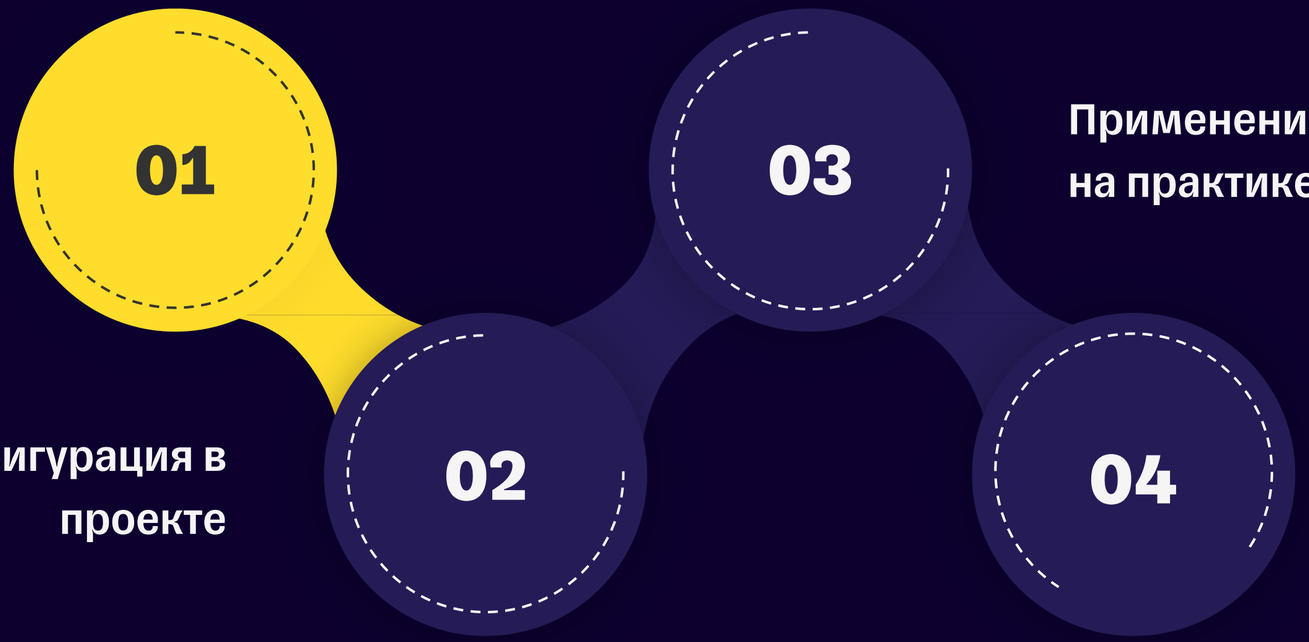
Т БАНК

Debezium Engine: практическое руководство по использованию

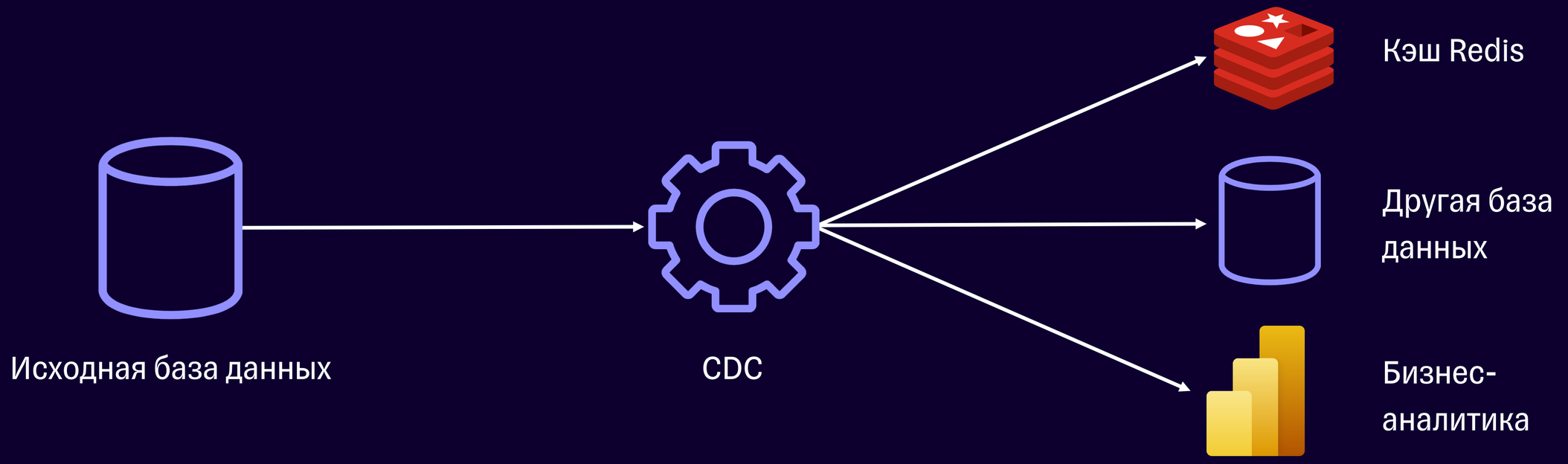
Анастасия Сашина | Java/Kotlin разработчик Т-Банка

План доклада



Откуда взялся Debezium Engine?

Захват изменения данных Change Data Capture (CDC)[1]



1 – Confluent: What is Change Data Capture ? (https://clck.ru/3CPzAr)

Применение Change Data Capture

- Репликация данных из базы данных;

Применение Change Data Capture

- Репликация данных из базы данных;
- Интеграция с системами бизнес-аналитики;

Применение Change Data Capture

- Репликация данных из базы данных;
- Интеграция с системами бизнес-аналитики;
- Репликация данных с обогащением/обработкой данных.

Применение Change Data Capture

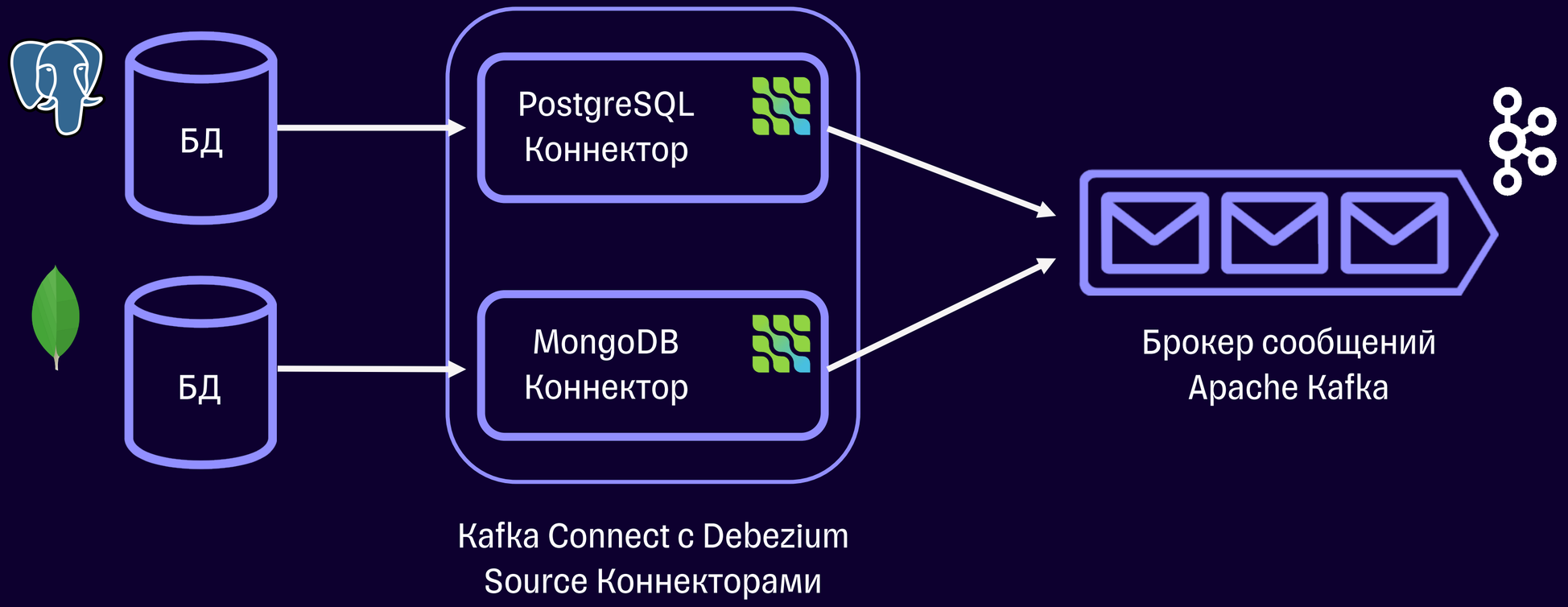
- Репликация данных из базы данных;
- Интеграция с системами бизнес-аналитики;
- Репликация данных с обогащением/обработкой данных.
- ...

Debezium[2]

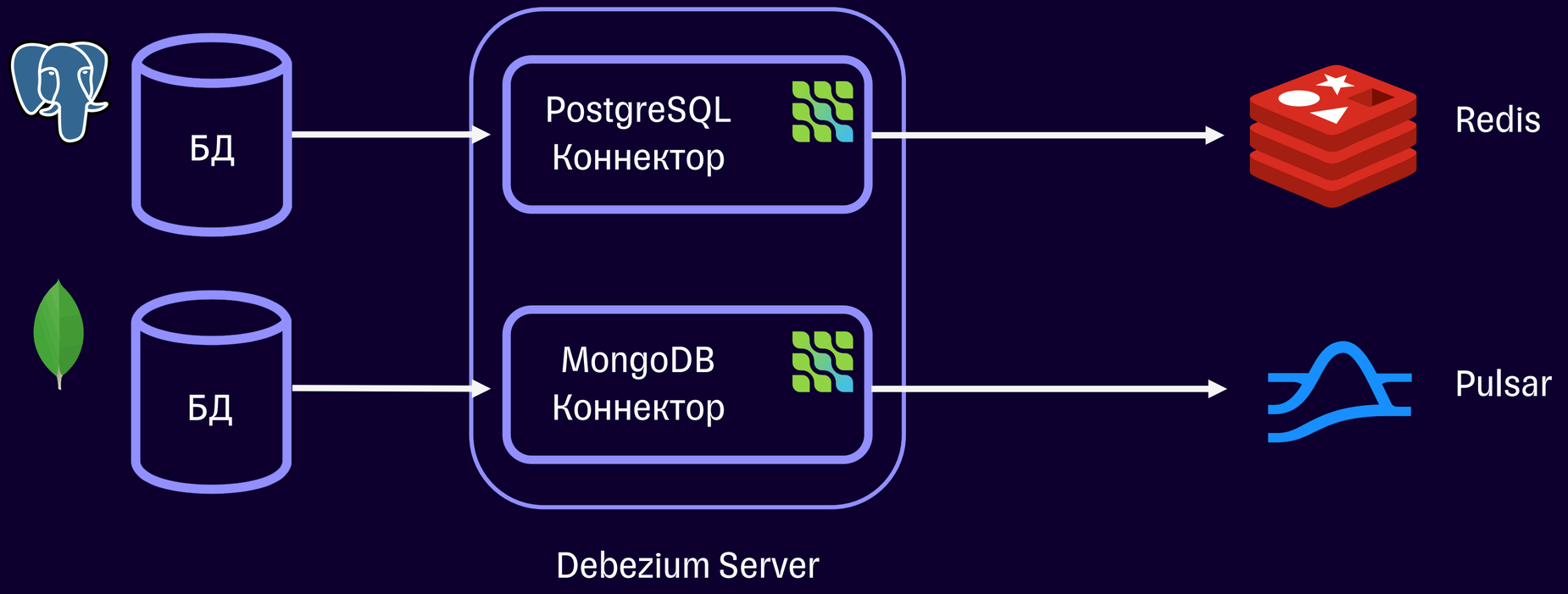
Платформа, созданная для захвата изменения данных, использующая журналы транзакций базы данных. Реализует паттерн Change Data Capture.

2 – Debezium: Документация (https://clck.ru/3CPzHp)

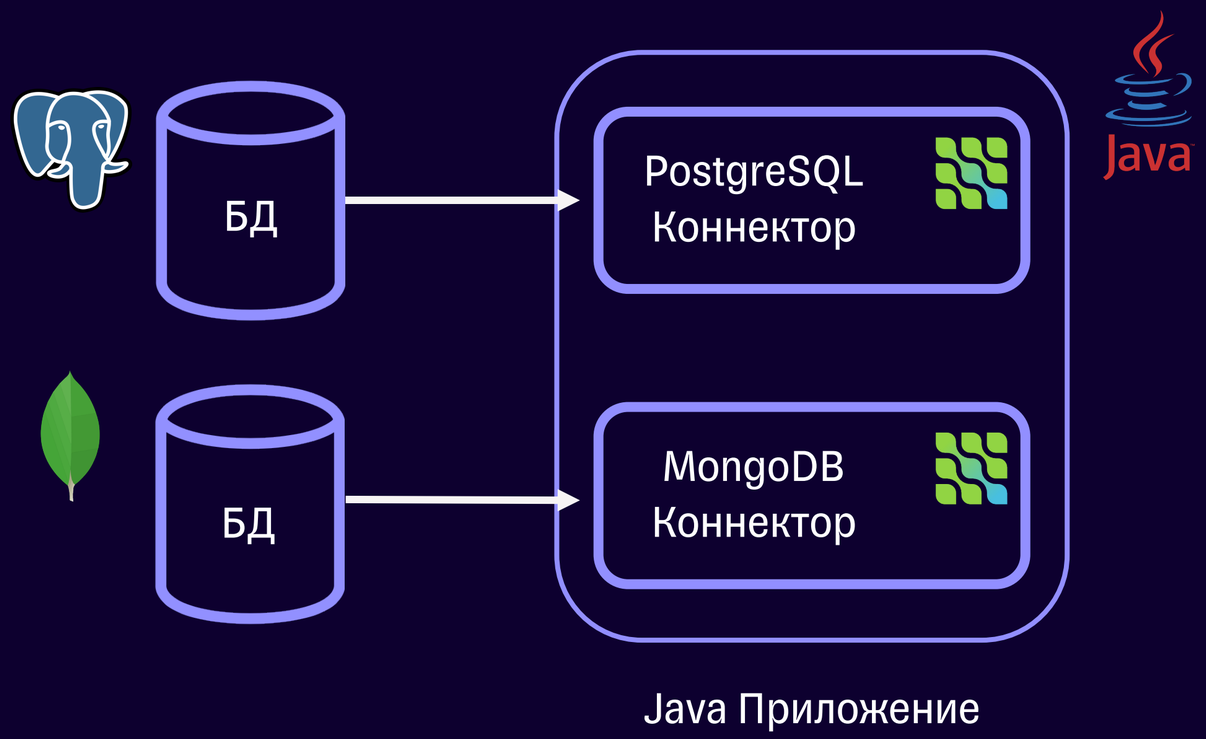
Debezium как Apache Kafka Source Connector

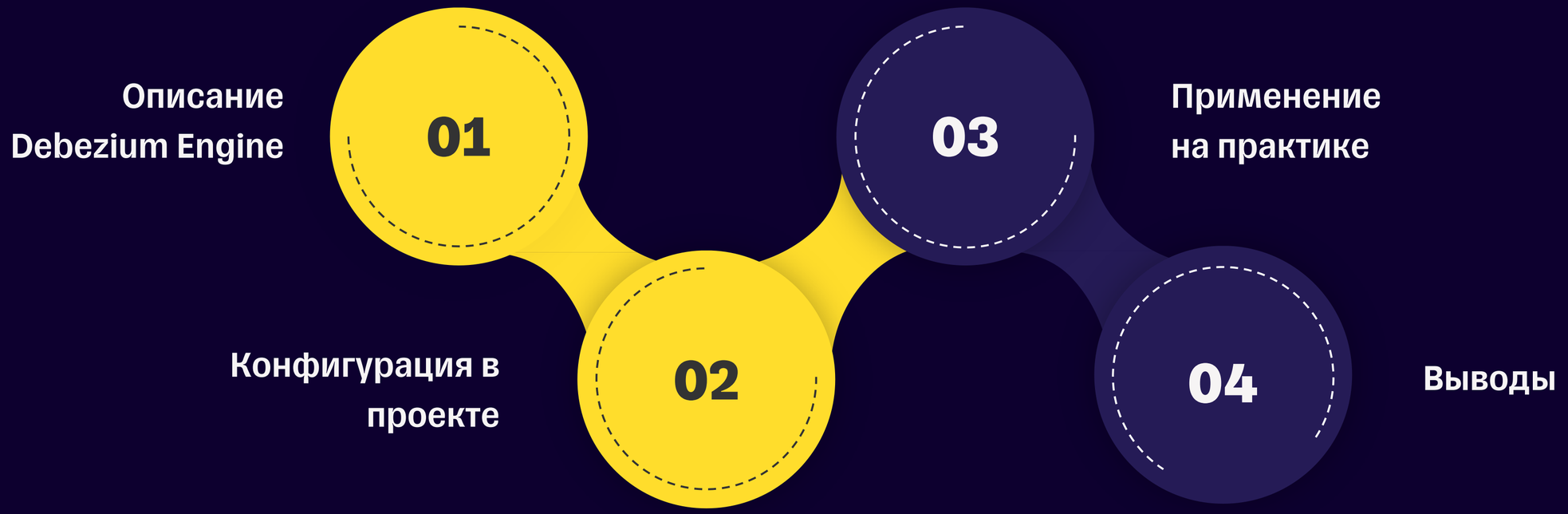


Debezium Server

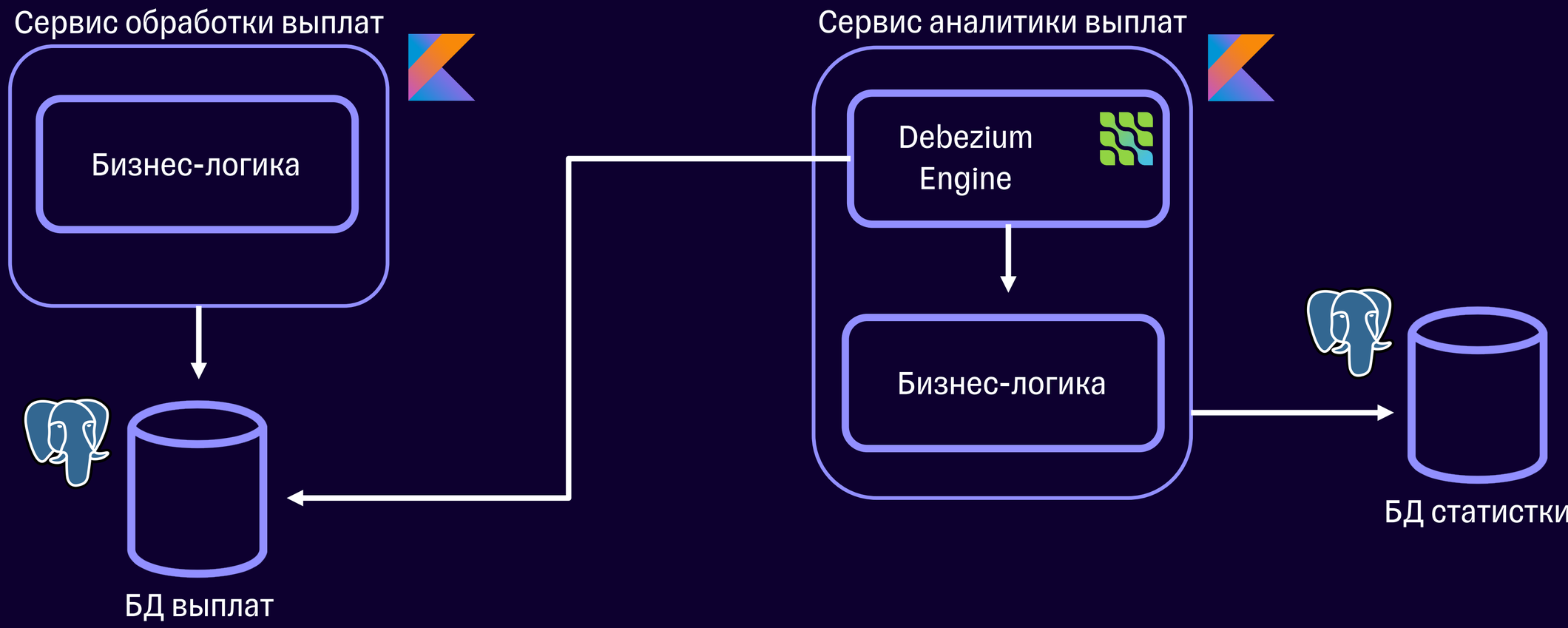


Debezium Engine

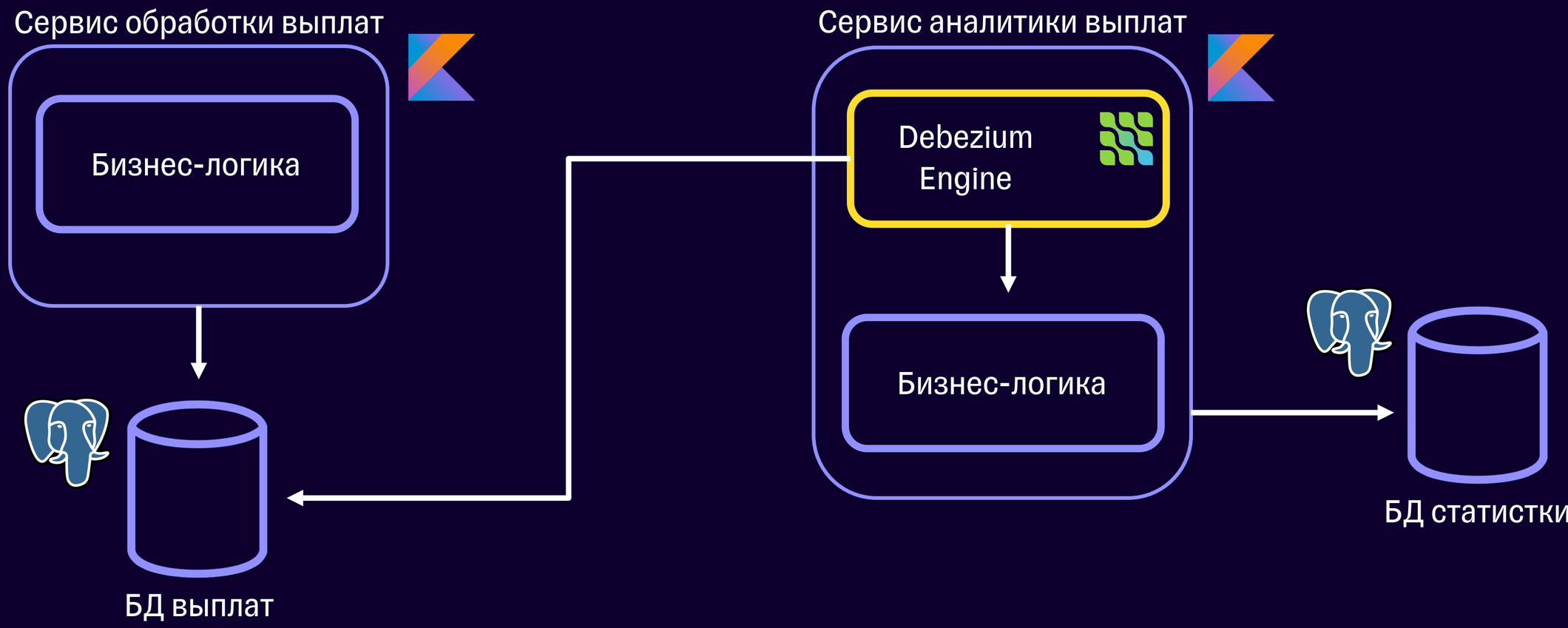



Описание
Debezium Engine
01
Конфигурация в
проекте
02
03
Применение
на практике
04
Выводы

Структура тестового приложения



Структура тестового приложения



Структура объекта «Выплата»

```
Payment
  (
     id          BIGSERIAL PRIMARY KEY,
     payer       VARCHAR(250) NOT NULL,
     recipient   VARCHAR(250) NOT NULL,
     amount      NUMERIC(20, 2) NOT NULL,
     status      VARCHAR(50) NOT NULL,
     comment     VARCHAR(5000)
  )
```

Как использовать?

Конфигурация Debezium Engine для PostgreSQL

```
{
  "name": "analyse-payment-engine",
  "topic.prefix": "analyse-payment-topic",
  "connector.class": "*.PostgresConnector",
  "offset.storage": "*.FileOffsetBackingStore",
  "offset.storage.file.filename": "offsets.dat",
  "database.hostname": "localhost",
  "database.port": "5435",
  "database.user": ”postgres",
  "database.password": "pa$$word",
  "database.dbname": "payments",
  "plugin.name": "pgoutput"
}
```

Конфигурация Debezium Engine для PostgreSQL

```
{
  "name": "analyse-payment-engine",
  "topic.prefix": "analyse-payment-topic",
  "connector.class": "*.PostgresConnector",
  "offset.storage": "*.FileOffsetBackingStore",
  "offset.storage.file.filename": "offsets.dat",
  "database.hostname": "localhost",
  "database.port": "5435",
  "database.user": ”postgres",
  "database.password": "pa$$word",
  "database.dbname": "payments",
  "plugin.name": "pgoutput"
}
```

Инициализация обработчика событий

```
@Component
class DebeziumListener(debeziumConfig: io.debezium.config.Configuration) {

    private val debeziumEngine = DebeziumEngine
        .create(ChangeEventFormat.of(Connect::class.java))
        .using(debeziumConfig.asProperties())
        .notifying { /* Обрабатываем событие об изменении */ }
        .build()

}
```

Инициализация обработчика событий

```
@Component
class DebeziumListener(debeziumConfig: io.debezium.config.Configuration) {

    private val debeziumEngine = DebeziumEngine
        .create(ChangeEventFormat.of(Connect::class.java))
        .using(debeziumConfig.asProperties())
        .notifying { /* Обрабатываем событие об изменении */ }
        .build()

}
```

Инициализация обработчика событий

```
@Component
class DebeziumListener(debeziumConfig: io.debezium.config.Configuration) {

    private val debeziumEngine = DebeziumEngine
        .create(ChangeEventFormat.of(Connect::class.java))
        .using(debeziumConfig.asProperties())
        .notifying { /* Обрабатываем событие об изменении */ }
        .build()

}
```

Инициализация обработчика событий

```
@Component
class DebeziumListener(debeziumConfig: io.debezium.config.Configuration) {

    private val debeziumEngine = DebeziumEngine
        .create(ChangeEventFormat.of(Connect::class.java))
        .using(debeziumConfig.asProperties())
        .notifying { /* Обрабатываем событие об изменении */ }
        .build()

}
```

Пример записи об изменении выплаты

```
{
  "op": "c",
  "after": {
    "id": 2,
    "payer": "Alice",
    "recipient": "Bob",
    "amount": 150.00,
    "status": "NEW"
  },
  "before": null,
  "source": {
    "connector": "postgresql",
    "name": "analyse-payment-topic",
    "ts_ms": 1723190956635,
    "db": "payment",
    "schema": "public",
    "table": "payments",
    …
  },
  …
}
```

Пример записи об изменении выплаты

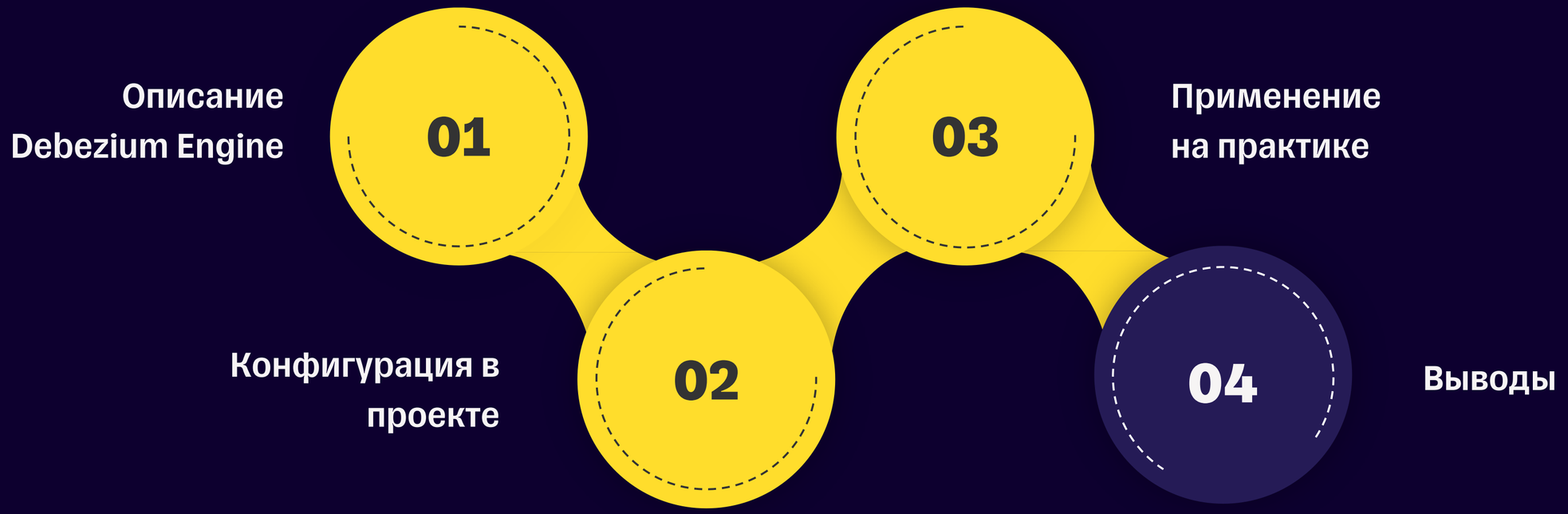
```
{
  "op": "c",
  "after": {
    "id": 2,
    "payer": "Alice",
    "recipient": "Bob",
    "amount": 150.00,
    "status": "NEW"
  },
  "before": null,
  "source": {
    "connector": "postgresql",
    "name": "analyse-payment-topic",
    "ts_ms": 1723190956635,
    "db": "payment",
    "schema": "public",
    "table": "payments",
    …
  },
  …
}
```

Пример записи об изменении выплаты

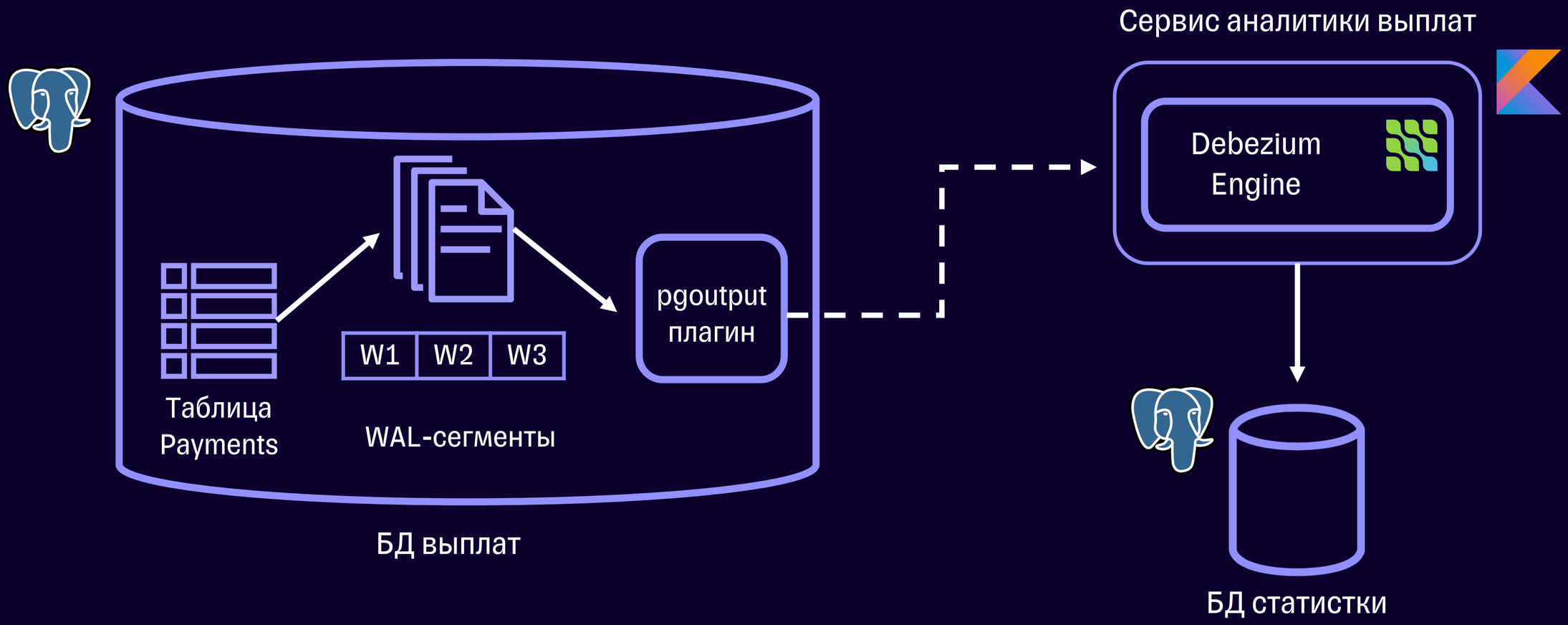
```
{
  "op": "c",
  "after": {
    "id": 2,
    "payer": "Alice",
    "recipient": "Bob",
    "amount": 150.00,
    "status": "NEW"
  },
  "before": null,
  "source": {
    "connector": "postgresql",
    "name": "analyse-payment-topic",
    "ts_ms": 1723190956635,
    "db": "payment",
    "schema": "public",
    "table": "payments",
    …
  },
  …
}
```

Пример записи об изменении выплаты

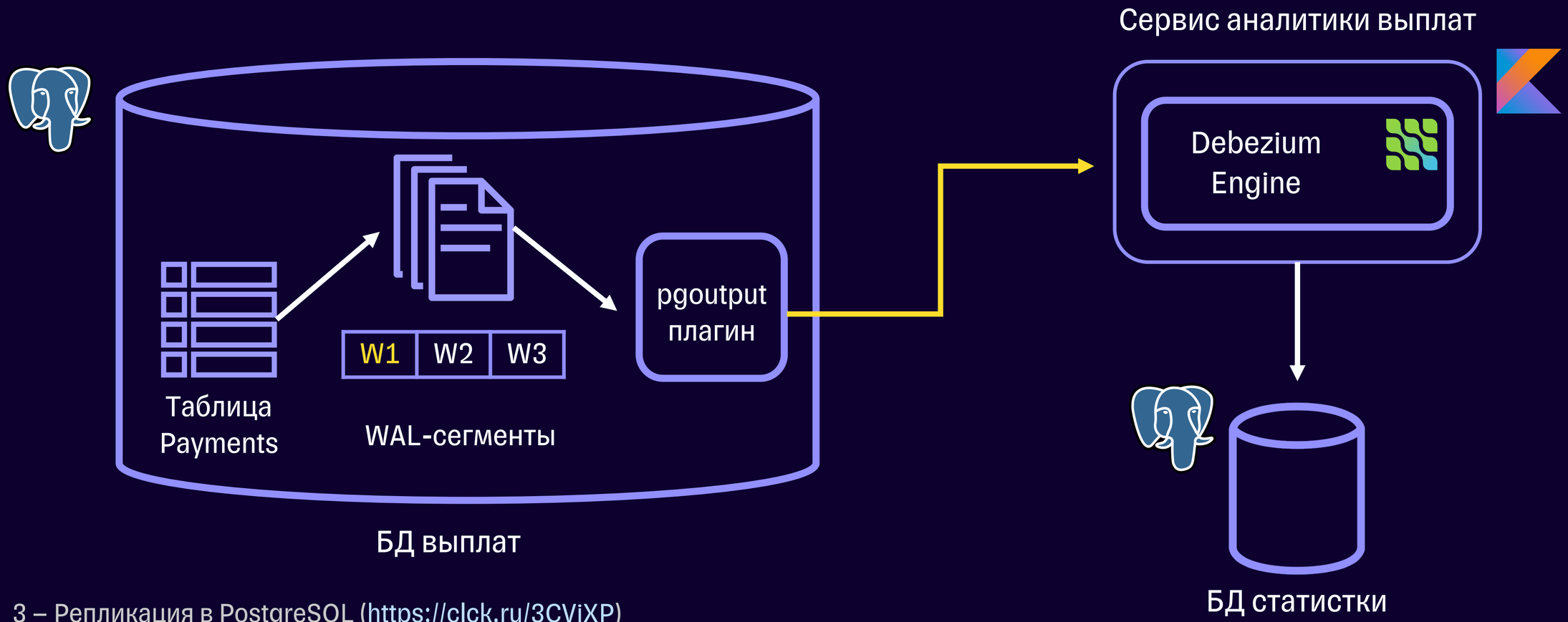
```
{
  "op": "c",
  "after": {
    "id": 2,
    "payer": "Alice",
    "recipient": "Bob",
    "amount": 150.00,
    "status": "NEW"
  },
  "before": null,
  "source": {
    "connector": "postgresql",
    "name": "analyse-payment-topic",
    "ts_ms": 1723190956635,
    "db": "payment",
    "schema": "public",
    "table": "payments",
    …
  },
  …
}
```


Описание
Debezium Engine
01
Конфигурация в
проекте
02
03
Применение
на практике
04
Выводы

Первый запуск

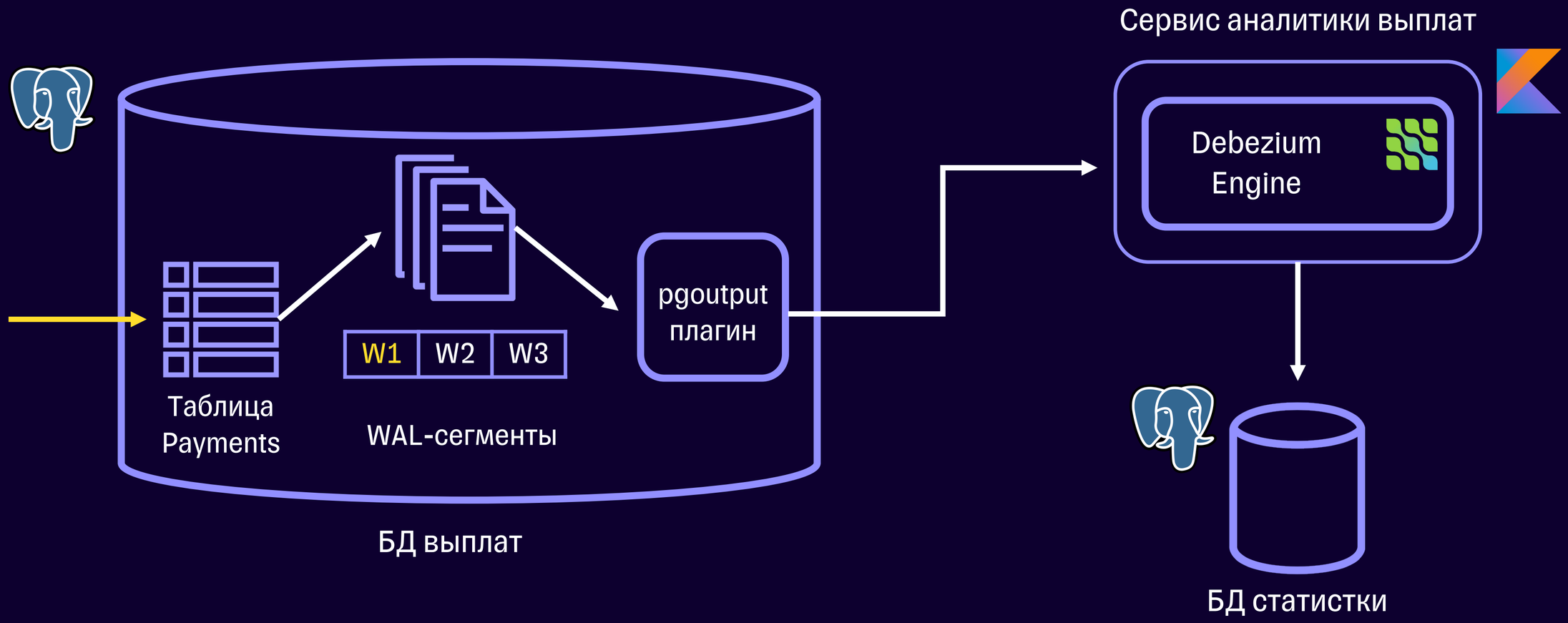


Первый запуск[3]

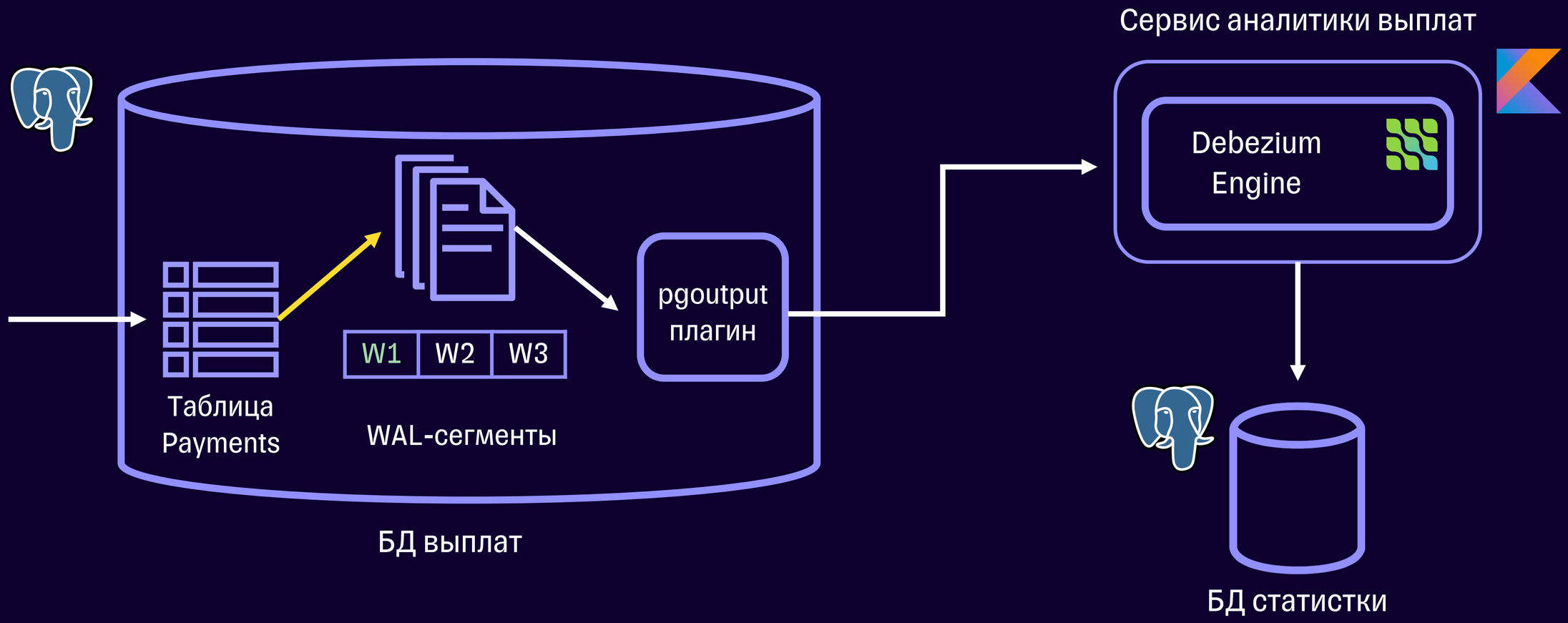


3 – Репликация в PostgreSQL (https://clck.ru/3CViXP)

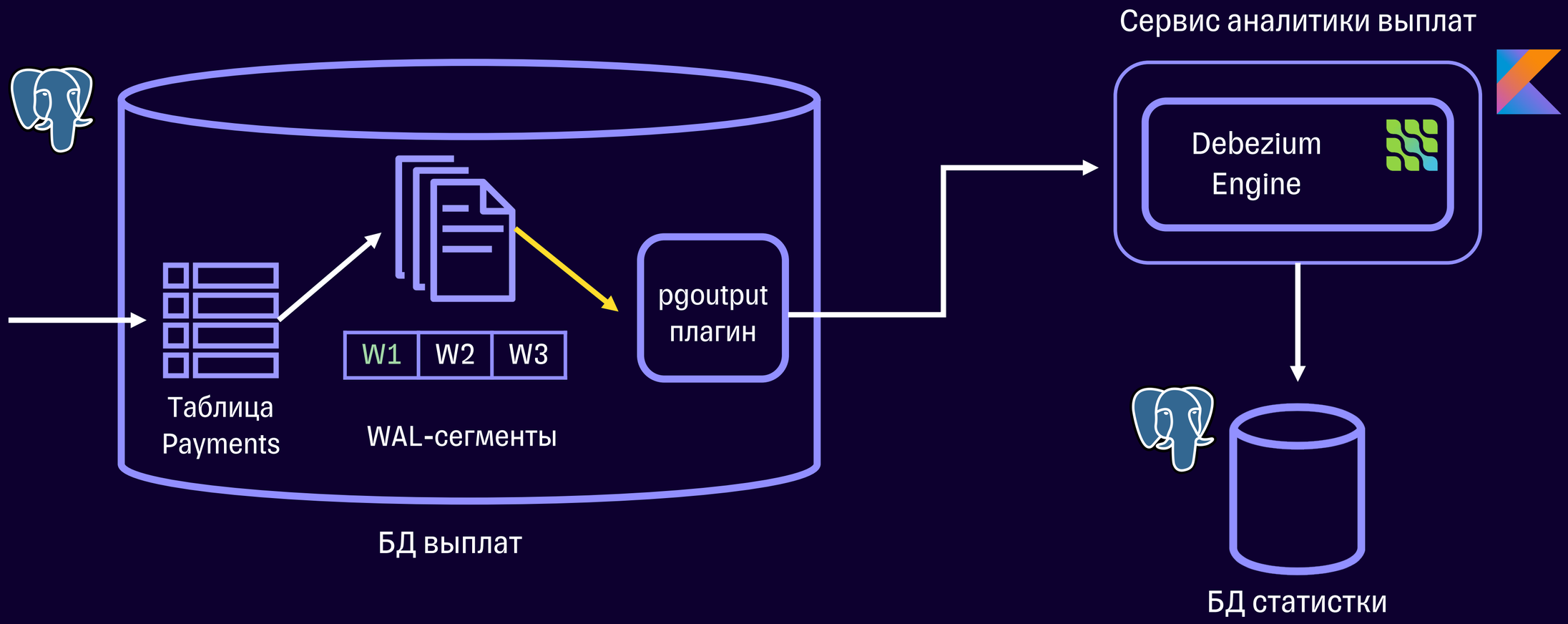
Получение и обработка изменений



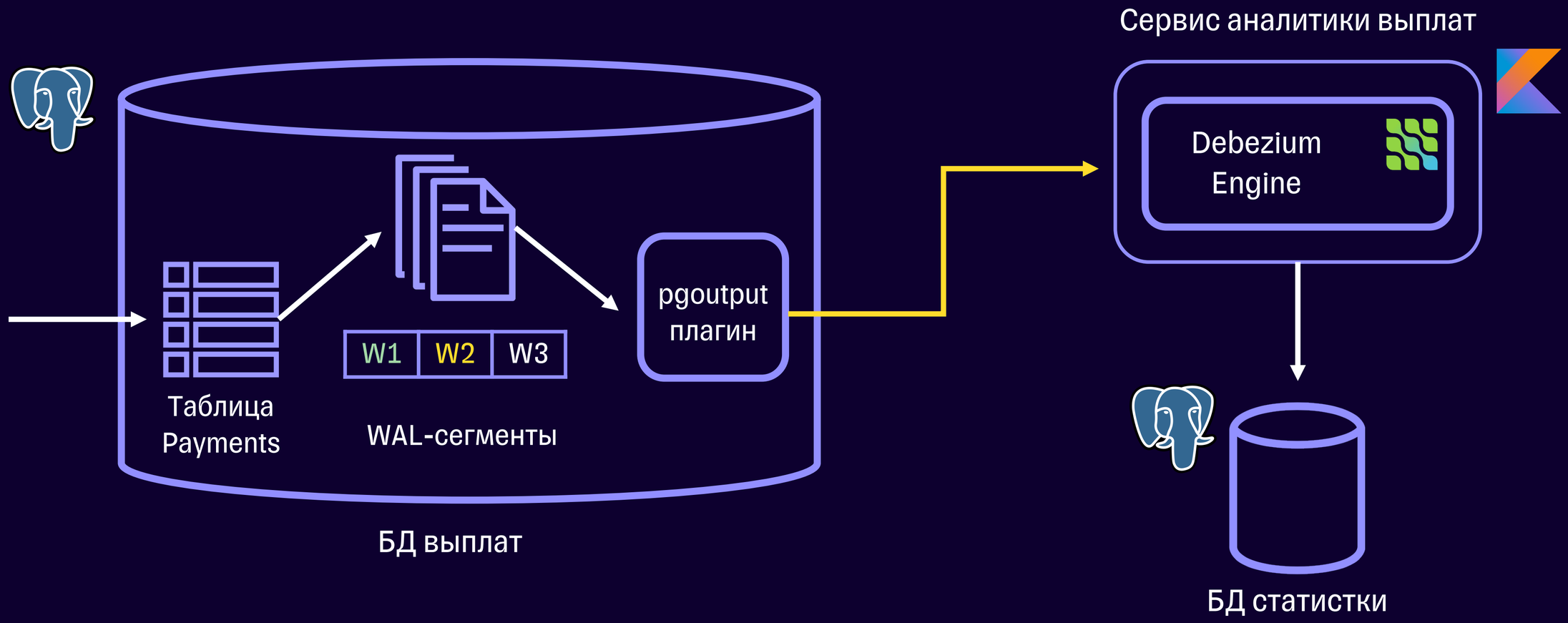
Получение и обработка изменений



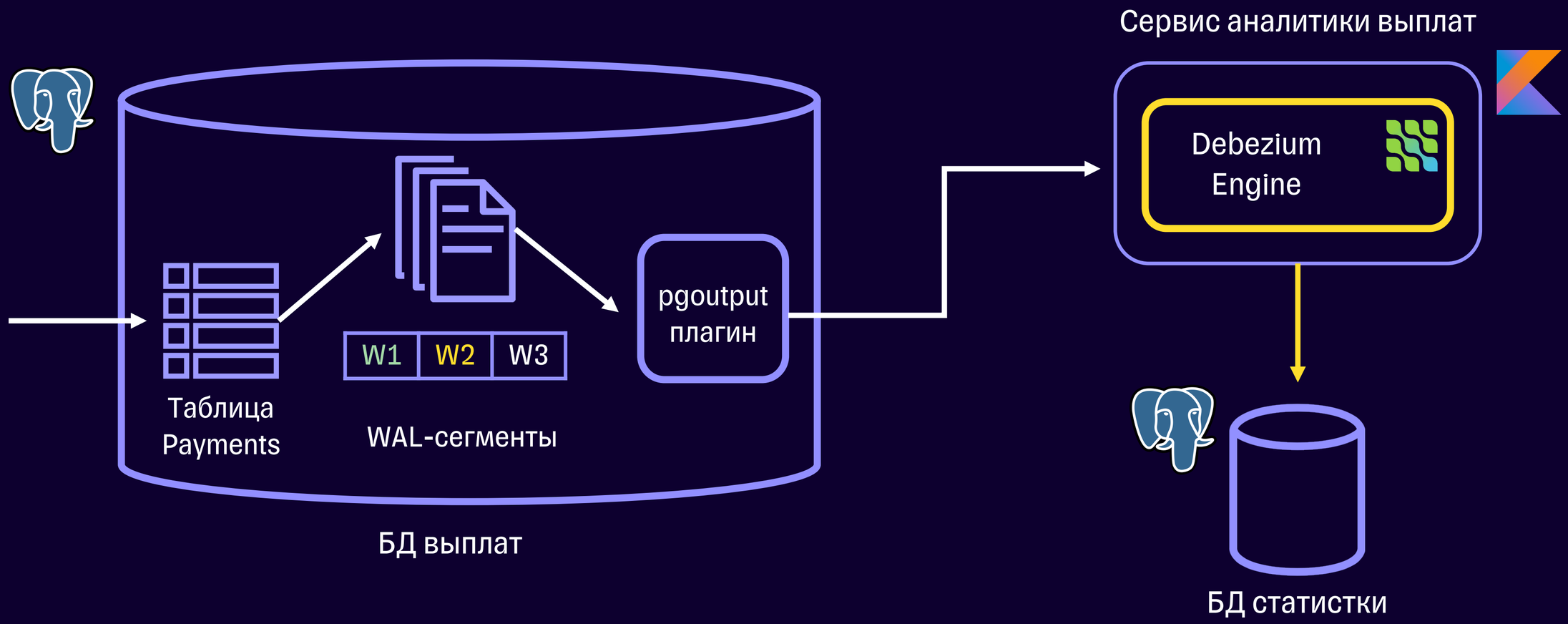
Получение и обработка изменений



Получение и обработка изменений

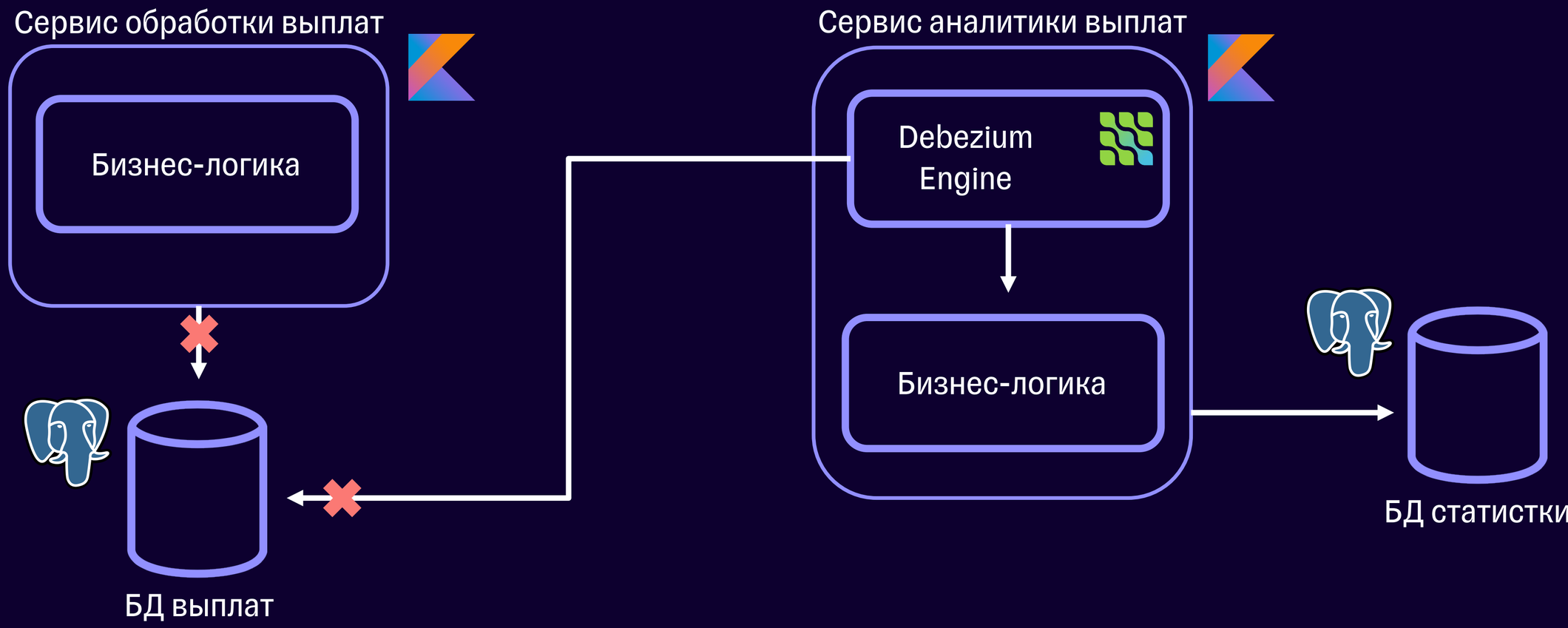


Получение и обработка изменений



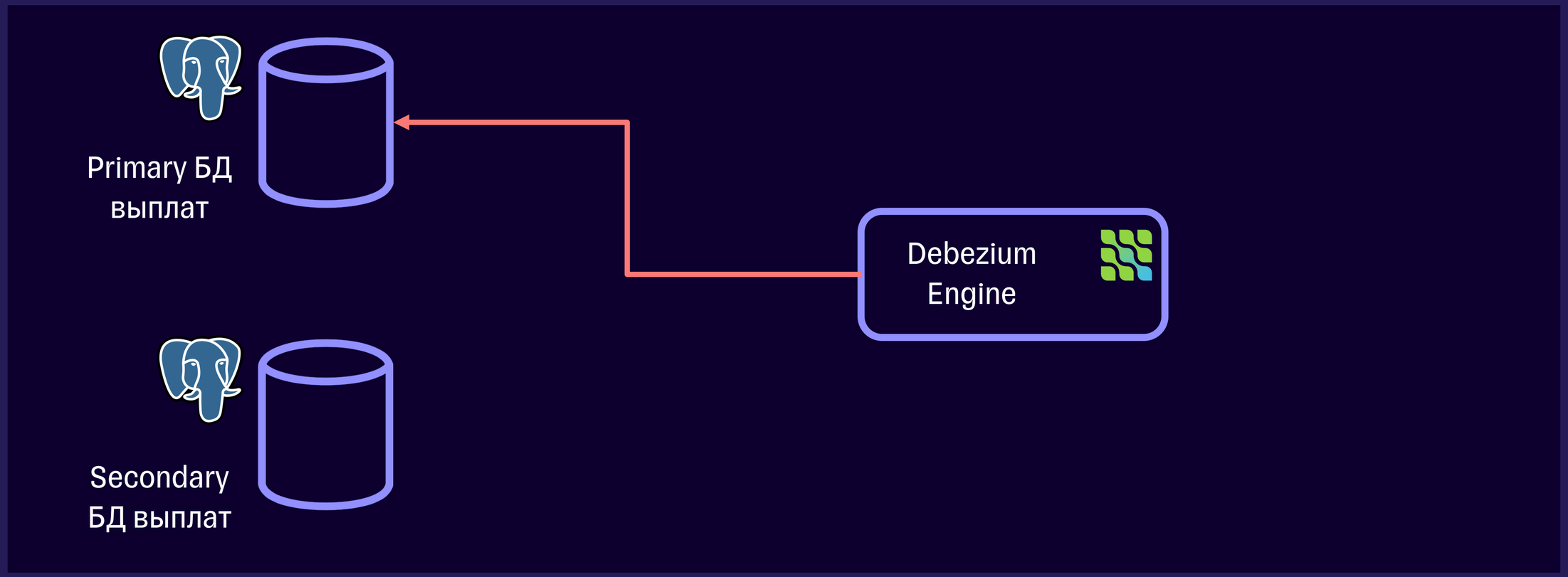
А что, если ..?

... упадет БД?


Сервис обработки выплат
Бизнес-логика
Сервис аналитики выплат
Debezium
Engine
Бизнес-логика
БД выплат
БД статистки

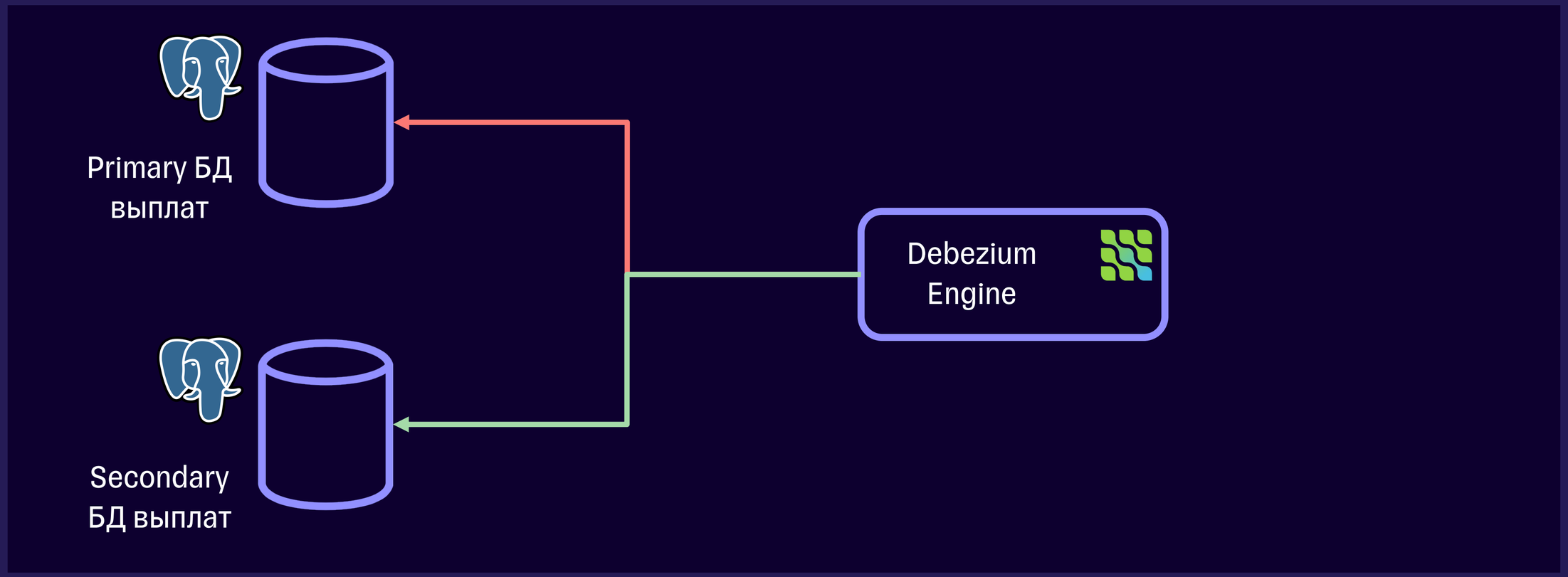
Идея!

Добавить Secondary базу данных.



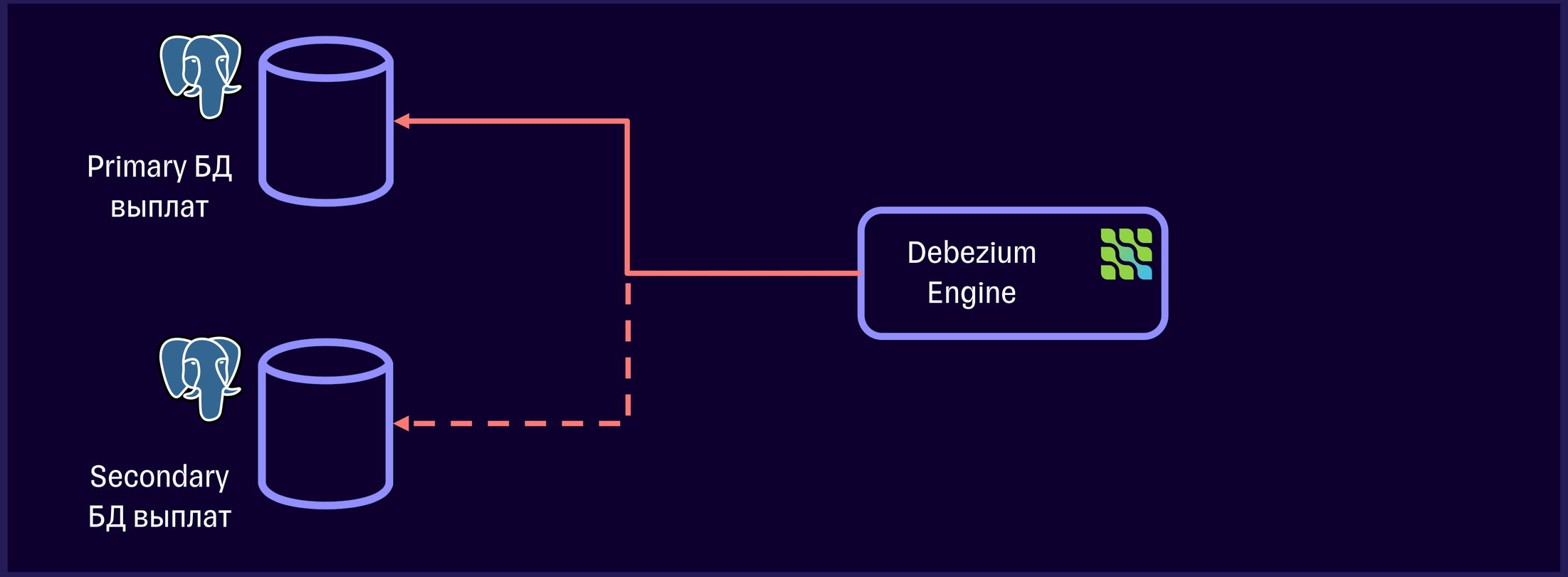
Ожидание

Debezium Engine имеет механизм поддержки Failover из коробки.

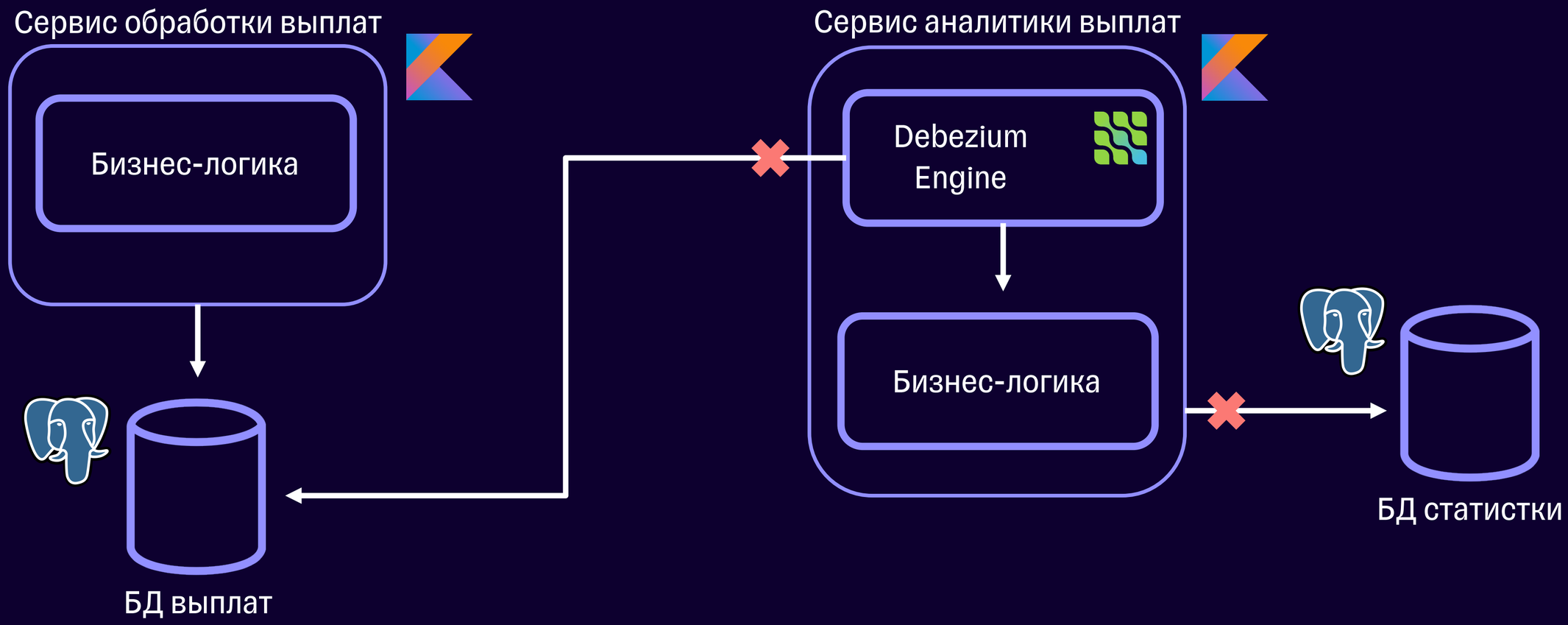


Реальность

У Debezium Engine нет такого механизма переключения, т.к. параметры конфигурации предполагают схемы «Один коннектор – Одна база данных».

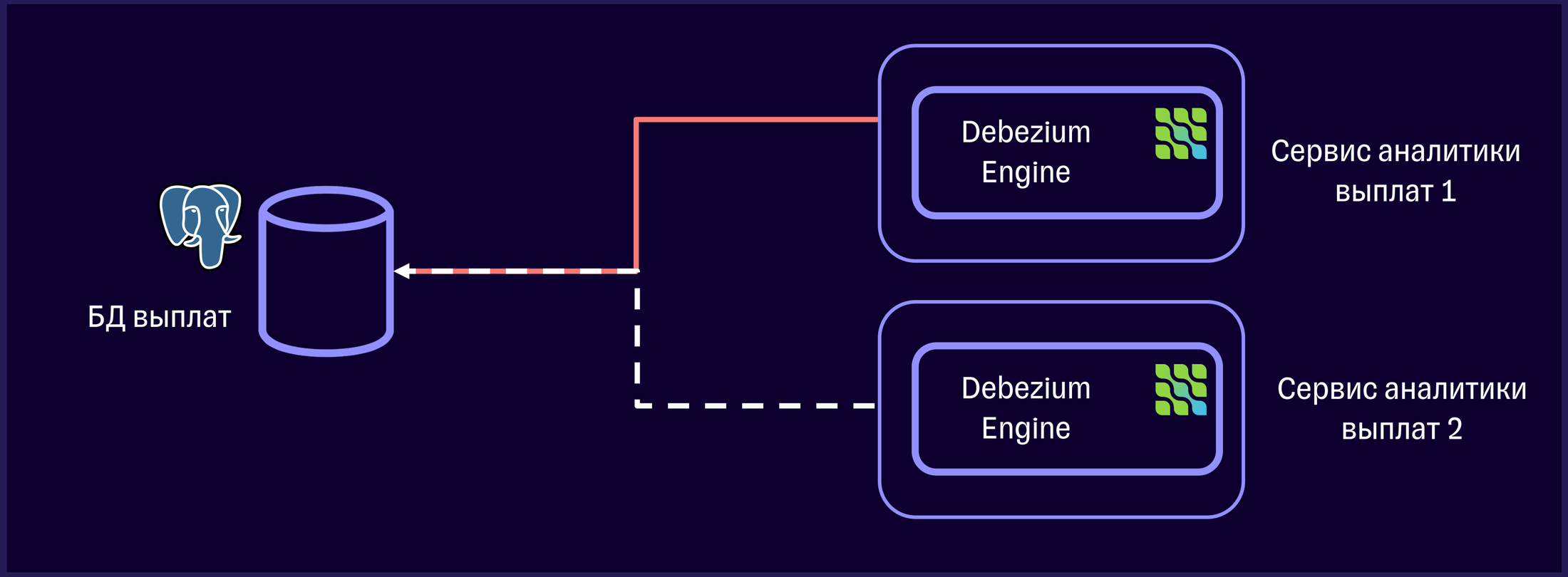


... упадет сервис аналитики?



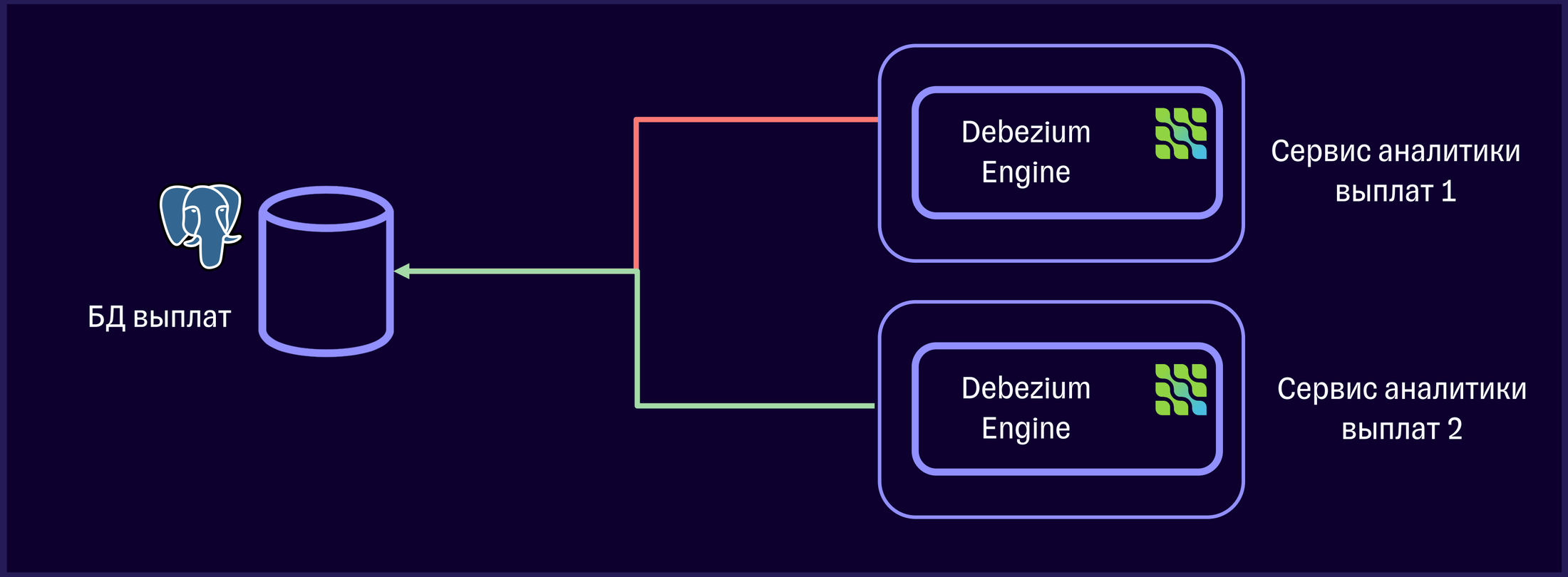
Идея!

Добавить еще один сервис аналитики с Debezium Engine.



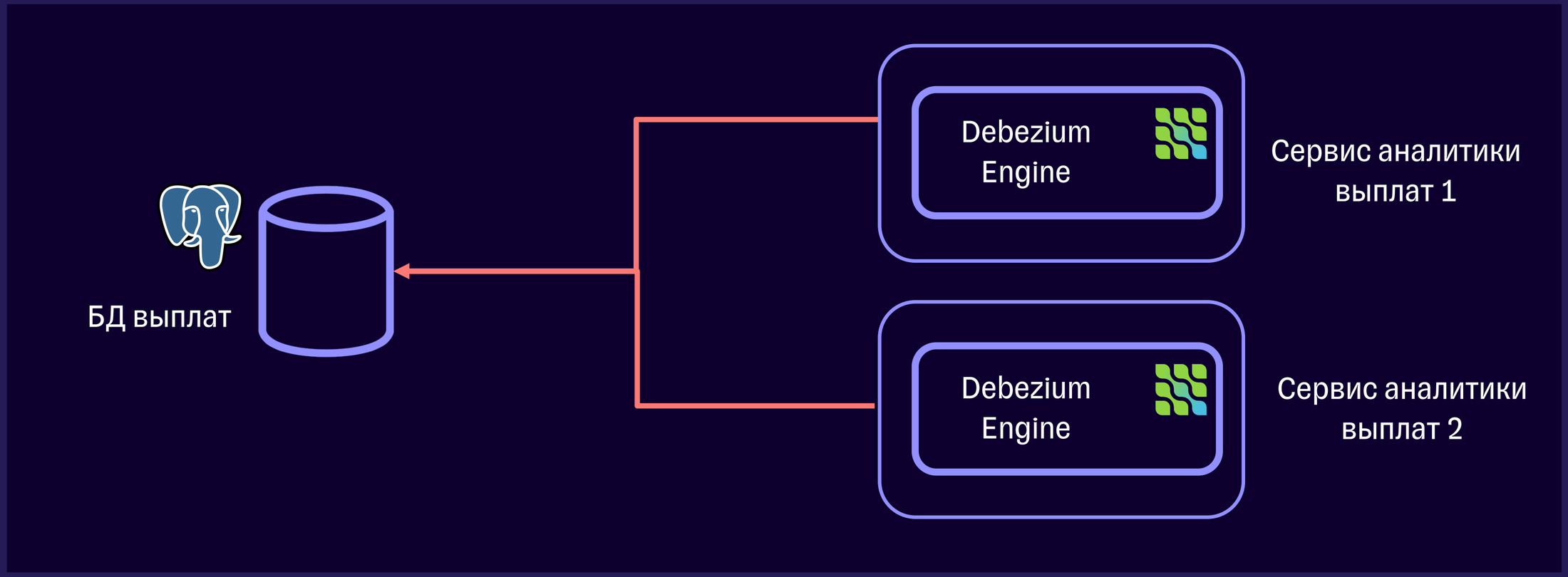
Ожидание

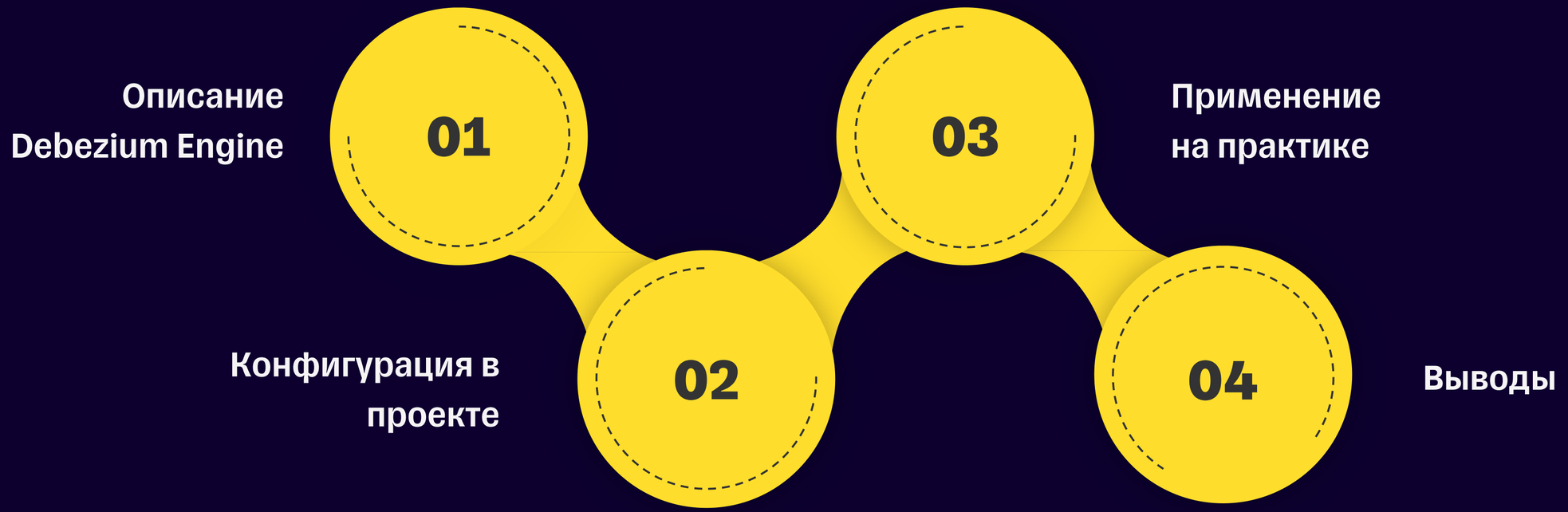
Debezium Engine можно горизонтально масштабировать.



Реальность

Debezium Engine такое решение из коробки не поддерживает.




Описание
Debezium Engine
01
Конфигурация в
проекте
02
03
Применение
на практике
04
Выводы

Преимущества и недостатки Debezium Engine

- Нет необходимости в конфигурации дополнительных сервисов, например Apache Kafka;

Преимущества и недостатки Debezium Engine

- Нет необходимости в конфигурации дополнительных сервисов, например Apache Kafka;
- **Поддержка различных коннекторов из коробки;**

Преимущества и недостатки Debezium Engine

- Нет необходимости в конфигурации дополнительных сервисов, например Apache Kafka;
- Поддержка различных коннекторов из коробки;
- **Отправка сообщений в почти реальном времени.**

Преимущества и недостатки Debezium Engine

- Нет необходимости в конфигурации дополнительных сервисов, например Apache Kafka;
- Поддержка различных коннекторов из коробки;
- Отправка сообщений в почти реальном времени.

- **Нетривиальная настройка;**

Преимущества и недостатки Debezium Engine

- Нет необходимости в конфигурации дополнительных сервисов, например Apache Kafka;
- Поддержка различных коннекторов из коробки;
- Отправка сообщений в почти реальном времени.

- Нетривиальная настройка;
- **Для поддержки Failover нужно кастомное решение;**

Преимущества и недостатки Debezium Engine

- Нет необходимости в конфигурации дополнительных сервисов, например Apache Kafka;
- Поддержка различных коннекторов из коробки;
- Отправка сообщений в почти реальном времени.

- Нетривиальная настройка;
- Для поддержки Failover нужно кастомное решение;
- **Нет возможности горизонтального масштабирования.**

Заключение

Готовое решение без дополнительных сервисов.

Заключение

- Готовое решение без дополнительных сервисов.
- Для высоконагруженных приложений – нужно докручивать.

Заключение

- Готовое решение без дополнительных сервисов.
- Для высоконагруженных приложений – нужно докручивать.
- Можно использовать в тех системах, где ограничены ресурсы на выделение новых сервисов.

Анастасия Сашина

Java/Kotlin разработчик

andr0me

Ссылка на тестовый проект

Т БАНК

Поделитесь

мнением

Спасибо!