Я Доставка

Архитектура платформы данных

План лекции

- знакомство
- базовая теория в архитектуре
- продовый пример архитектуры современной платформы данных
- платформа данных как продукт

Знакомство

- выпускник бакалавриата МФТИ
- выпускник кафедры БИТ по направлению “Высоконагруженные распределенные системы”
- руководитель платформы данных Яндекс Доставки
- работал в X5, Eapteka, SBER

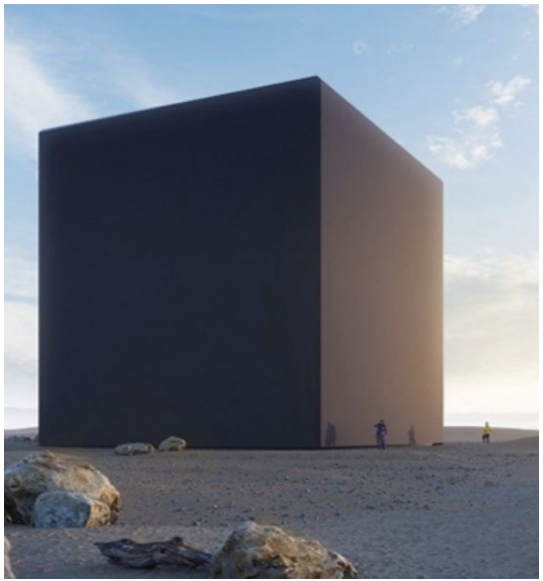
Базовая теория в архитектуре

- Эволюция архитектуры
  - Монолиты
  - Микросервисы и ESB
  - Data Lake
- Форматы хранилищ данных
  - Реляционные (SQL)
  - Key-value
- Подходы к обработке данных
  - Batch
  - Stream
- Современные архитектуры
  - Lambda vs Kappa

Эволюция архитектуры

Монолиты

- Одна глобальная база данных
- Одно большое приложение
- Только реляционные структуры
- Большой IO
- Stored Procedures
- Предназначена для операционной нагрузки
- Вся бизнес-логика завязана на работу с данной БД



Монолиты

Пример: Банковская сфера

- Сверхнадежная база данных
- Очень строгие SLA
- Страхование на случай отказов
- Географически распределенная система ДЦ
- Повышенные требования к безопасности систем хранения и обработки данных

Монолиты

Пример: Биллинг в телекоме

- Обработка в реальном времени
- Огромный поток CDR
- Большое количество бизнес-логики
- Разный биллинг в зависимости от локации

Монолиты

Пример: Ecommerce

- Специфический формат и структура данных
- Only real time
- Множество интеграций с внешними системами
    - Соцсети (авторизация и т.д.)
    - Платежные системы
    - Логистические системы

Проблемы монолитов

- Масштабируемость
- Связанность логики
- Высокая нагрузка
- Отсутствие аналитических инструментов
- Низкий t2m релизов и высокая зависимость компонент системы
- Жесткая структура база данных

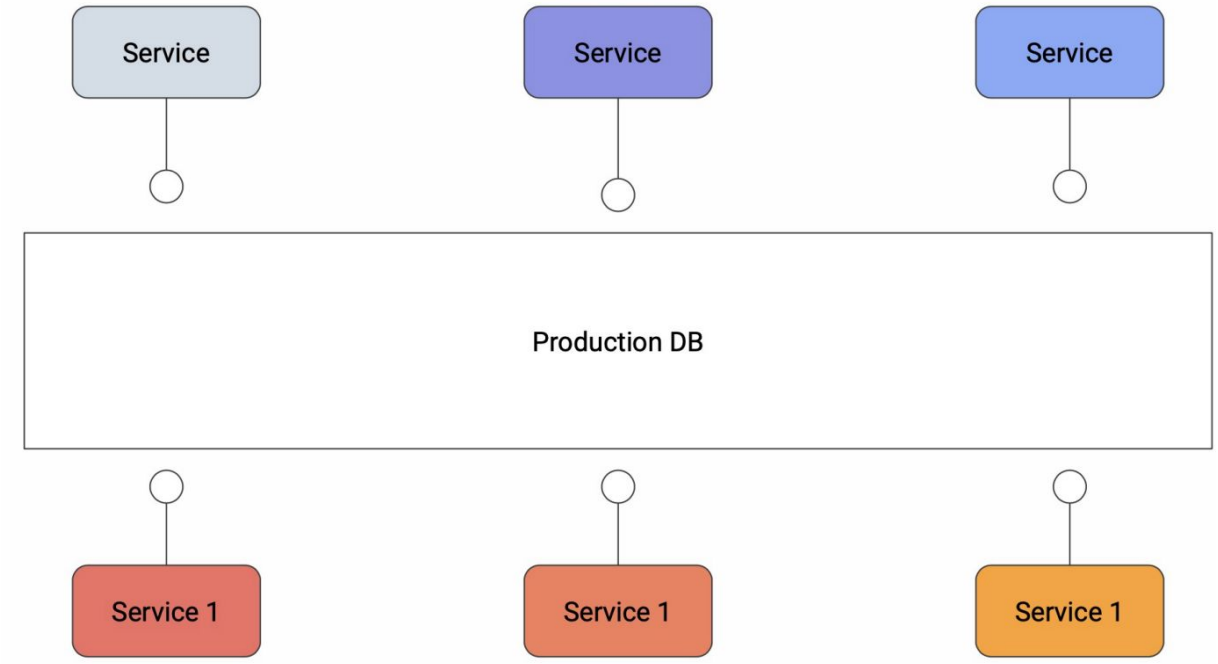
Микросервисы

Микросервисный подход:

- Много слабо связанных баз данных
- Эксперименты с key-value
- Ориентированы на бизнес-задачи (фокус на t2m релиза фичей)
- Разделение логики
- Разделение CI/CD
- Integration overvhead

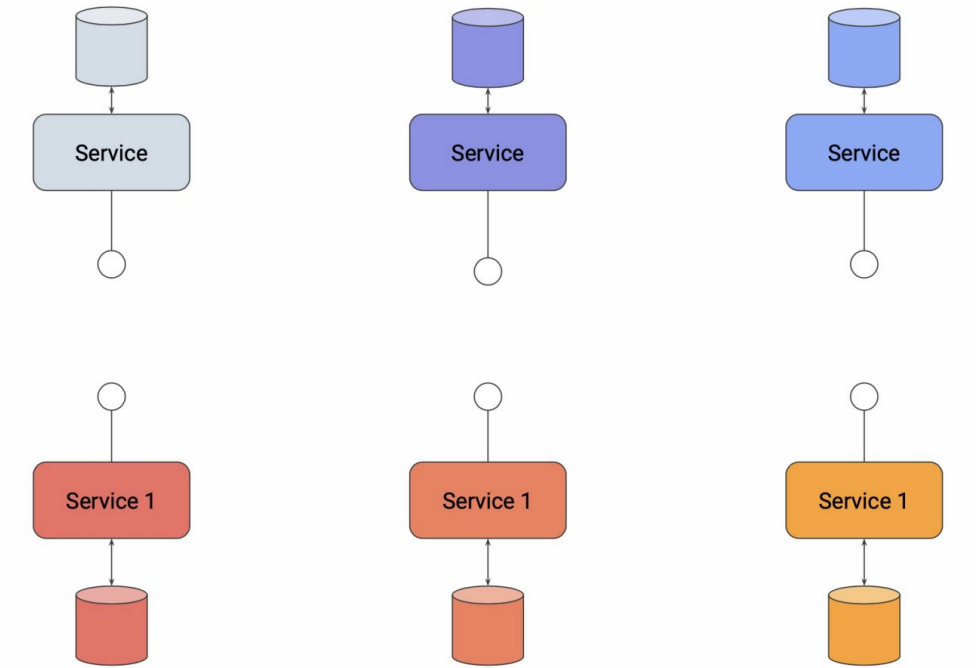
Микросервисы

Шаг 1 - перенос бизнес-логики в отдельные приложения



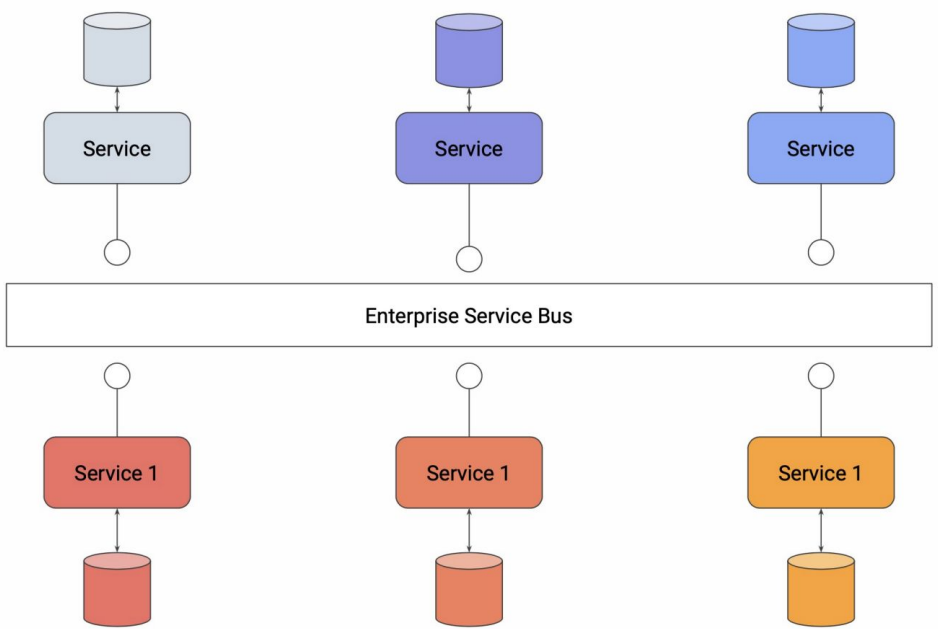
Микросервисы

Шаг 2 - перенос данных в отдельные базы данных



Микросервисы

Шаг 3 - добавляем шину данных



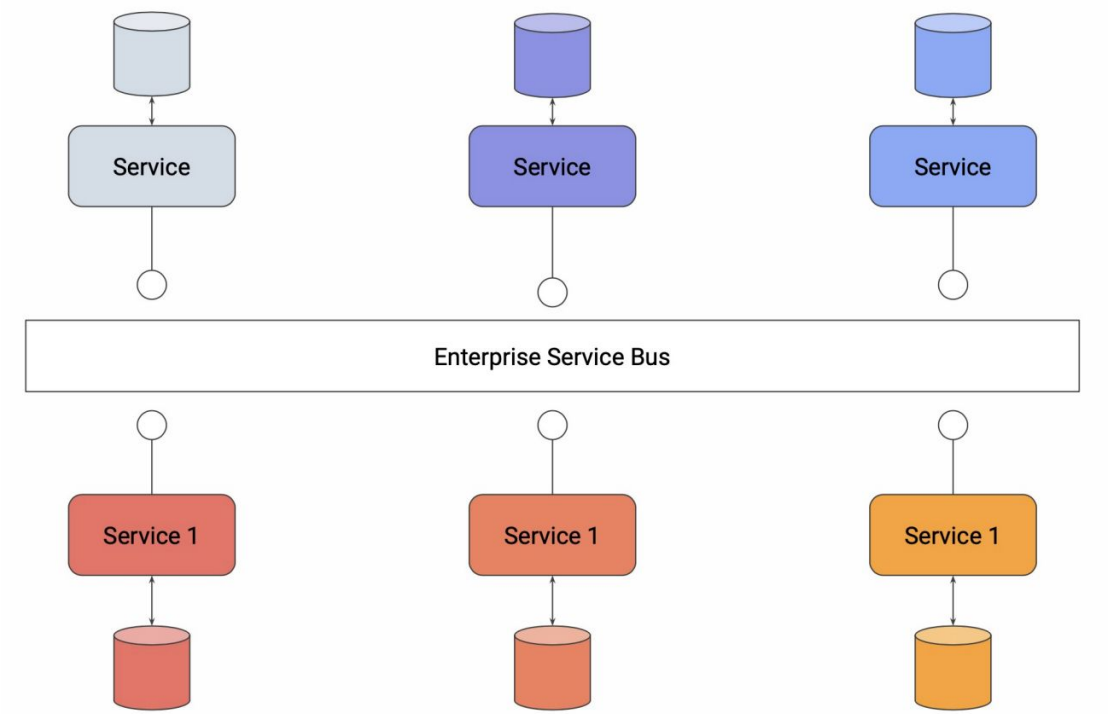
Data Lake

Потребность в хранении больших данных

- real-time
- любые данные в компании
- связанность
- масштабируемость
- скорость вычислений
- высокая аналитическая нагрузка

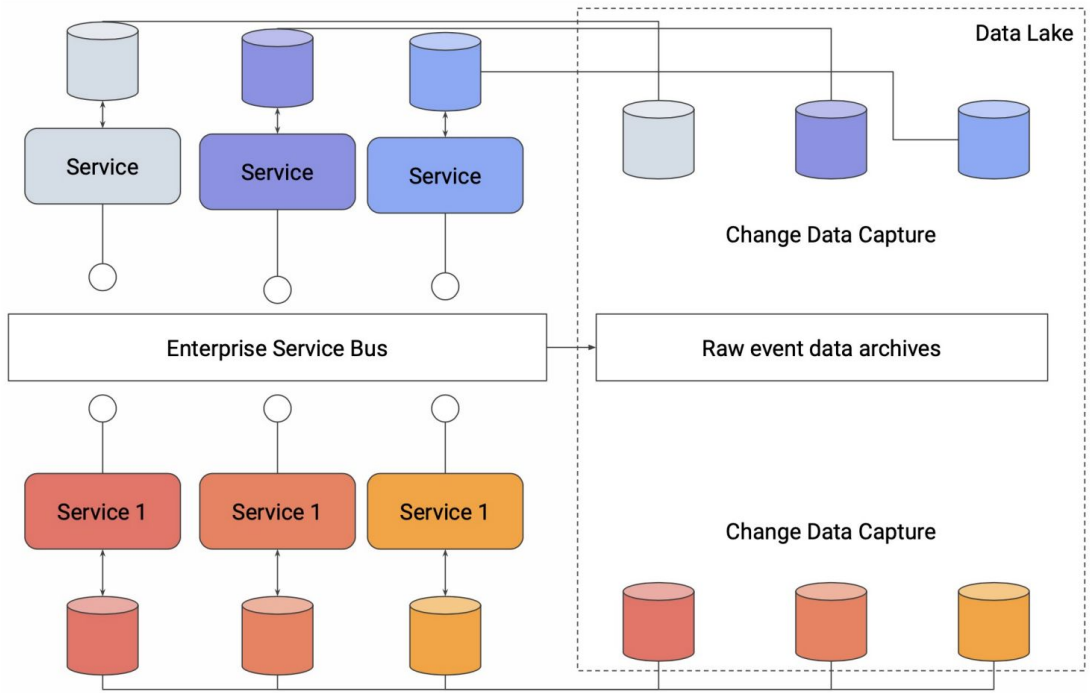
Data Lake

Проблема - данные разрознены по нескольким БД



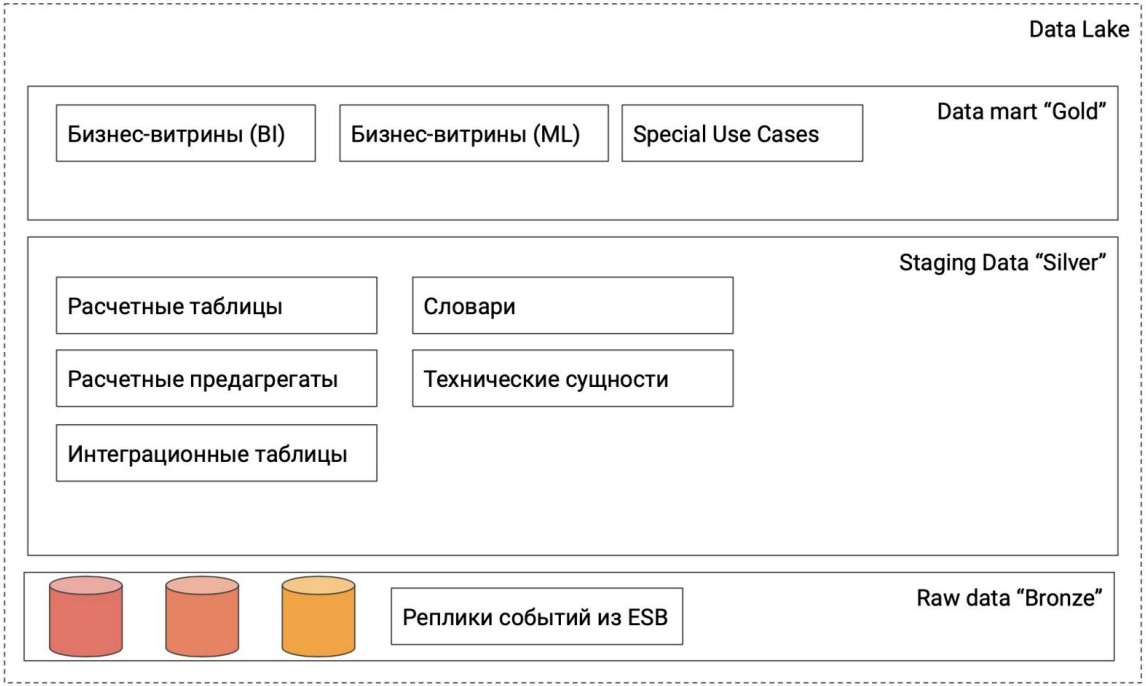
Data Lake

Необходимо надежное и масштабируемое решение для их хранения



Data Lake

Data Lake – это не просто архив данных. Это инфраструктурная компонента.



Форматы хранилищ данных

Реляционные базы данных

Реляционные БД

- Предполагается ACID
  - Atomic
  - Consistent
  - Isolation
  - Durable
- SQL-compliant
- JDBC/ODBC-compliant
- Ключевые возможности
  - Insert/upsert/delete
  - Партицирование
  - Шардинг
  - Индексы

Реляционные базы данных

ACID

- Atomicity (Атомарность)
  - Транзакции могут состоять из нескольких операций. Атомарность гарантирует, что транзакция либо выполняется полностью, либо не выполняется вообще
- Consistency (Консистентность)
  - Любая запись в базу сохраняет ее нормальное состояние (корректно работают связанные объекты)
- Isolation (Изоляция)
  - Гарантирует, что параллельные транзакции выполняются так же, как если бы они выполнялись последовательно
- Durability (Надежность)
  - Если транзакция подтверджена, ее изменения навсегда останутся в базе (даже если сервис выключится)

Реляционные базы данных

Проблемы при параллельных транзакциях

- Потерянное обновление
  - Блок данных изменяется одновременно двумя транзакциями, применяется только последнее обновление
- Грязное чтение
  - Чтение данных, изменение которых не подтвердится
- Неповторяющееся чтение
  - Данные изменились во время чтения в одной транзакции
- Фантомное чтение
  - При повторном чтении внутри одной транзакции количество строк изменилось

Реляционные базы данных

Summary

- это **“не плохо”**
- это не **“не модно”**

Это:

- Хорошо развитая экосистема
- Понятное поведение БД
- Доступ из любого языка программирования

Всегда стоит подумать дважды, выбирая БД под высоконагруженный проект.

Вполне возможно, что обычная реляционная база данных способна справится с нагрузкой на вашем проекте.

Key value stores

Key value stores

- Не предполагается ACID
- В большинстве случаев не SQL-compliant
- JDBC/ODBC-compliant
- Ключевые возможности
  - Индексы
  - Insert/upsert/delete by key
  - Партицирование
  - Шардинг

Key value stores

Ключевые идеи

- Данные хранятся в виде коллекций из ключей и значений
- Схема данных **легко расширяется**
- Основная задача – параллельные, **сверхбыстрые чтение и запись по ключу**

Key value stores

Summary

- да, это “модно”

Это

- Очень быстрая запись и чтение по ключу
- Легко изменяемая структура хранения данных
- Зачастую требует большого количества настроек и DevOps процессов

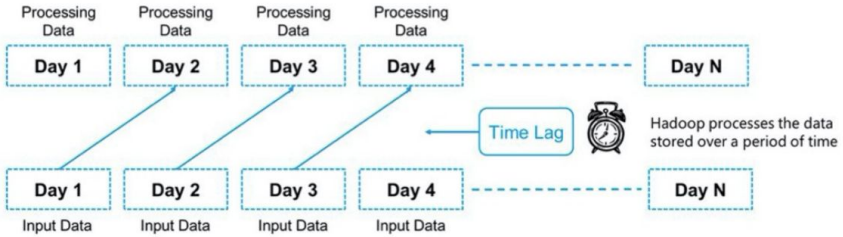
Всегда стоит подумать трижды, выбирая KV под высоконагруженный проект.

Различные KV дают различные возможности, нужно рассматривать каждый отдельный use-case.

Подходы к обработке данных

Batch обработка данных

- Самый распространенный подход
- Идея
  - Читаем данные блоком
  - Изменяем данные блоком
  - Записываем данные блоком
- Большинство ETL/ELT - это Batch



Batch обработка данных

Плюсы

- простой репроцессинг
- высокая эффективность на больших объемах

Минусы

- создает пиковые нагрузки
- медленнее доставляет данные

Batch обработка данных

Инструменты

- Python + Airflow
- Spark - стандарт в этой области
- MapReduce - заветы древних
- Hive - тот же SQL

Stream обработка данных

Stream обработка

- обрабатываем данные на лету
- различные процессы обрабатывают различные события
- постоянно работающее streaming приложение
- low overhead на обработку единичных событий
- высокая масштабируемость - одно из ключевых требований
- low-latency

Stream обработка данных

Плюсы

- ровная нагрузка на ресурсы
- отсутствие дублирования кода, сервисов и данных из-за отсутствия batch-layer
- быстрее доставляют данные

Минусы

- сложный репроцессинг
- необходима идемпотентность записи из одного потока в другой

Stream обработка данных

Используемые инструменты

- Spark Streaming (микробатчи)
- Flink (стримминг)
- Storm (стримминг)

Современные архитектуры

Lambda vs Kappa

Проблема

Все чаще системы обработки данных должны выдавать результат в real-time

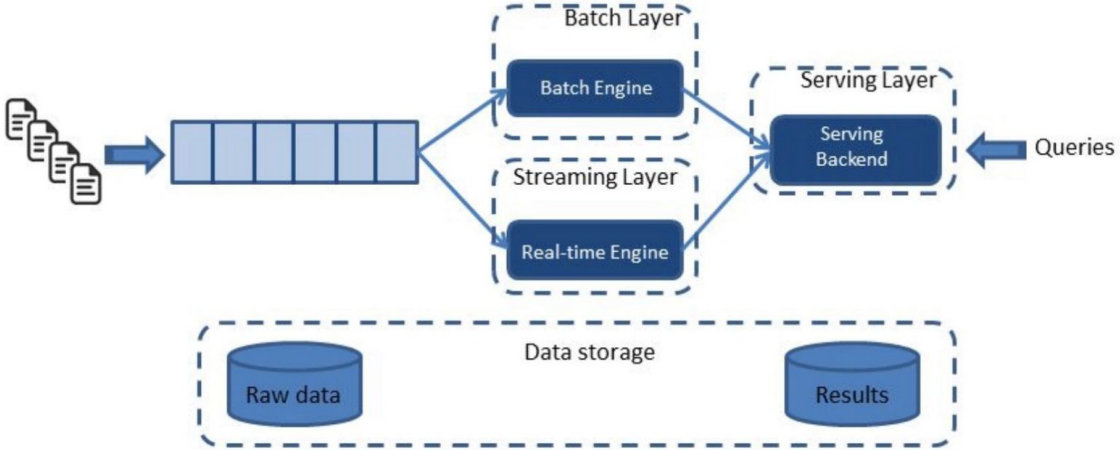
- ужесточились требования к бизнесу - нельзя ждать часы и дни, чтобы узнать значения важных метрик
- аналитика становится инструментом более тонкого мониторинга состояния бизнеса/продукта
- ml-модели принимают решения в реальном времени

Lambda

Первый вариант решения проблемы

- оставляем медленную пакетную обработку для хранения исторических данных и репроцессинга
- добавляем параллельно стриминговую обработку, которая готовит “горячие” данные быстрее пакетной
- объединяем продукты обеих веток в общих витринах данных

Lambda


Batch Layer
Batch Engine
Serving Layer
Serving Backend
Queries
Streaming Layer
Real-time Engine
Data storage
Raw data
Results

Lambda

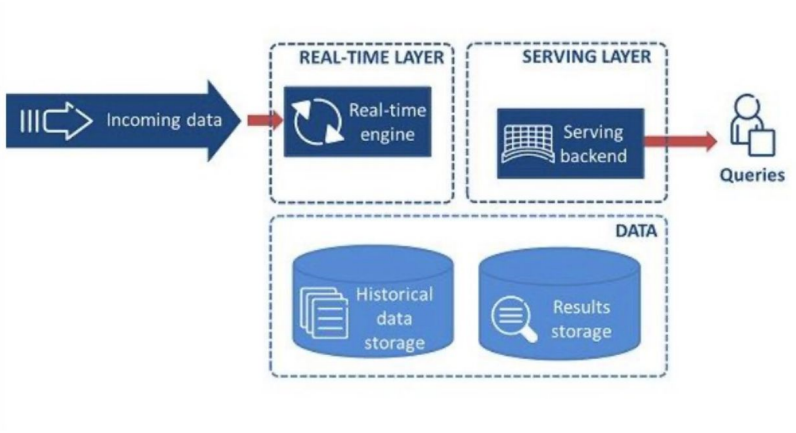
Плюсы

- есть real-time данные
- простой репроцессинг через batch layer
- эффективный доступ к сырым историческим данным

Минусы

- создает двойную нагрузку
- нужно поддерживать две версии кода
- нужно поддерживать два стека

Kappa


Incoming data
REAL-TIME LAYER
Real-time engine
SERVING LAYER
Serving backend
Queries
DATA
Historical data storage
Results storage

Kappa

Плюсы

- универсальный код
- меньше нагрузки на железо
- быстрая доставка данных

Минусы

- тяжелый репроцессинг
- проблемы с хранением сырых данных
- неэффективные форматы хранения промежуточных данных

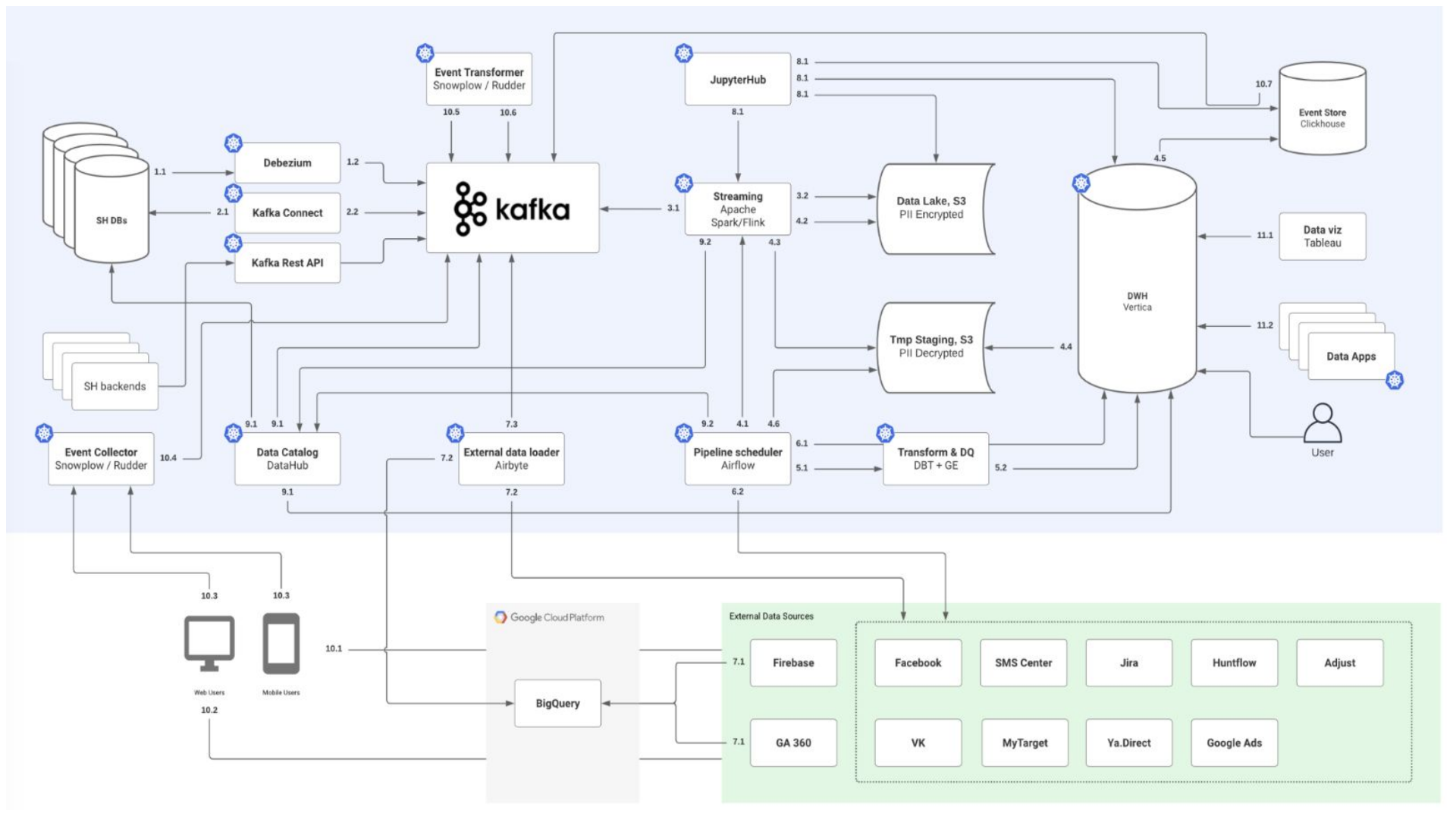
Сравнение

- Lambda более универсальна
- Lambda архитектура требует двух кодовых баз
  - Одна для Batch layer
  - Другая для Realtime layer
- Kappa архитектура предполагает что Batch обработка отсутствует

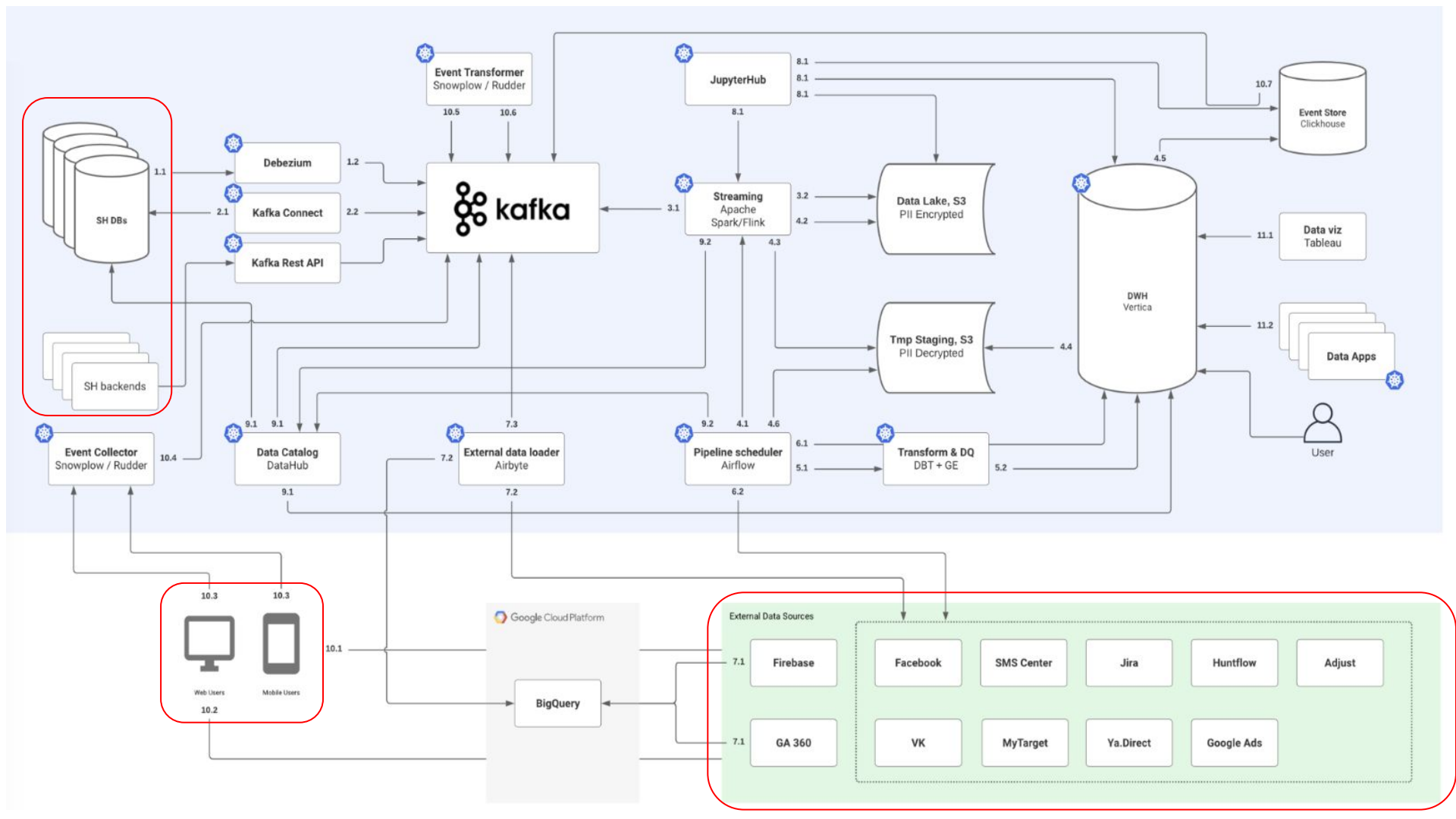
Сравнение

- Lambda более универсальна
- Lambda архитектура требует двух кодовых баз
  - Одна для Batch layer
  - Другая для Realtime layer
- Kappa архитектура предполагает что Batch обработка отсутствует

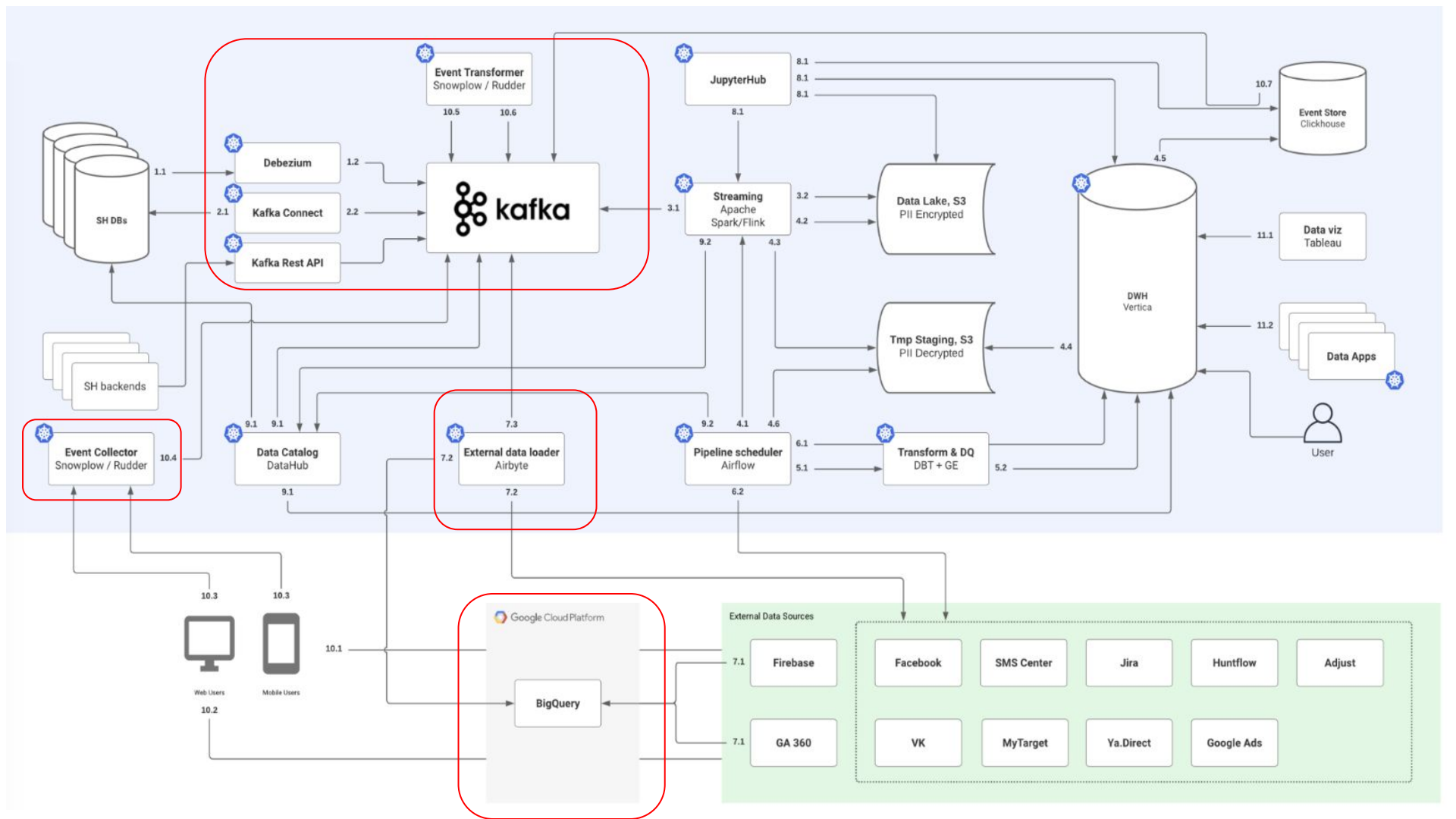
Архитектура современной платформы данных


Event Transformer
Snowplow / Rudder
JupyterHub
Event Store
Clickhouse
SH DBs
Debezium
Kafka Connect
Kafka Rest API
kafka
Streaming
Apache
Spark/Flink
Data Lake, S3
PII Encrypted
DWH
Vertica
Data viz
Tableau
Tmp Staging, S3
PII Decrypted
Data Apps
SH backends
Event Collector
Snowplow / Rudder
Data Catalog
DataHub
External data loader
Airbyte
Pipeline scheduler
Airflow
Transform & DQ
DBT + GE
User
Web Users
Mobile Users
Google Cloud Platform
BigQuery
External Data Sources
Firebase
Facebook
SMS Center
Jira
Huntflow
Adjust
GA 360
VK
MyTarget
Ya.Direct
Google Ads

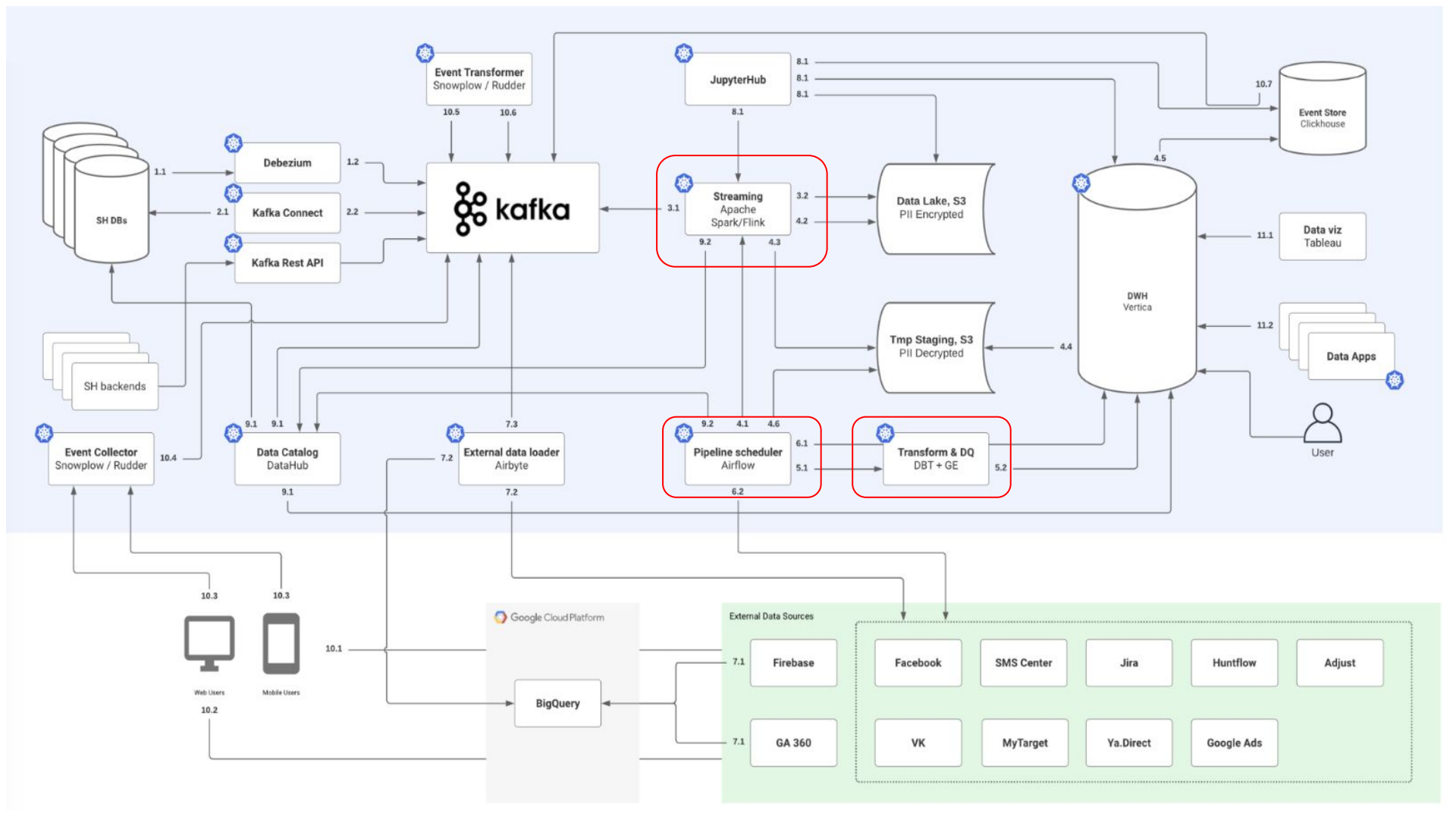
Источники данных


SH DBs
SH backends
Event Transformer
Snowplow / Rudder
Debezium
Kafka Connect
Kafka Rest API
kafka
JupyterHub
Streaming
Apache
Spark/Flink
Data Lake, S3
PII Encrypted
Tmp Staging, S3
PII Decrypted
Event Store
Clickhouse
DWH
Vertica
Data viz
Tableau
Data Apps
User
Event Collector
Snowplow / Rudder
Data Catalog
DataHub
External data loader
Airbyte
Pipeline scheduler
Airflow
Transform & DQ
DBT + GE
Web Users
Mobile Users
Google Cloud Platform
BigQuery
External Data Sources
Firebase
Facebook
SMS Center
Jira
Huntflow
Adjust
GA 360
VK
MyTarget
Ya.Direct
Google Ads

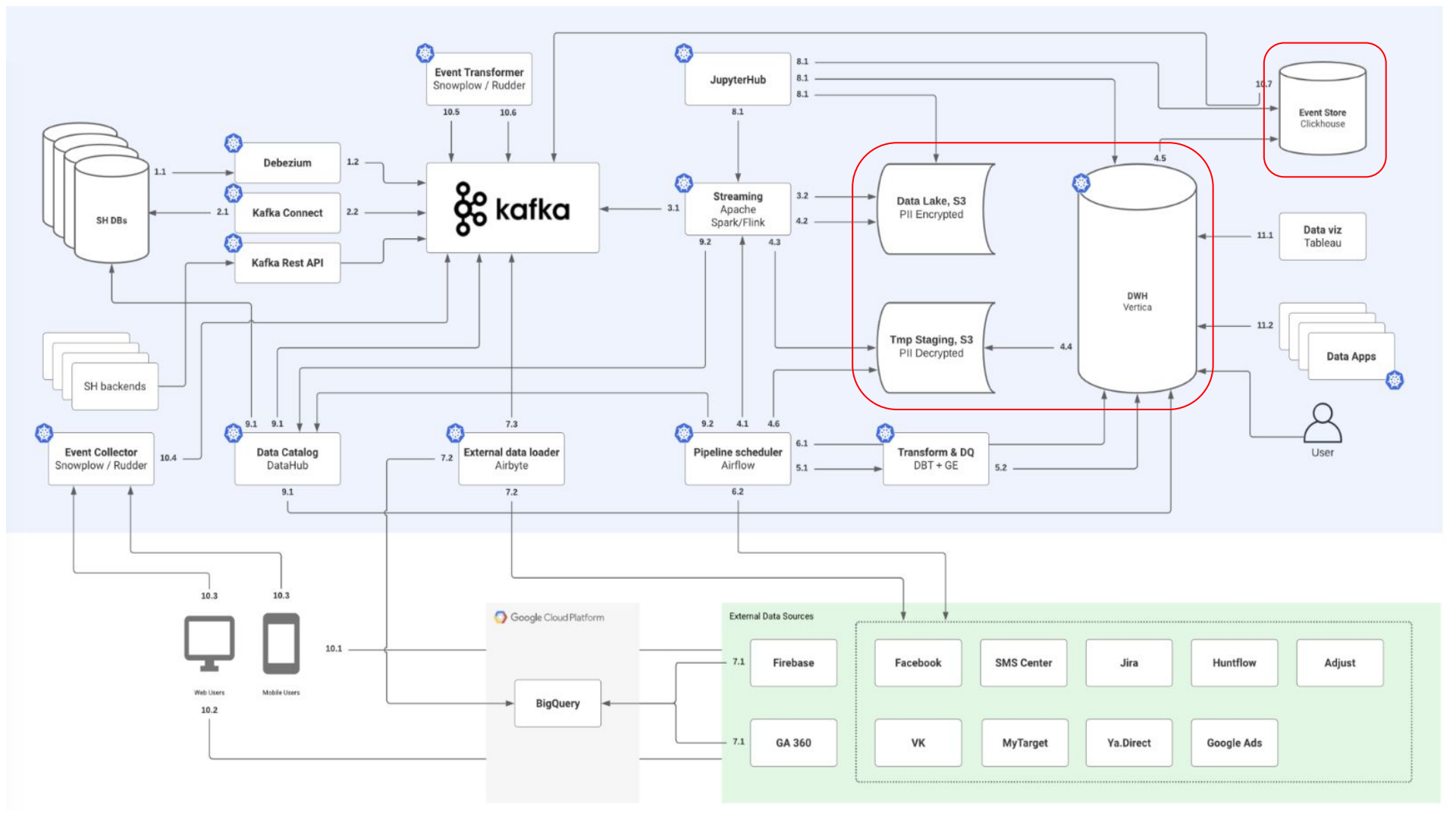
Экстракторы данных


Event Transformer
Snowplow / Rudder
JupyterHub
Event Store
Clickhouse
SH DBs
Debezium
Kafka Connect
Kafka Rest API
kafka
Streaming
Apache
Spark/Flink
Data Lake, S3
PII Encrypted
DWH
Vertica
Data viz
Tableau
Tmp Staging, S3
PII Decrypted
Data Apps
SH backends
User
Event Collector
Snowplow / Rudder
Data Catalog
DataHub
External data loader
Airbyte
Pipeline scheduler
Airflow
Transform & DQ
DBT + GE
Web Users
Mobile Users
Google Cloud Platform
BigQuery
External Data Sources
Firebase
Facebook
SMS Center
Jira
Huntflow
Adjust
GA 360
VK
MyTarget
Ya.Direct
Google Ads

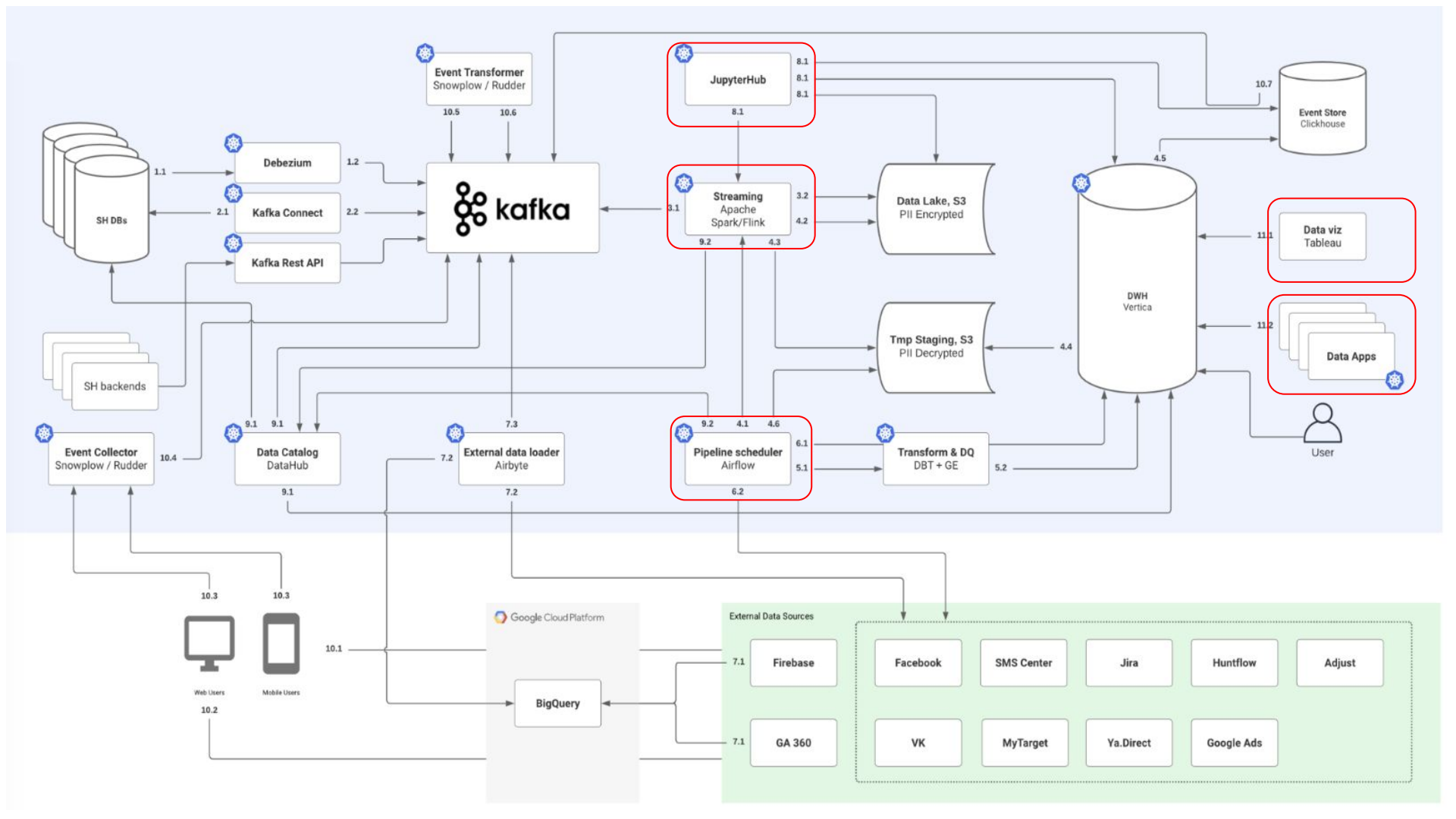
Движки обработки данных


Event Transformer
Snowplow / Rudder
JupyterHub
Event Store
Clickhouse
SH DBs
Debezium
Kafka Connect
Kafka Rest API
kafka
Streaming
Apache
Spark/Flink
Data Lake, S3
PII Encrypted
DWH
Vertica
Data viz
Tableau
Tmp Staging, S3
PII Decrypted
Data Apps
SH backends
Event Collector
Snowplow / Rudder
Data Catalog
DataHub
External data loader
Airbyte
Pipeline scheduler
Airflow
Transform & DQ
DBT + GE
User
Web Users
Mobile Users
Google Cloud Platform
BigQuery
External Data Sources
Firebase
Facebook
SMS Center
Jira
Huntflow
Adjust
GA 360
VK
MyTarget
Ya.Direct
Google Ads

Система хранения данных


Event Transformer
Snowplow / Rudder
JupyterHub
Event Store
Clickhouse
SH DBs
Debezium
Kafka Connect
Kafka Rest API
kafka
Streaming
Apache
Spark/Flink
Data Lake, S3
PII Encrypted
DWH
Vertica
Data viz
Tableau
Tmp Staging, S3
PII Decrypted
Data Apps
SH backends
User
Event Collector
Snowplow / Rudder
Data Catalog
DataHub
External data loader
Airbyte
Pipeline scheduler
Airflow
Transform & DQ
DBT + GE
Web Users
Mobile Users
Google Cloud Platform
BigQuery
External Data Sources
Firebase
Facebook
SMS Center
Jira
Huntflow
Adjust
GA 360
VK
MyTarget
Ya.Direct
Google Ads

Сервисы для пользователей


Event Transformer
Snowplow / Rudder
JupyterHub
Event Store
Clickhouse
SH DBs
Debezium
Kafka Connect
Kafka Rest API
kafka
Streaming
Apache
Spark/Flink
Data Lake, S3
PII Encrypted
DWH
Vertica
Data viz
Tableau
Data Apps
Tmp Staging, S3
PII Decrypted
SH backends
Event Collector
Snowplow / Rudder
Data Catalog
DataHub
External data loader
Airbyte
Pipeline scheduler
Airflow
Transform & DQ
DBT + GE
User
Web Users
Mobile Users
Google Cloud Platform
BigQuery
External Data Sources
Firebase
Facebook
SMS Center
Jira
Huntflow
Adjust
GA 360
VK
MyTarget
Ya.Direct
Google Ads

Платформа данных как продукт

Платформа данных как продукт (идея)

Domain maturity index **72%**

Индекс использования данных **59%**

- Учет инструментов (BI системы, Grafana)
- Разный вес для разных ролей, учет количества дашбордов
- Разный вес для дашбордов (сертификация)

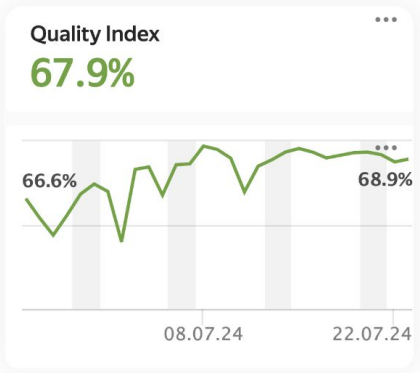
Индекс качества данных **60%**

- Количество использования ключевых витрин в дашбордах
- Качество расчета ключевых витрин и их источников
- Процент покрытия DQ для CDM/REP слоя домена
- Доля багов в платформе данных / количество событий в событийных данных

Индекс стабильности данных **95%**

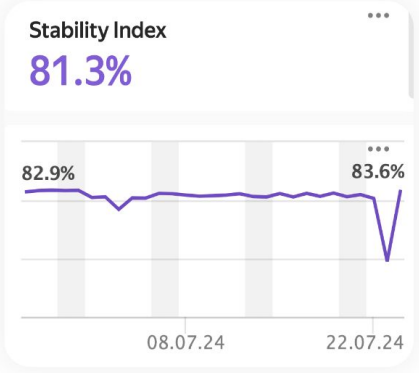
- Стабильность ключевых витрин и их расчета
- SLA обновления витрин 10 утра
- Разный вес витрин в зависимости от их важности для бизнеса (70% vs 30% события)

Платформа данных как продукт (реализация)



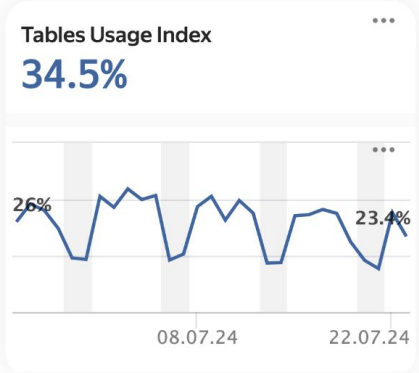
Индекс качества

DQ Coverage x0.4 +
Metric Run DQ OK x0.6



Индекс стабильности

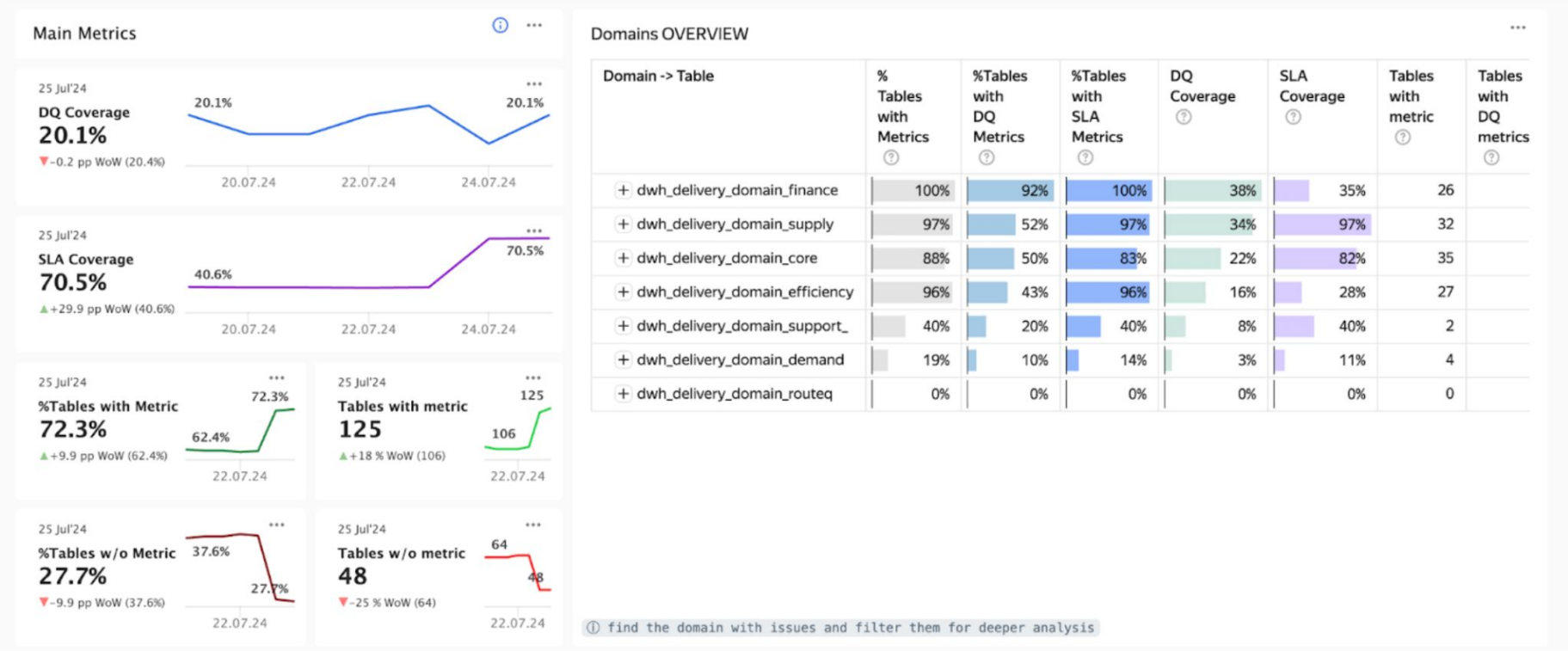
SLA Coverage x0.25 +
Metric Run SLA OK x0.4 +
Task Run Success x0.35



Индекс использования данных

Delivery DWH Usage x0.6 +
Users Used Delivery DWH >5% x0.3 +
Delivery vs Other DWH Usage x0.1

Можно видеть уровень покрытие DQ для каждого домена в деталях



Следим за качеством данных и инцидентами

Task Health Calendar

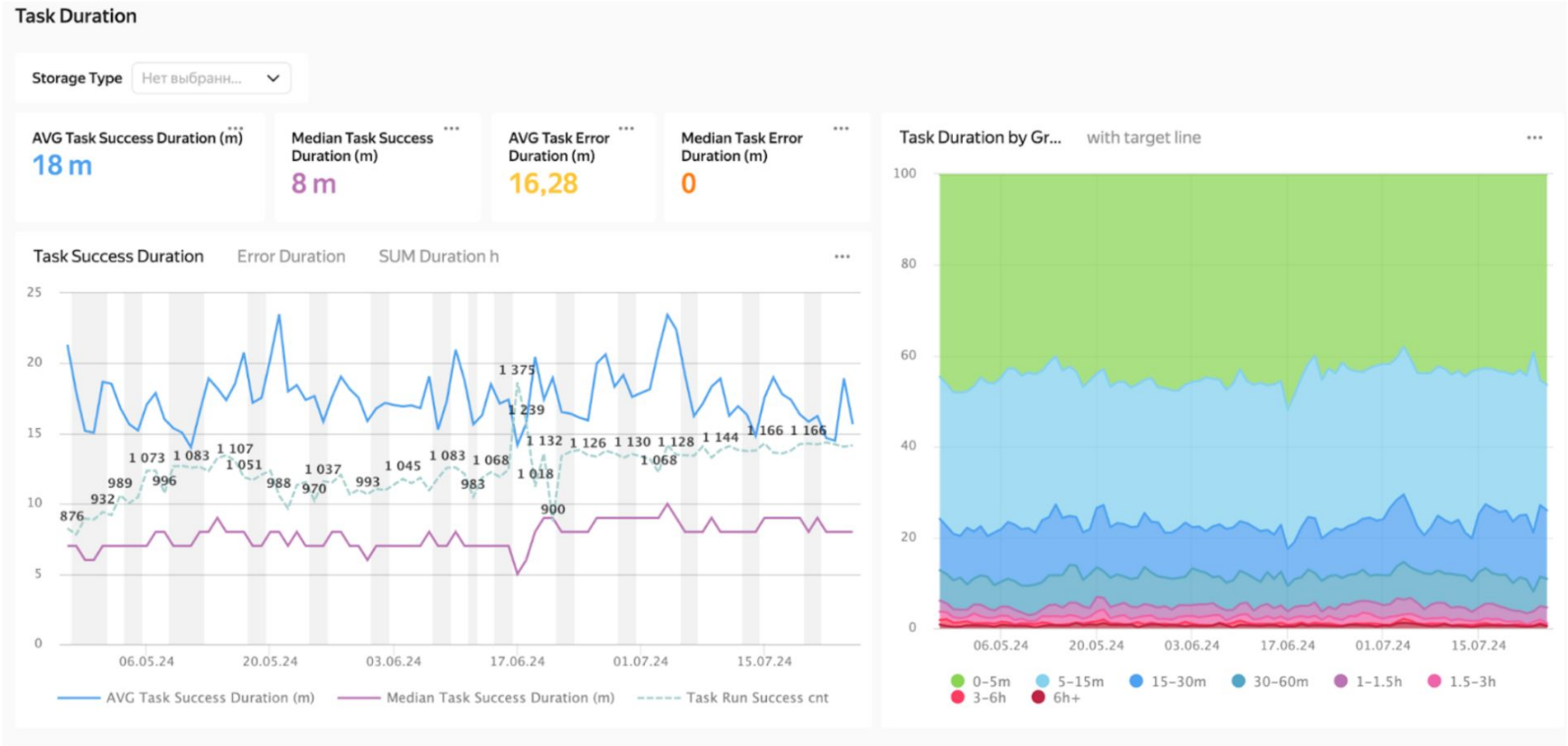
In calendar we show number of TasksRun with ERROR
*intensity of color depends on %Error TaskRun from total

Task Calendar Health Calendar Flag

Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
06.05.2024	11	12	3	0	0	515	118
13.05.2024	721	720	351	7	3	0	30
20.05.2024	0	9	20	19	6	1	29
27.05.2024	3	20	7	40	6	6	6
03.06.2024	37	10	7	2	0	1	1
10.06.2024	2	9	0	27	1	0	0
17.06.2024	102	48	9	27	29	76	0
24.06.2024	9	133	2	2	1	0	0
01.07.2024	56	38	46	16	25	1	2
08.07.2024	9	13	2	25	2	0	0
15.07.2024	0	5	0	0	0	0	9
22.07.2024	11	580	16	23			

ⓘ pay attention to not green days

Следим за трендом времени исполнения наших пайплайнов обработки данных в DWH



Следим за тем, какие данные используют аналитики из доменов