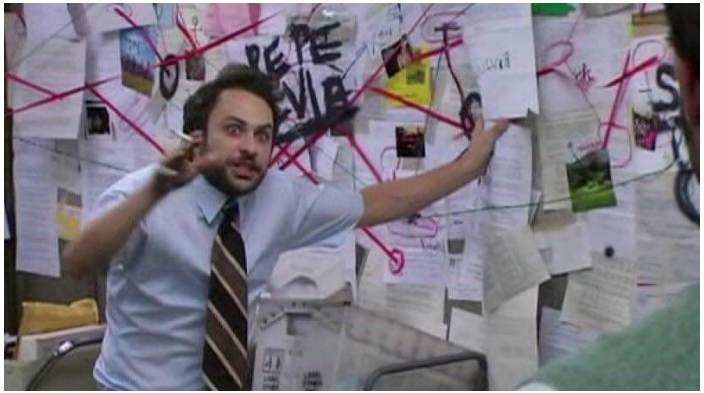
Как было раньше:

- Хранилище: Azure SQL Data Warehouse.
- Процессинг: Azure Data Factory (ADF).

Почему Databricks:

- Отказ от Azure Data Factory (невозможно поддерживать решения, отсутствие контроля).
- Отказ от Azure SQL Data Warehouse (очень дорого).
- Переходим на Spark для процессинга. Нужно managed-решение.
- Строим Data Lake. Технология Delta Lake – подходящее решение.
- Хорошая документация и поддержка со стороны Databricks.

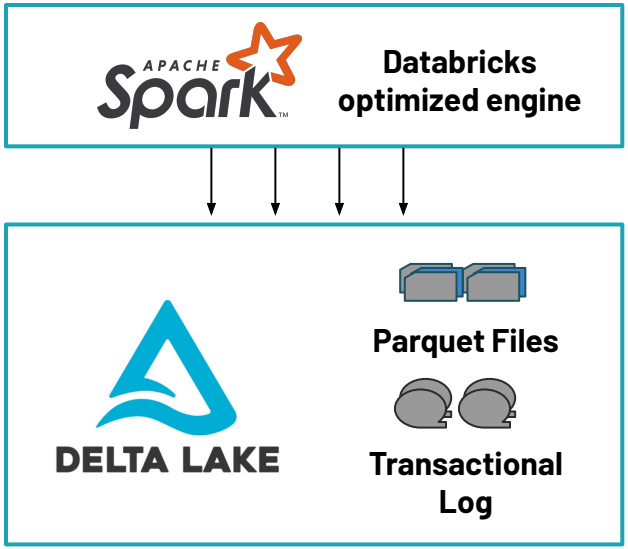
Архитектура



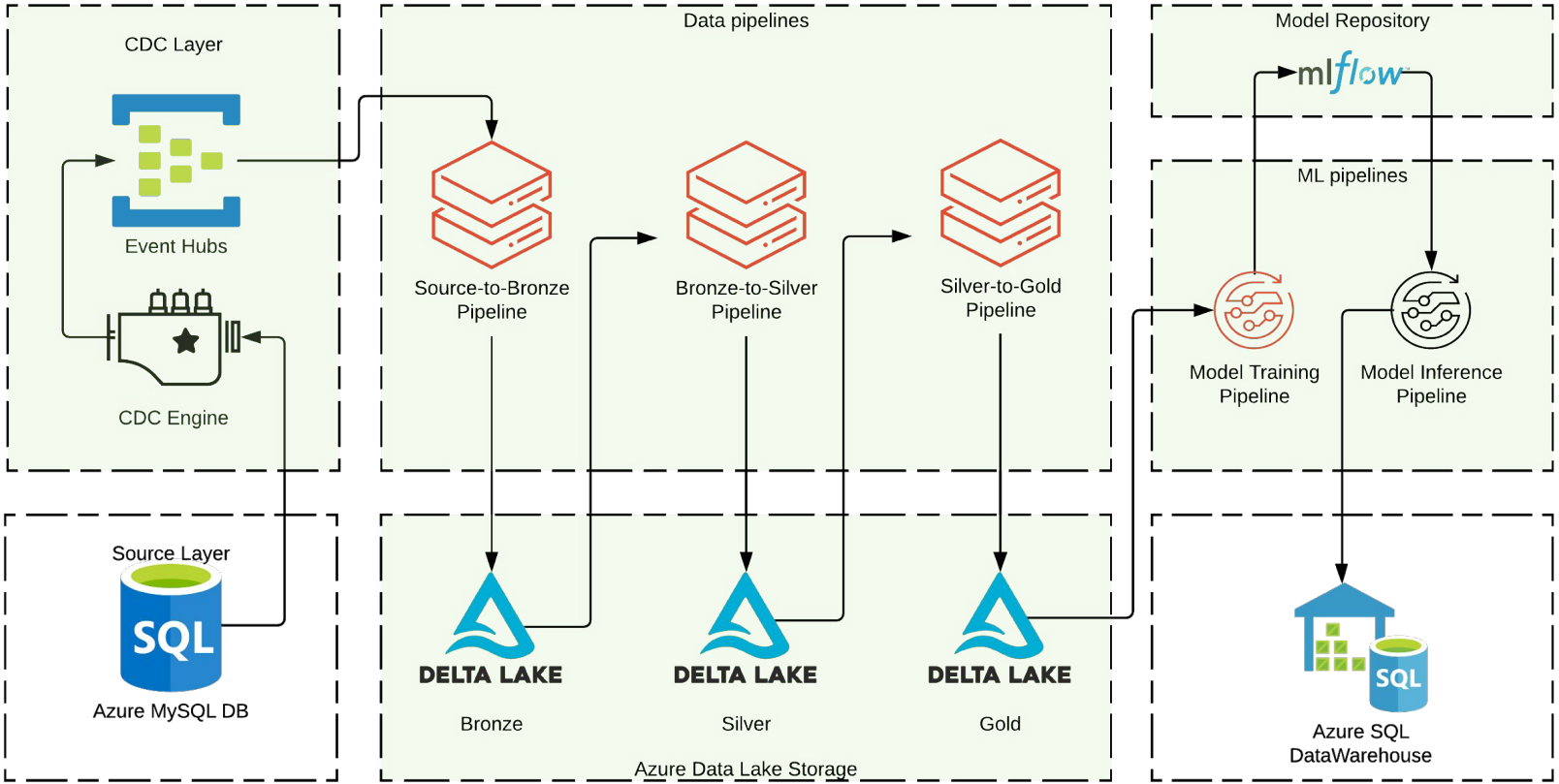
- Managed решение для Data и ML workloads
- Более 5000 клиентов
- Основана создателями OSS проектов

Delta Lake

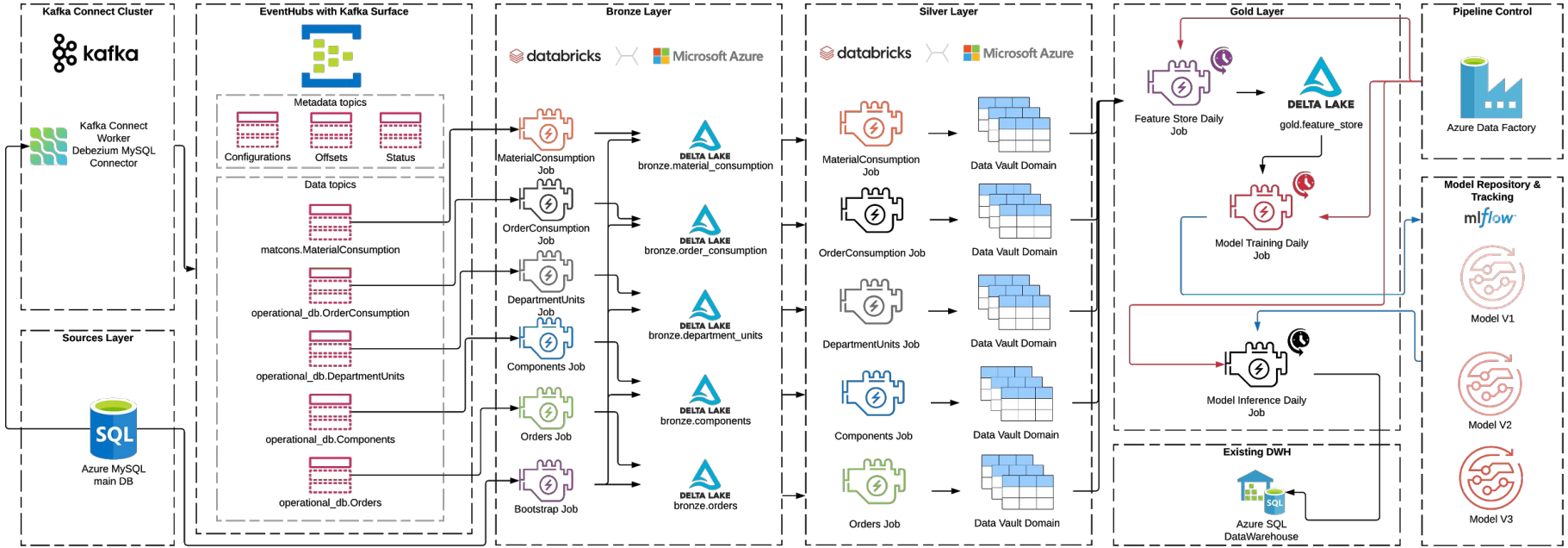
- Гибкий и управляемый подход к хранению данных в cloud storage
- Ключевой функционал:
  - Batch / Streaming APIs
  - Insert / Update / Delete support
  - Time travel / Snapshots
  - Z-ordering
  - File compaction
  - Delta Cache
  - Delta Engine



Верхнеуровневая архитектура



Детальная архитектура

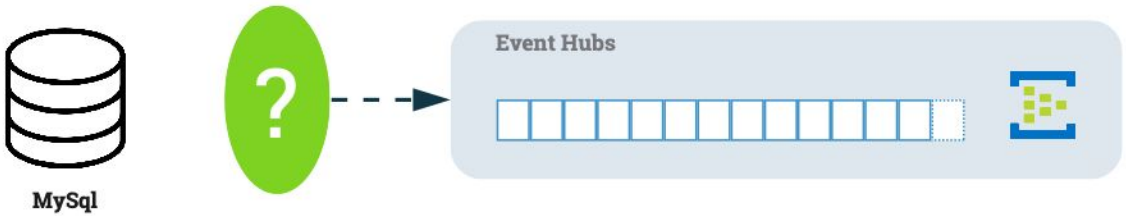


CDC from MySql to EventHubs

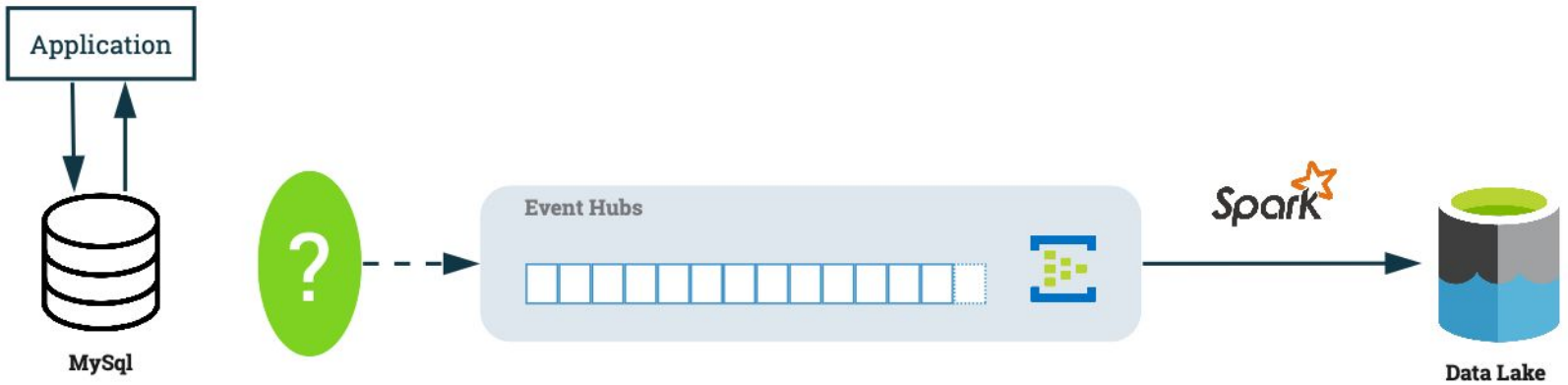
Using Kafka Connect and Debezium

Agenda

1. Что такое Change Data Capture и его идея
2. MySql - как устроена репликация данных (binlog, binlog_format)
3. Что такое Debezium
4. Чем занимается Kafka Connect
5. Как работает Kafka Connect с Debezium
6. Проблемы



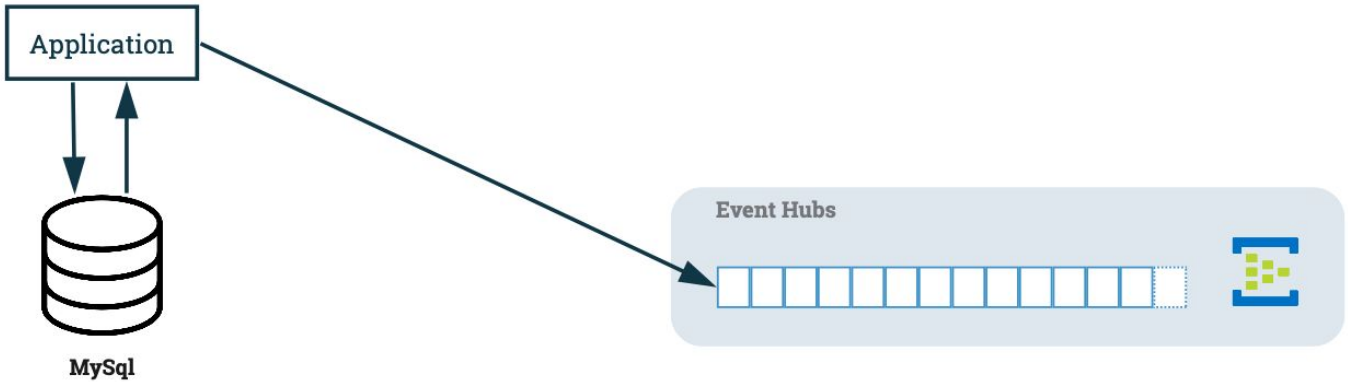
What is Change Data Capture (CDC)?



CDC implementation

Options:

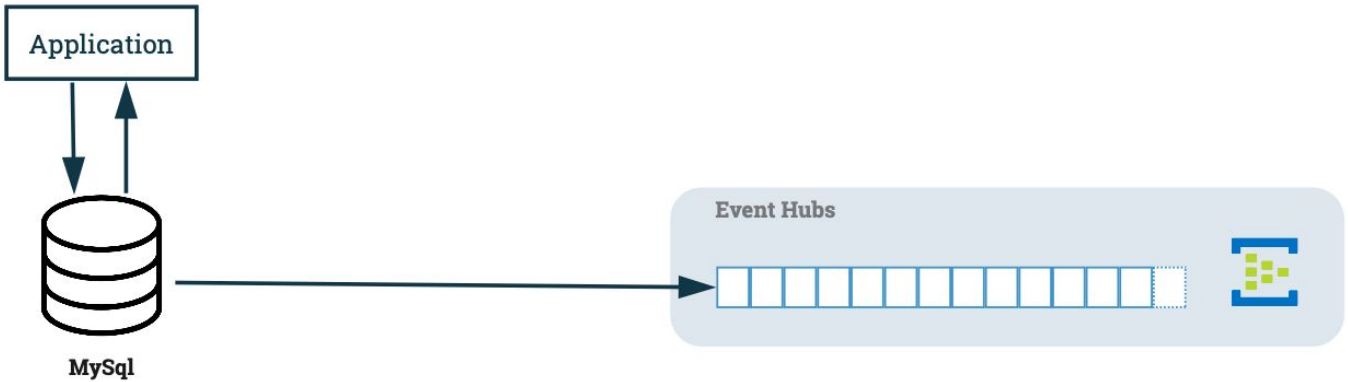
1. Application level = business events



CDC implementation

Options:

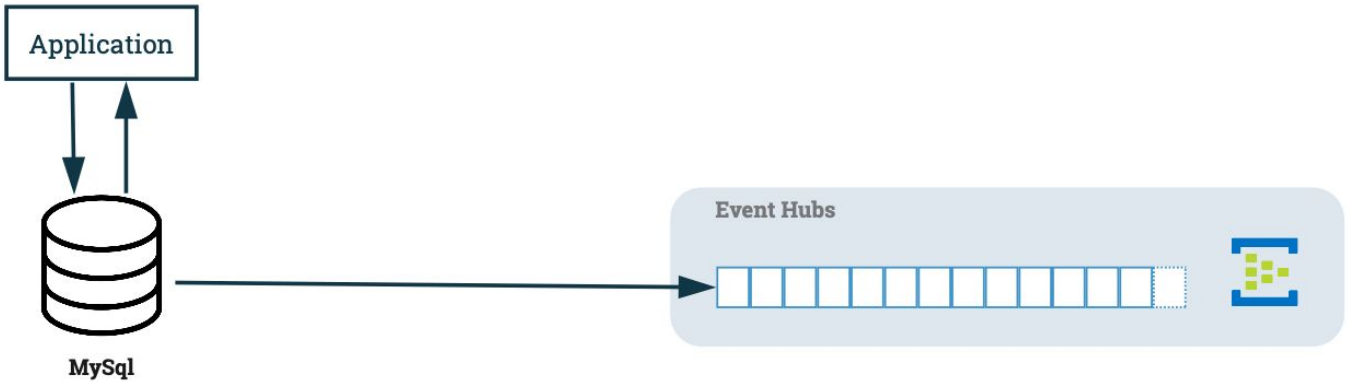
1. Application level = business events
2. Database level = trigger-based CDC



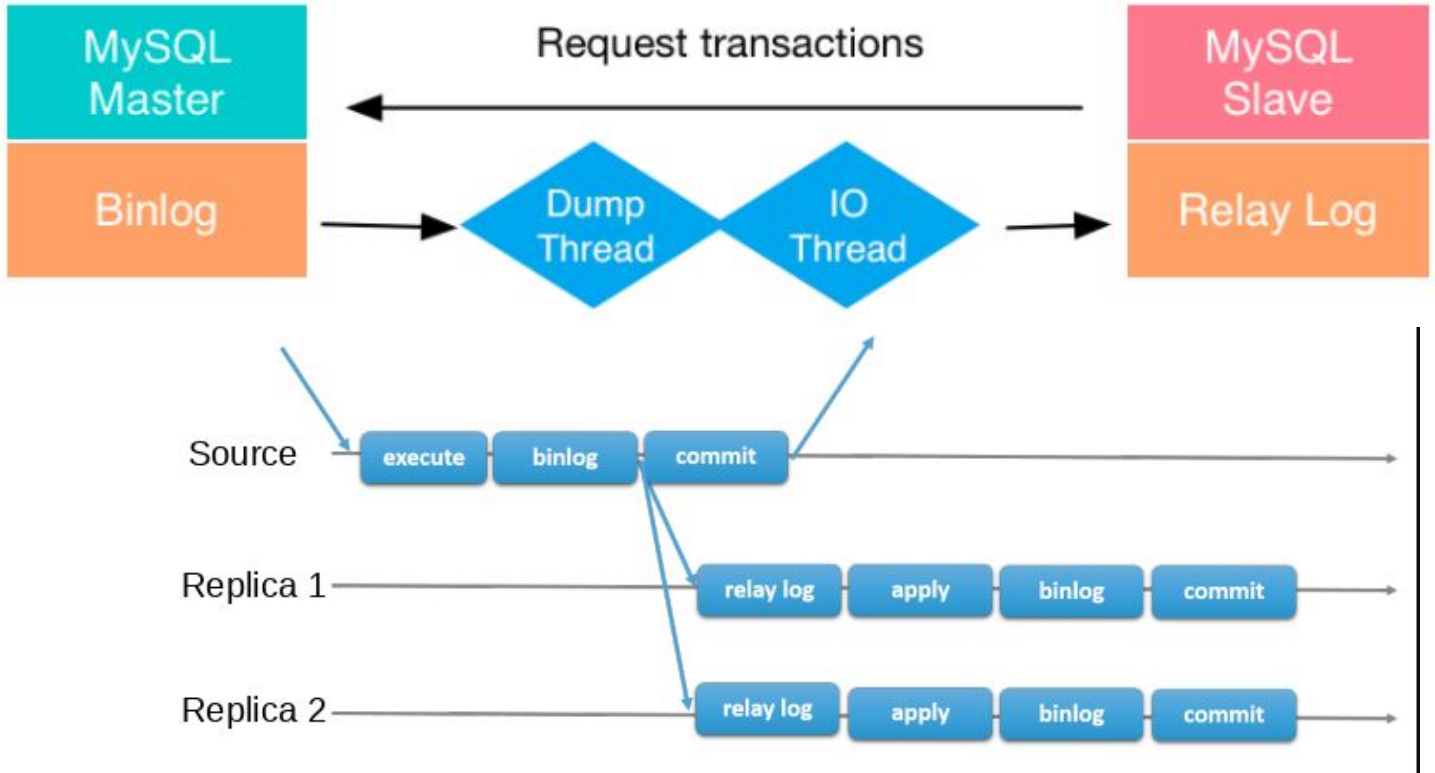
CDC implementation

Options:

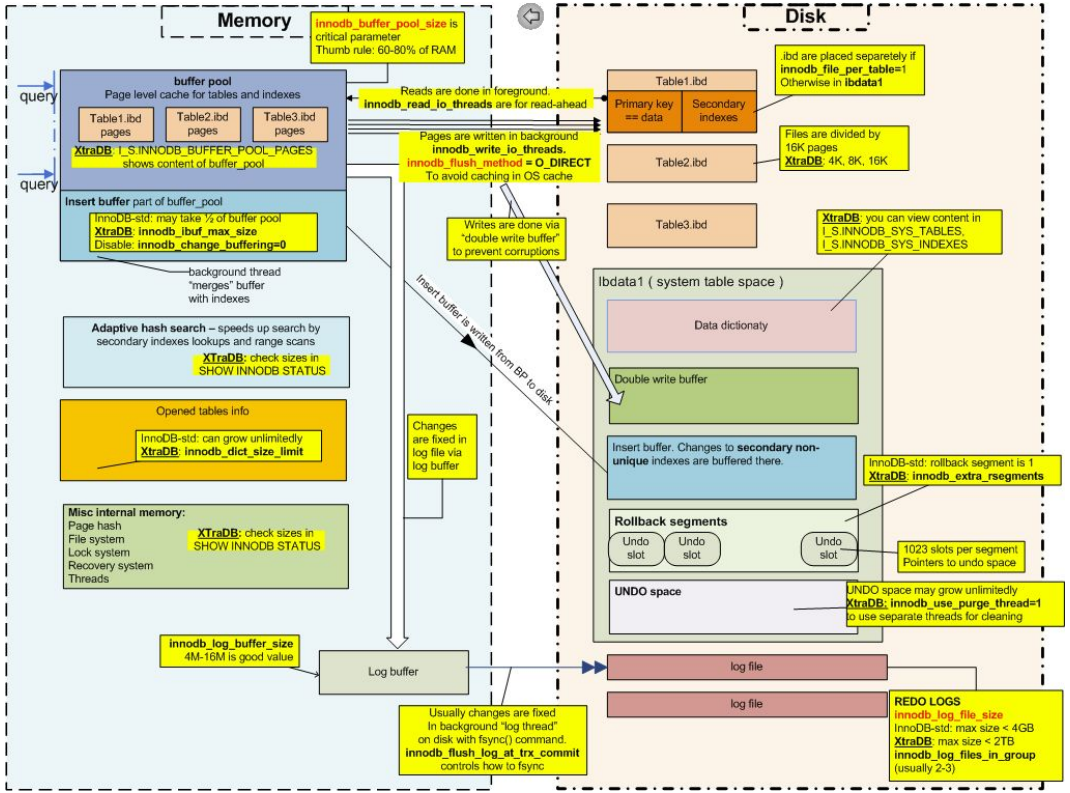
1. Application level = business events
2. Database level = trigger-based CDC
3. Database level = log-based CDC



MySql: replication overview



MySql: replication overview



MySql: binlog

Contains

- DML-statements (row modification)
- DDL-statements (schema modification)

Formats

- Statement-based
- Row-based
- Mixed-based

MySql: binlog formats

Statement-based logging

```
INSERT INTO customer (first_name, last_name, email) VALUES('IVA2', 'TITOV',
'iva2.titov@gmail.com')
/*!*/;
# at 1794
#200715 23:44:11 server id 1  end_log_pos 1825 CRC32 0x82a97489     Xid = 81
COMMIT/*!*/;
```

Row-based logging

```
BINLOG '
J2oPXxMBAAAAQAAAAFAFAAAAANsCAAAAAAEABHRlc3QACGN1c3RvbWVyAAM
PDw8GUABQAFAABwID
/P8AZ7Drug==
J2oPXx4BAAAAQgAAAJIFAAAAANsCAAAAAAEAAgAD/wADSVZBBVRJVE9WE2l2Y
S50aXRvdkBnbWFp
bC5jb23mpU0F
'/*!*/;
### INSERT INTO `test`.`customer`
### SET
###   @1='IVA'
###   @2='TITOV'
###   @3='iva.titov@gmail.com'
```

MySql: row-based logging

Basic:

Event	Before image	After image
create	None	Row to insert
delete	Row to delete	None
update	Column values before update	Column values after update

Requirements:

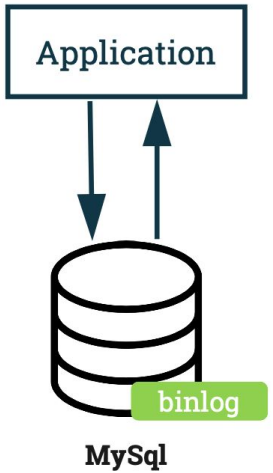
- GTID **(optional)** /primary key


```
{
 "schema": { ... },
 "payload": {
  "before": {
   "id": 1004,
   "first_name": "Anne Marie",
   "last_name": "Kretchmar",
   "email": "annek@noanswer.org"
  },
  "after": null,
  "source": {
   "version": "1.2.0.Final",
   "connector": "mysql",
   "name": "mysql-server-1",
   "ts_ms": 1465581,
   "snapshot": false,
   "db": "inventory",
   "table": "customers",
   "server_id": 223344,
   "gtid": null,
   "file": "mysql-bin.000003",
   "pos": 805,
   "row": 0,
   "thread": 7,
   "query": "DELETE FROM customers WHERE id=1004"
  },
  "op": "d",
  "ts_ms": 1465581902461
```

MySql

Настройка

- log_bin = ON
- binlog_format = row
- binlog_row_image = full



Debezium

Change Data Capture Platform

- **CDC for multiple databases**
  - Based on transaction logs
  - **Snapshotting**, Filtering etc..
- Via **Apache Kafka Connect** or embedded

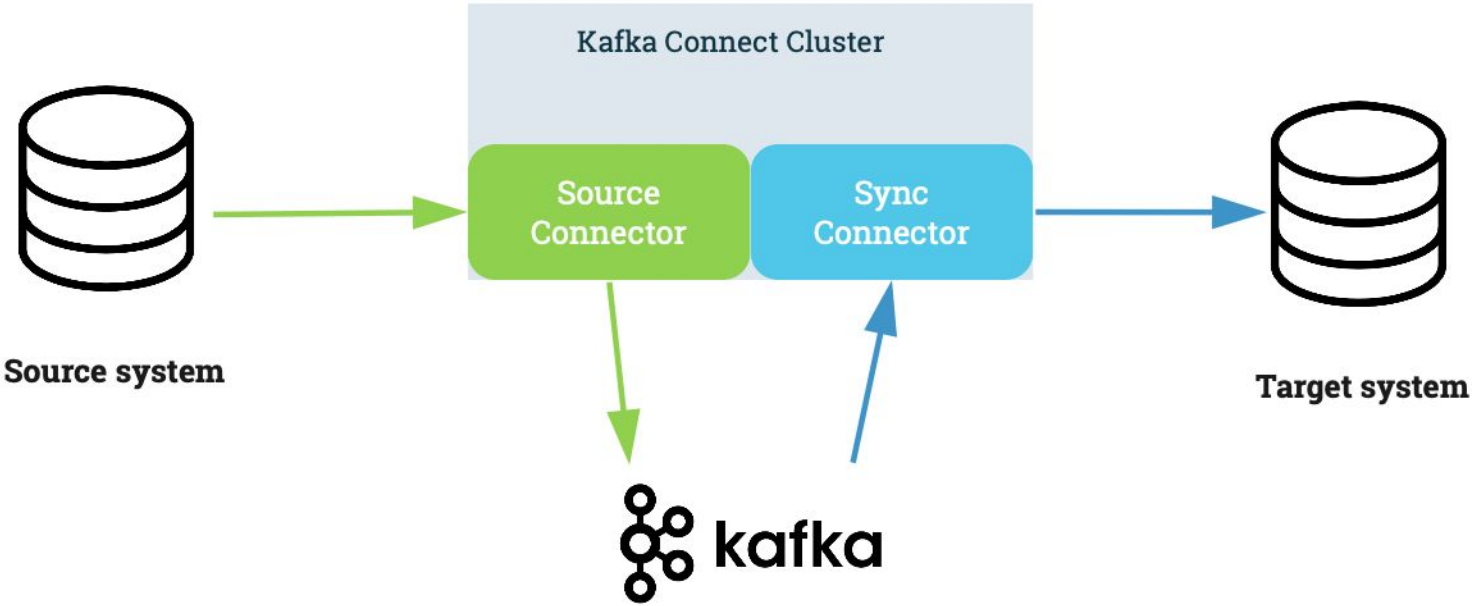
Features

- all data changes are captured
- very low delay
- no changes to your data model
- can capture deletes
- Fault tolerant

Snapshotting

- Global Read Lock
- Repeatable Read

Kafka Connect

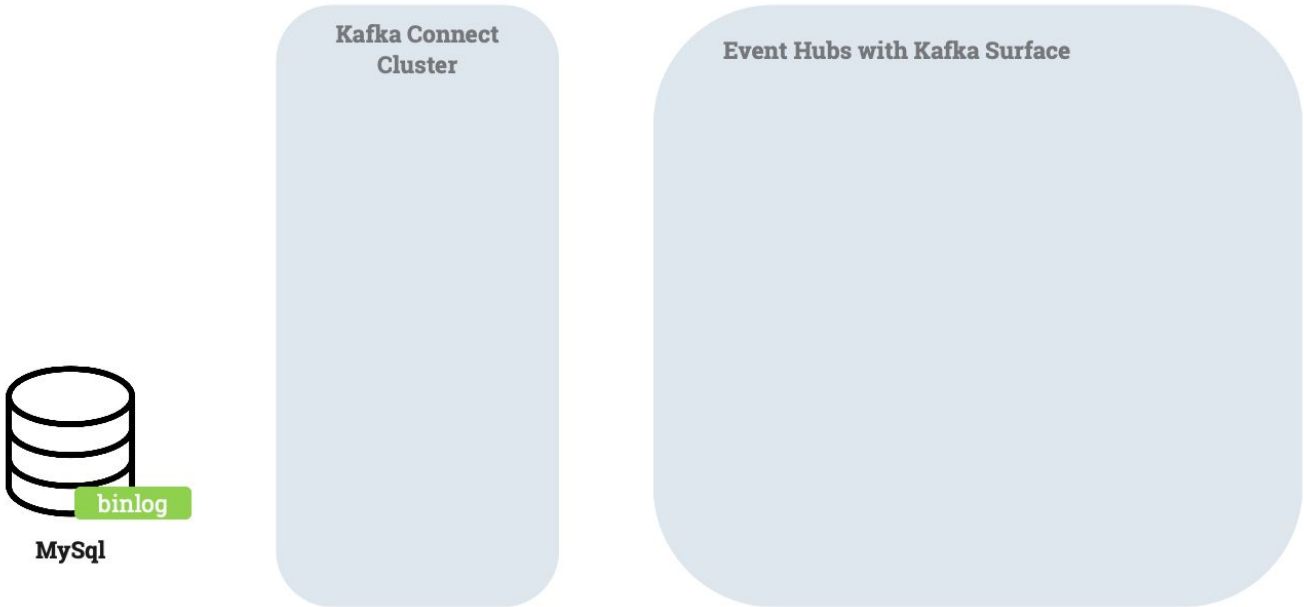


Using Kafka Connect + Debezium

- Kafka Cluster (EventHubs)

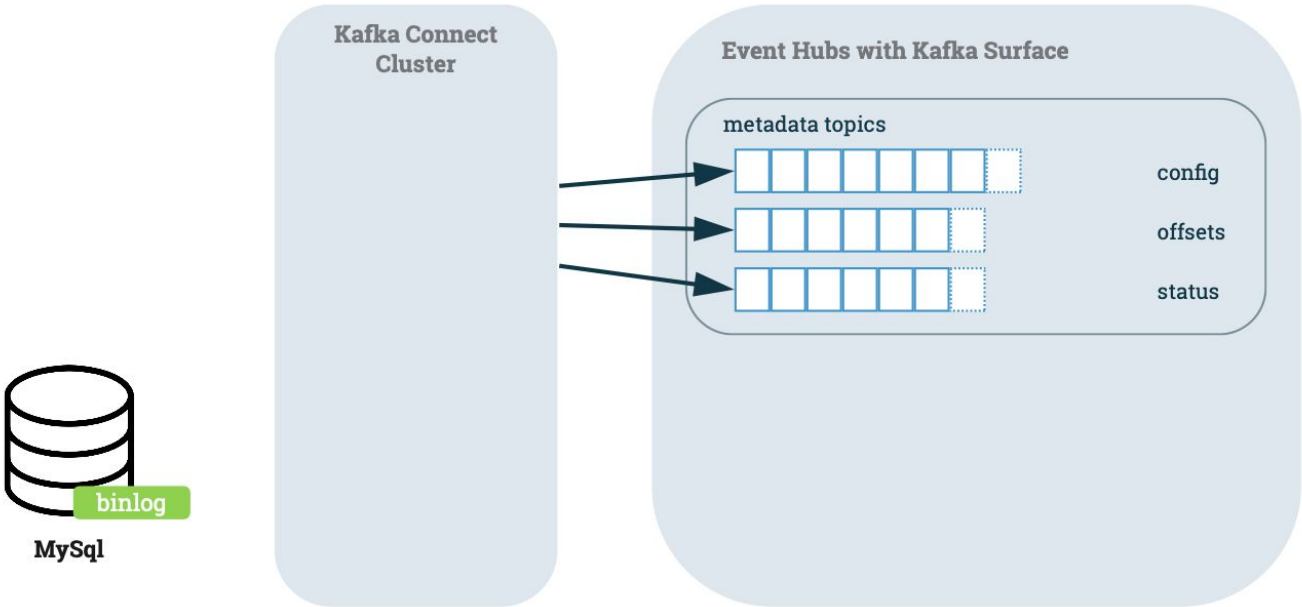
Using Kafka Connect + Debezium

- Kafka Cluster (EventHubs)
- Enable binlog MySql



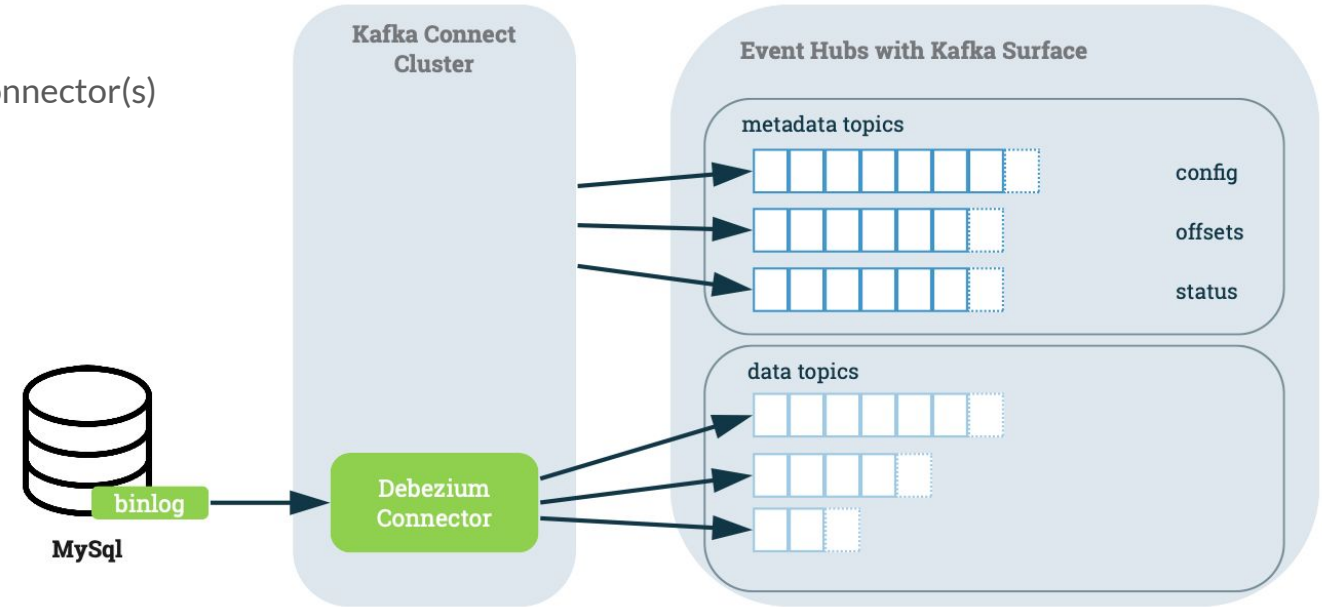
Using Kafka Connect + Debezium

- Kafka Cluster (EventHubs)
- Enable binlog MySql
- Start Kafka Connect



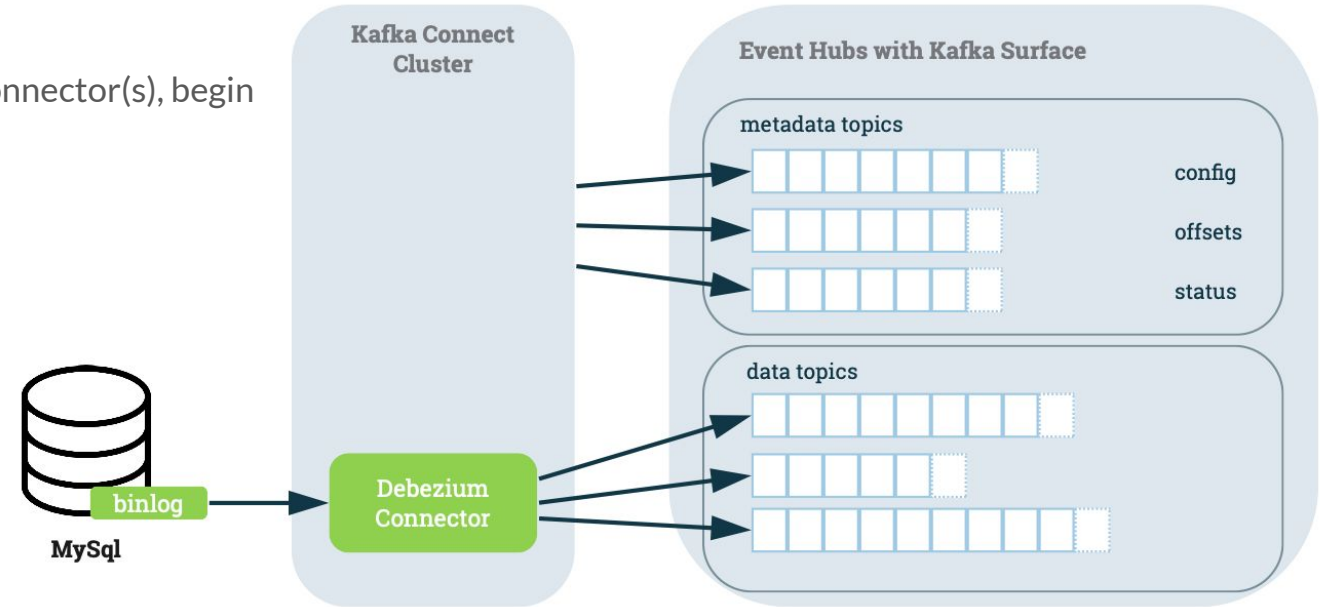
Using Kafka Connect + Debezium

- Kafka Cluster (EventHubs)
- Enable binlog MySql
- Start Kafka Connect
- Deploy Debezium connector(s)



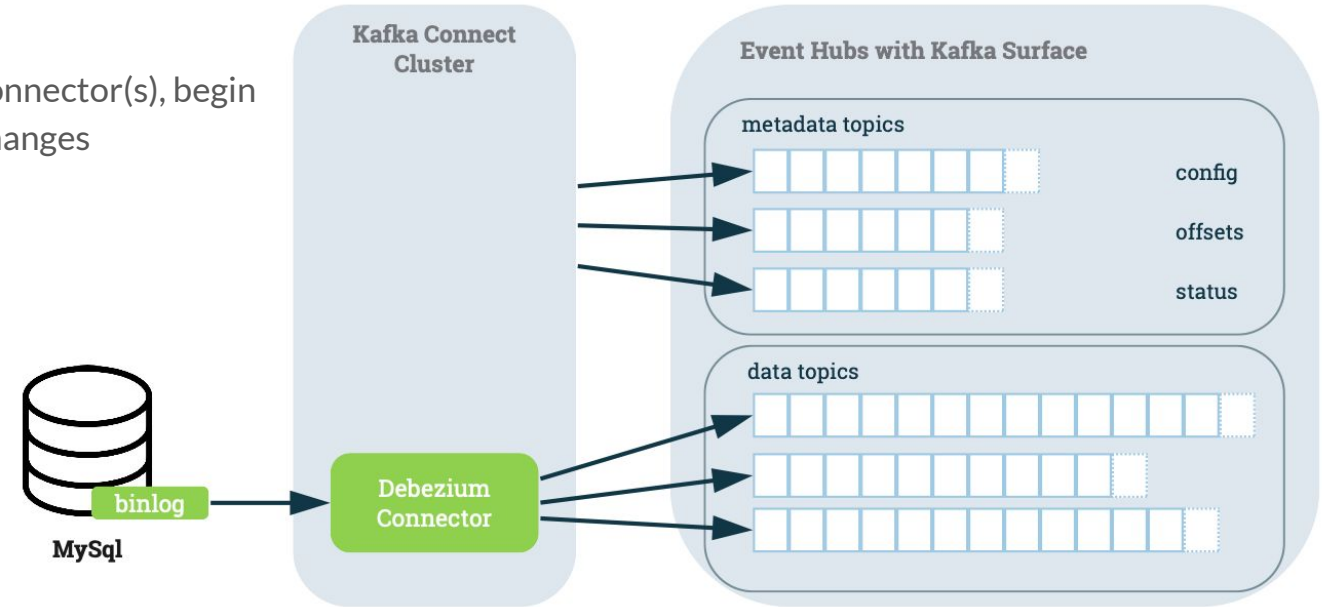
Using Kafka Connect + Debezium

- Kafka Cluster (EventHubs)
- Enable binlog MySql
- Start Kafka Connect
- Deploy Debezium connector(s), begin snapshot



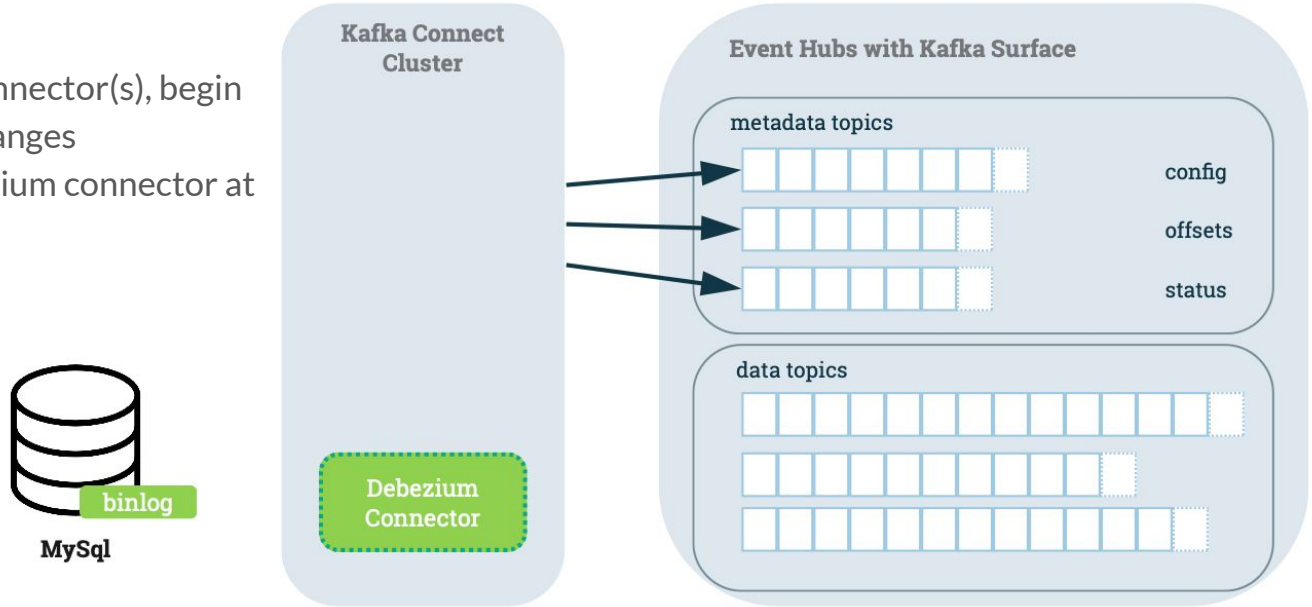
Using Kafka Connect + Debezium

- Kafka Cluster (EventHubs)
- Enable binlog MySql
- Start Kafka Connect
- Deploy Debezium connector(s), begin snapshot, capture changes



Using Kafka Connect + Debezium

- Kafka Cluster (EventHubs)
- Enable binlog MySql
- Start Kafka Connect
- Deploy Debezium connector(s), begin snapshot, capture changes
- Stop, redeploy Debezium connector at any time



Problems

Problem 1: MySql connector и Azure MySql

- Azure MySQL при подключении к binary log имеет нестандартную аутентификацию
- Библиотека **mysql-binlog-connector-java** < 0.20.1 не работает с Azure MySQL

Solution

Заменяем **mysql-bin-connector-java.jar** файл в Debezium Docker Image на нужную версию (>=0.20.1)

Problem 2: Kafka History Topic и EventHubs

```
org.apache.kafka.connect.errors.ConnectException: Creation of database history topic failed, please create the topic manually
  at io.debezium.relational.history.KafkaDatabaseHistory.initializeStorage(KafkaDatabaseHistory.java:365)
  at io.debezium.relational.HistorizedRelationalDatabaseSchema.initializeStorage(HistorizedRelationalDatabaseSchema.java:63)
  at io.debezium.connector.sqlserver.SqlServerConnectorTask.start(SqlServerConnectorTask.java:82)
  at io.debezium.connector.common.BaseSourceTask.start(BaseSourceTask.java:77)
```

https://github.com/Azure/azure-event-hubs-for-kafka/issues/61

Solution

```
database.history: "io.debezium.relational.history.FileDatabaseHistory",
database.history.file.filename: "/kafka/history/matcons.dat",
```

Problem 3: Flush Timeout

```
2020-06-11 10:25:35,661 ERROR  ||  WorkerSourceTask{id=matcons-connector-master-0} Failed to flush, timed out while waiting for producer
to flush outstanding 7 messages   [org.apache.kafka.connect.runtime.WorkerSourceTask]
```

https://stackoverflow.com/questions/58010247/azure-eventhub-kafka-org-apache-kafka-common-errors-timeoutexception-for-some-of/58385324#58385324

Solution

```
"producer.connections.max.idle.ms": "60000",
"producer.metadata.max.age.ms": "60000"
```

Problem 4: Snapshot loading (1/2)

- Debezium берет Global Read Lock

Для таблиц >> 1bln. строк процесс очень медленный

Solution

- Для большинства таблиц достаточно было запустить Spark jdbc с партицированием по index-колонке

```
df = (
  spark.read.format(“jdbc”)
  .option("dbtable", reader_query)
  .option("partitionColumn", "Id")
  .option("lowerBound", min_id)
  .option("upperBound", max_id)
  .option("numPartitions", 1000)
  .load()
)
```

Fetch size	Total Time (seconds)	Total Rows	Rows Per Second	Parallelism	Num Partitions
1000	330.0	9719786	26999	1	1
100000	120.6	9719786	80595	1	1
100000	62.4	9719786	155765	4	100

Problem 4: Snapshot loading (2/2)

- Для некоторых таблиц и этот подход очень медленный

Однако, в целевой таблице были определенны партиции на стороне DB

Solution

- Достаем список партиций и кануть по ним из information_schema.tables
- Группируем их для более равномерного распределения данных
- По каждой партиции запускаем параллельную загрузку

```
SELECT *
FROM db.table PARTITION(P2,P3,P4)
WHERE id_col >= x1 AND id_col <= X1
```

Результат: **3.4 часа вместо 72**

Azure Event Hubs

Features that are not yet supported

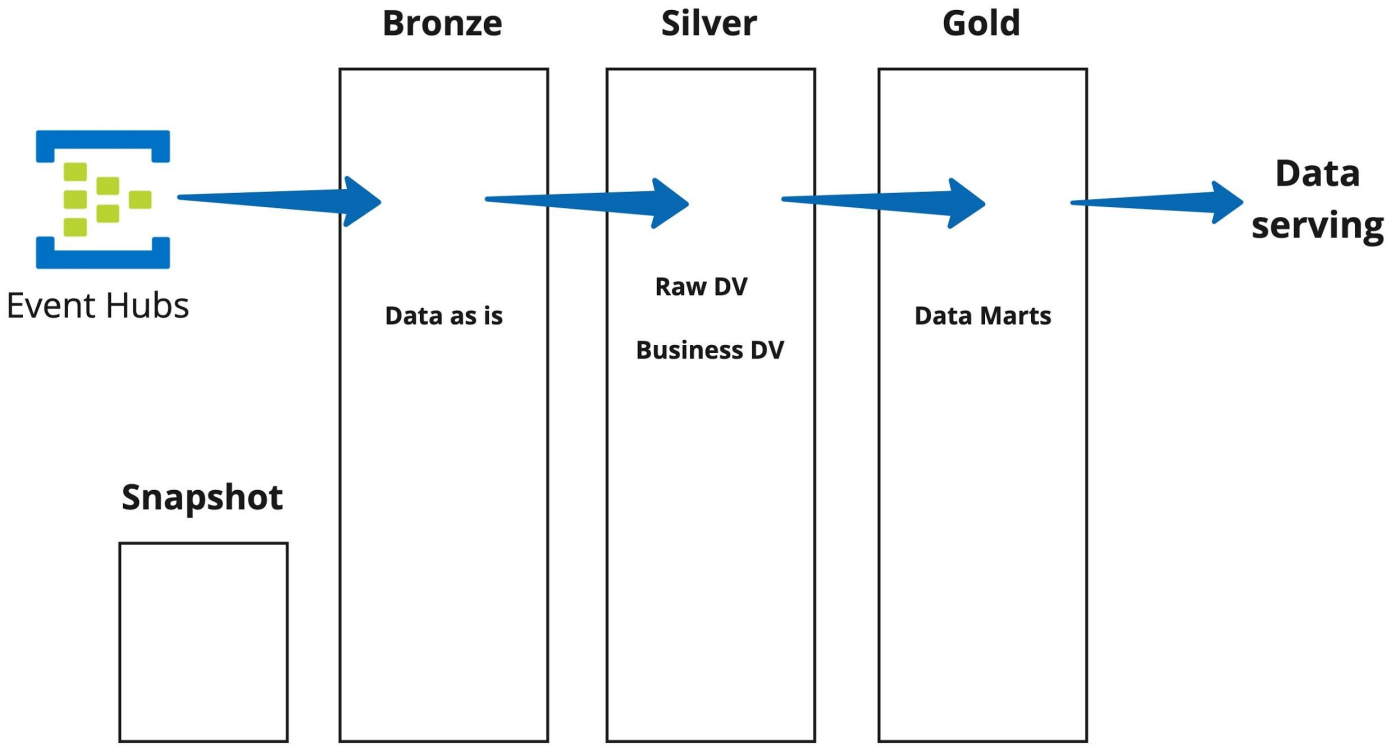
Idempotent producer
Transaction
Compression
Size-based retention
Log compaction
Adding partitions to an existing topic
HTTP Kafka API support
Kafka Streams

Problems

- 10 topic limitation
- Kafka Admin API not fully compatibility
- Max 7 days retention limitation
- 1MB per message

Q&A по первой части

Data modeling
NRT processing



- responsibility
- manage data access
- different latency

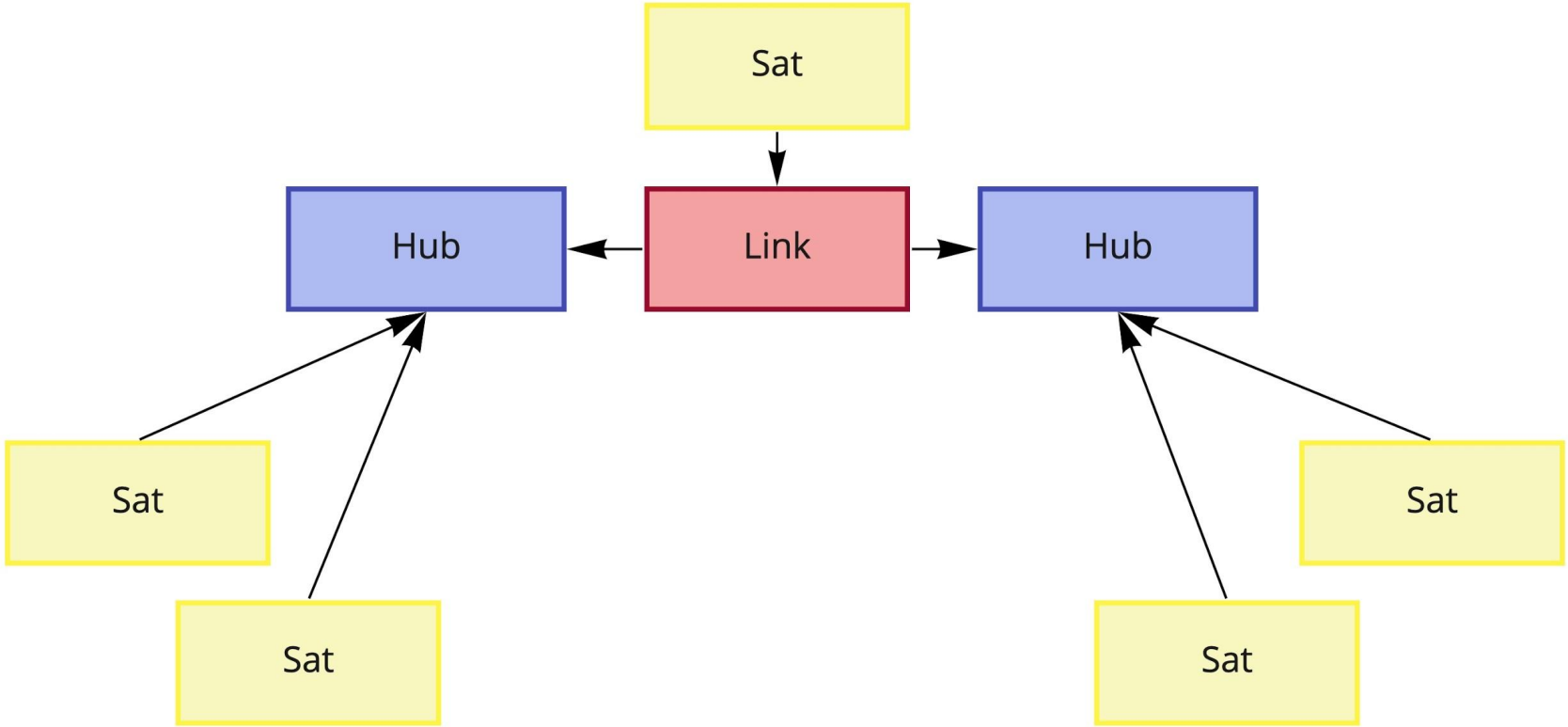
Why not 1?
Why not 8?

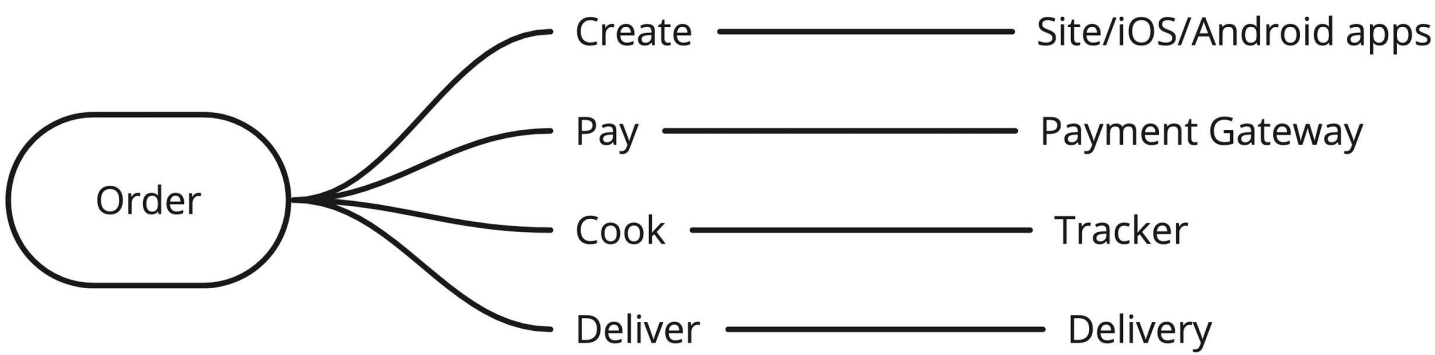
Bronze layer

Event Hubs


```
{
  "schema": {
    "type": "struct",
    "fields": [ ...
    ],
    "optional": false,
    "name": "mysql-server-1.inventory.customers.Envelope"
  },
  "payload": {
    "op": "c",
    "ts_ms": 1465491411815,
    "before": null,
    "after": {
      "id": 1004,
      "first_name": "Anne",
      "last_name": "Kretchmar",
      "email": "annek@noanswer.org"
    },
    "source": {
      "version": "1.2.0.Final",
      "connector": "mysql",
      "name": "mysql-server-1",
      "ts_ms": 0,
      "snapshot": false,
      "db": "inventory",
      "table": "customers",
      "server_id": 0,
      "gtid": null,
      "file": "mysql-bin.000003",
      "pos": 154,
      "row": 0,
      "thread": 7
    }
  }
}
```

What is Data Vault?




Order
Create
Site/iOS/Android apps
Pay
Payment Gateway
Cook
Tracker
Deliver
Delivery

Why Data Vault?

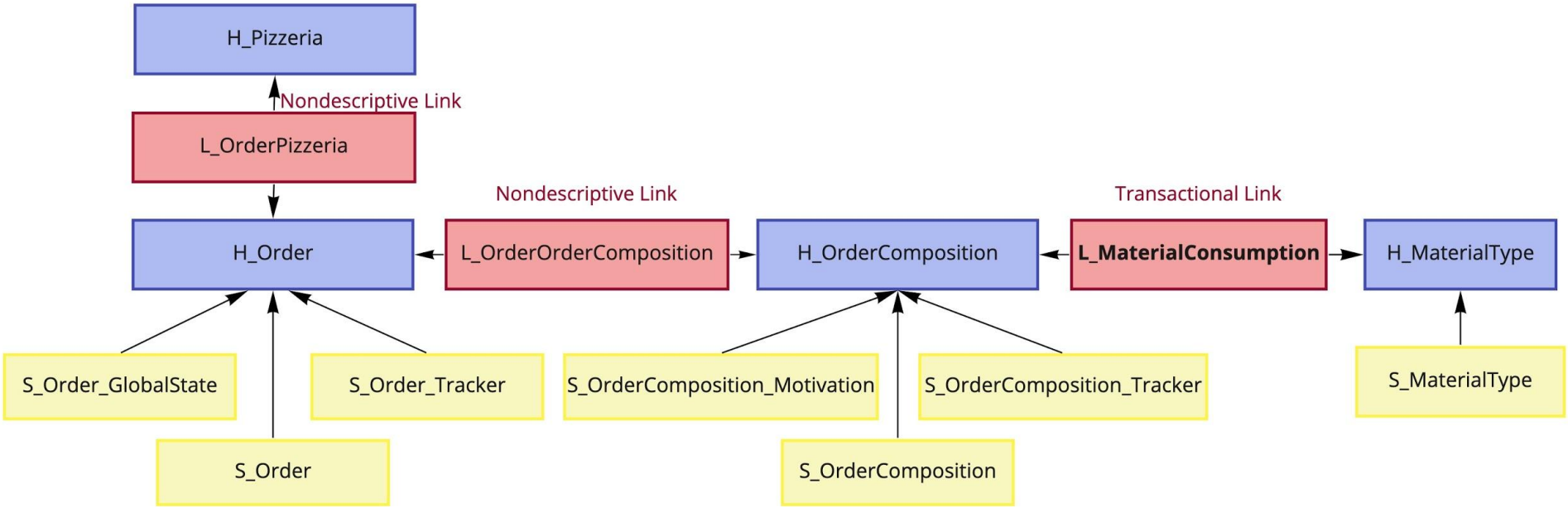
microservices

multiple contexts
for one entity

clear
responsibility
and data sources

fast changing links

Data Vault model

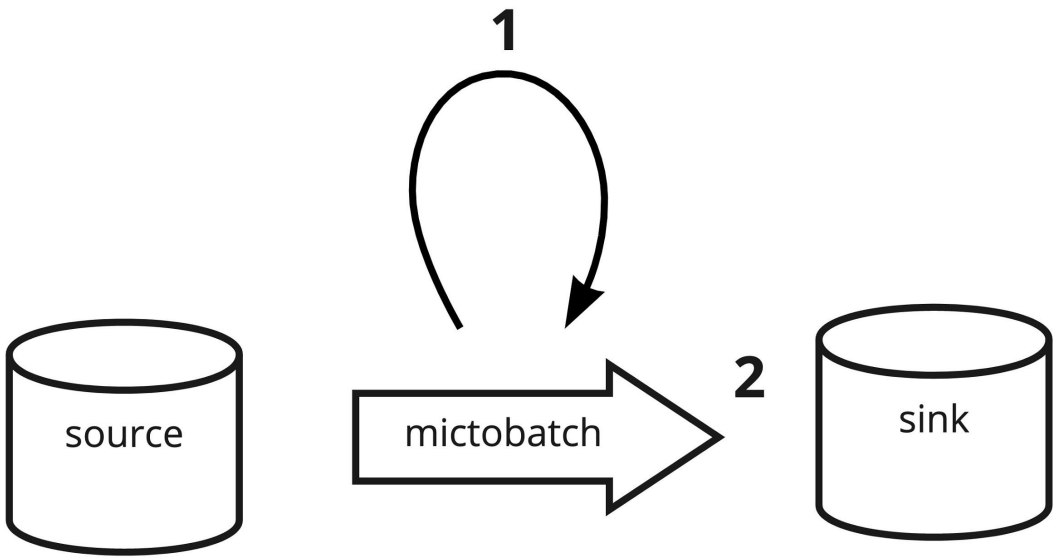


Data ingestion

- without deduplication
- with deduplication
- SCD 2

Demo

Deduplication



SCD 2

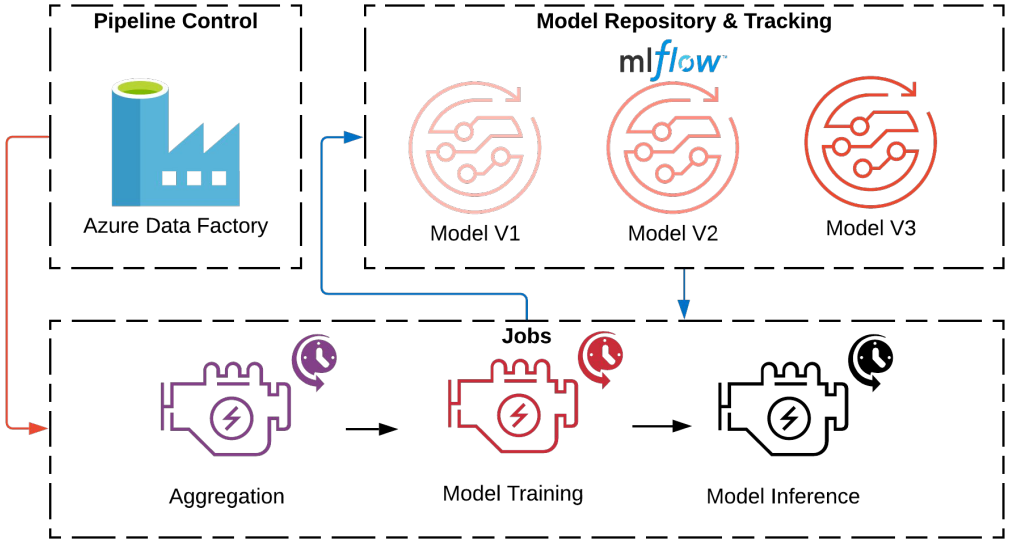
ProductId	ProductName	ProductPrice	Current	EffectiveFrom	EffectiveTo
1	Pepperoni	100	false	2020-06-01 10:00:00	2020-06-02 12:00:00
1	Pepperoni Update	300	true	2020-06-02 12:00:00	9999-12-31 00:00:00
2	Meat	150	false	2020-06-10 00:00:00	2020-06-11 16:00:00
2	Meat pizza Update	120	true	2020-06-11 16:00:00	9999-12-31 00:00:00

Golden layer

Интеграция с ML пайплайнами и MLflow

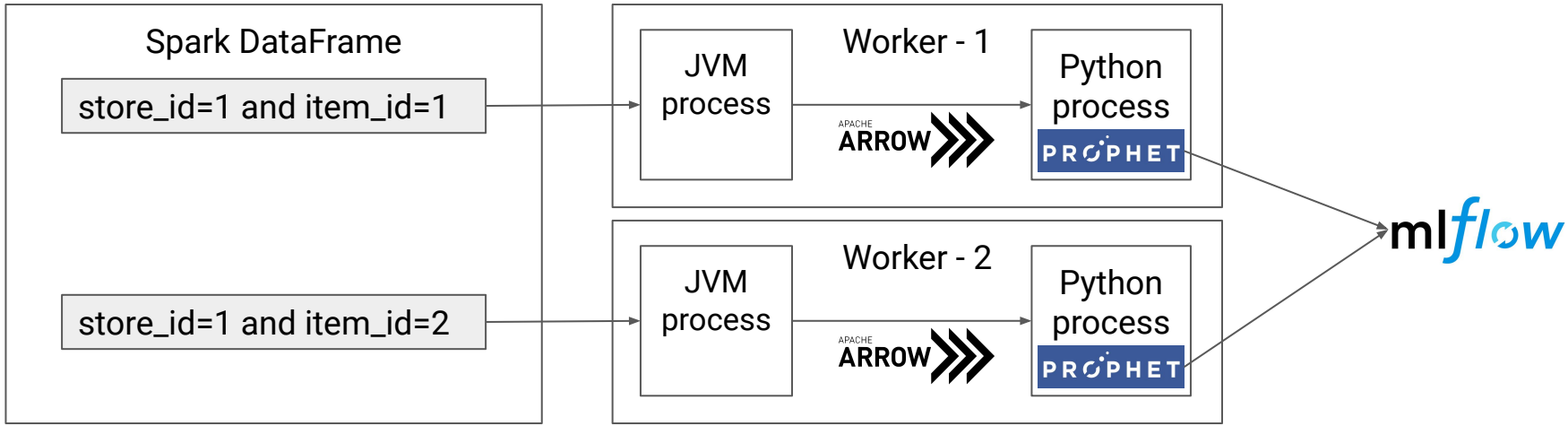
Оркестрация задач

- ADF (Azure Data Factory) запускает пайплайн по расписанию
- ML pipeline управляется через ADF
- В ADF встроена интеграция с Databricks



Распределенное обучение и PyArrow

- Подходит когда есть data parallelism
- Spark доставляет данные по воркерам
- Данные передаются в ру-процесс через PyArrow
- Построенные модели сохраняются в MLflow



Работа с моделями с помощью MLflow

Обучение

```
@pandas_udf(_output_schema, PandasUDFType.GROUPED_MAP)
def train(pdf):
    D_train, D_test = split(pdf)

    with mlflow.start_run() as run:
        # create model object
        model = Prophet()
        model.fit(D_train)

        mse = estimate(model, D_test)

        # log all related parameters
        mlflow.sklearn.log_model(model, "prophet")
        mlflow.log_param("some-param", model.params["some-param"])
        mlflow.log_metric("mse", mse)


        # set all tags for further search
        tags = {“store”: 1, “item”: 1, “table_version”: 1}
        mlflow.set_tags(tags)
```

Скоринг

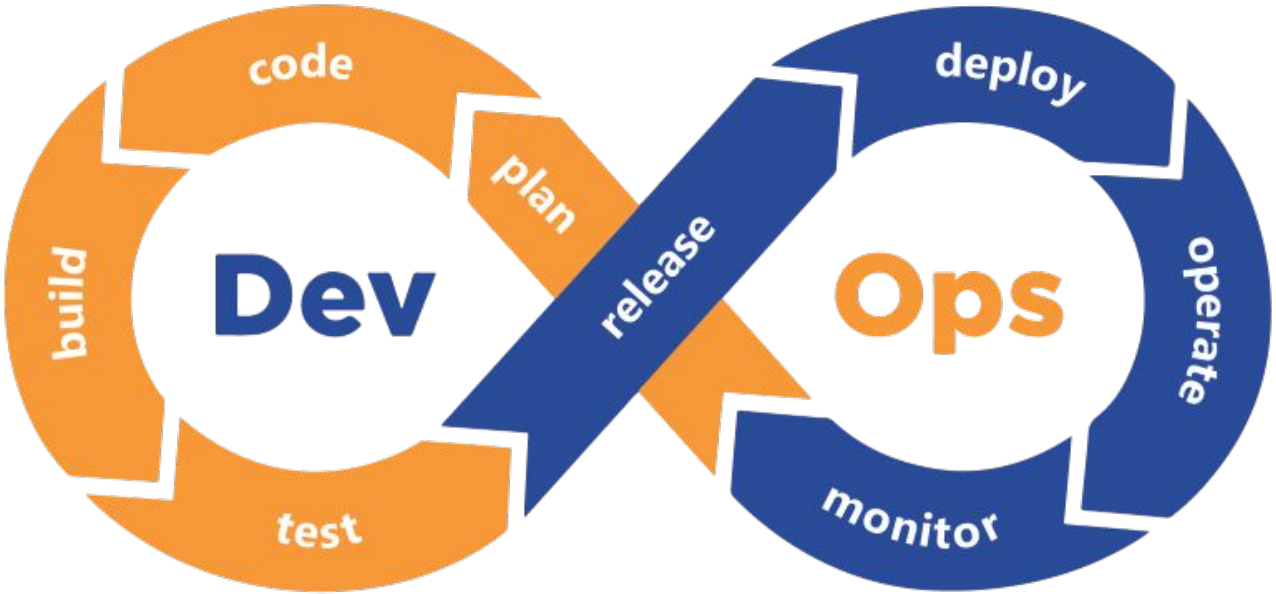
```
@pandas_udf(_output_schema, PandasUDFType.GROUPED_MAP)
def predict(pdf):
    model_uri = get_best_model_uri(tags)
    model = mlflow.sklearn.load_model(model_uri)

    future = model.make_future_dataframe(
        periods=10, freq='D', include_history=False)

    forecast = model.predict(future)

    return forecast
```

CI/CD для пайплайнов данных



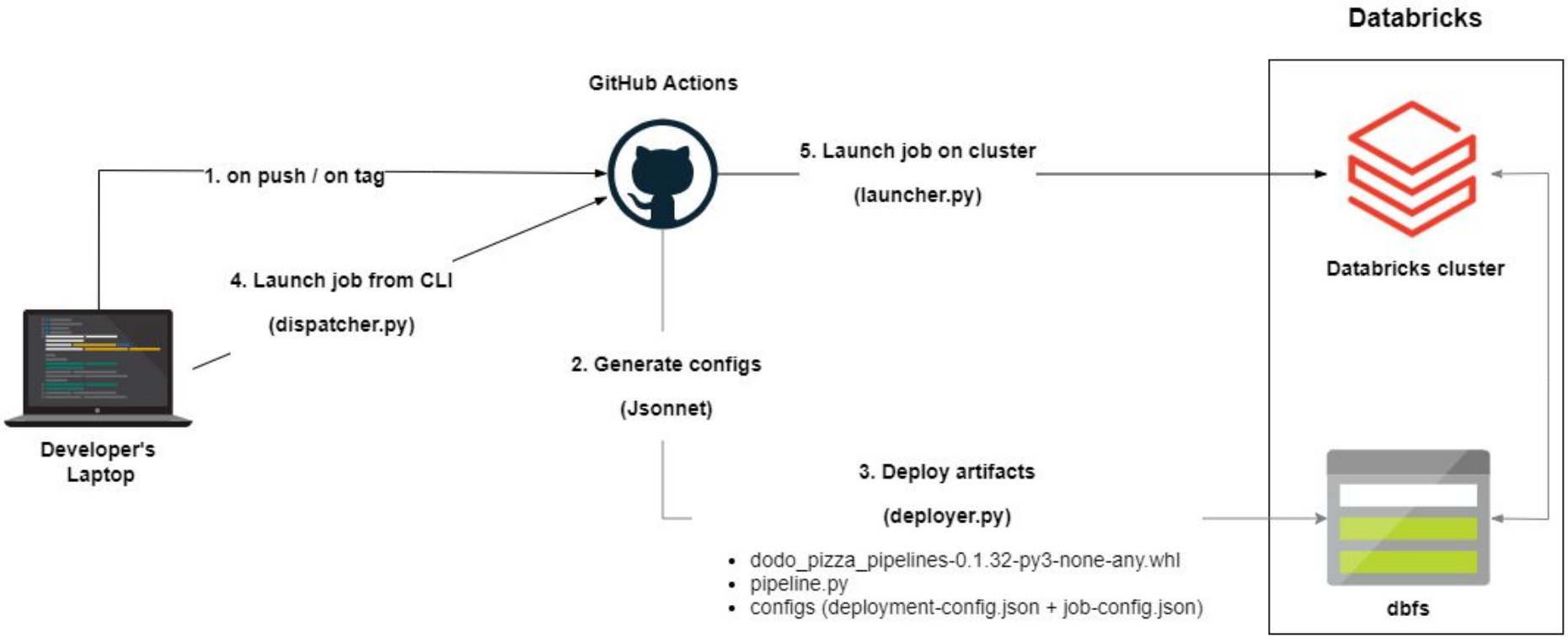
Требования к проекту

- Весь код хранится в репозитории на GitHub.
- Разработка с использованием IDE.
- Деплой наших spark job на кластер в автоматическом или полуавтоматическом режиме (из командной строки).
- Тестируемость решения.
- Возможность выполнять интеграционные тесты на кластере.
- Управление релизами.
- Запуск боевых spark job на кластере с возможностью релизить конкретные job’ы.

Инструменты

- IDE: PyCharm.
- CI/CD workflows: GitHub Actions.
- Deploy & launch jobs: Databricks CLI.
- Configs generation: Jsonnet.
- Tools: Make (Make? Да вы с ума сошли).

Test/Release flow



План DEMO

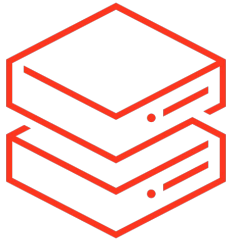
- Структура проекта.
- Как выглядит типовая spark job’a.
- Что мы деплоим на кластер.
- Генерация конфигов (Jsonnet).
- GitHub Actions workflows.
- Роль deployer.py.
- Роль dispatcher.py.
- Роль launcher.py.
- Локальные тесты.
- Публикация релизов и локальный запуск тестов из командной строки (Make).


THAT'S HOW WE DO THINGS
IN THE DEVOPS DIMENSION

Summary

Проблемы

- Поставка данных из оперативной БД
- Эффективное хранение данных в Data Lake
- Интегрированное обучение ML моделей



Решение

- Databricks + Spark Streaming для обработки данных
- Data Vault на Delta Lake
- MLflow для управления жизненным циклом моделей

Value

- Масштабируемое решение на managed платформе
- Понятная и управляемая модель данных
- Качественные прогнозы для бизнеса

Финальный Q&A