СБЕР ЗДОРОВЬЕ

Октябрь

Ingest-слой платформы данных: смешать, но не взбалтывать

Олег Кочергин

2022

СБЕР ЗДОРОВЬЕ

Олег Кочергин

Head of Data & Analytics @ SberHealth,
ex. Alfa-Bank

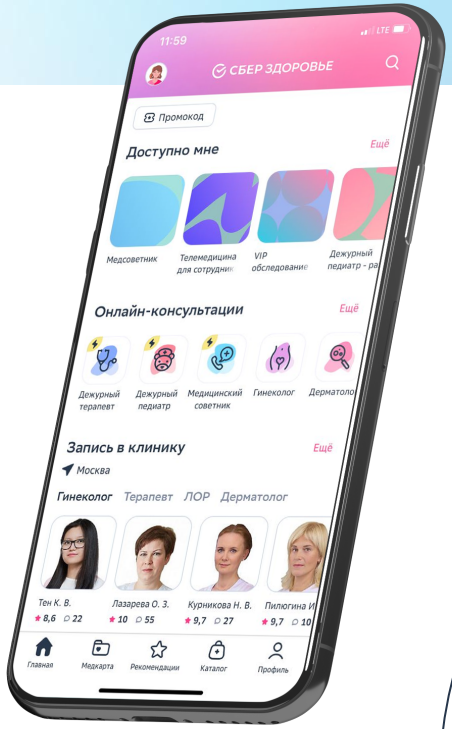
20+ лет в разработке, последние 7 в областях работы с данными и аналитикой

О чем поговорим

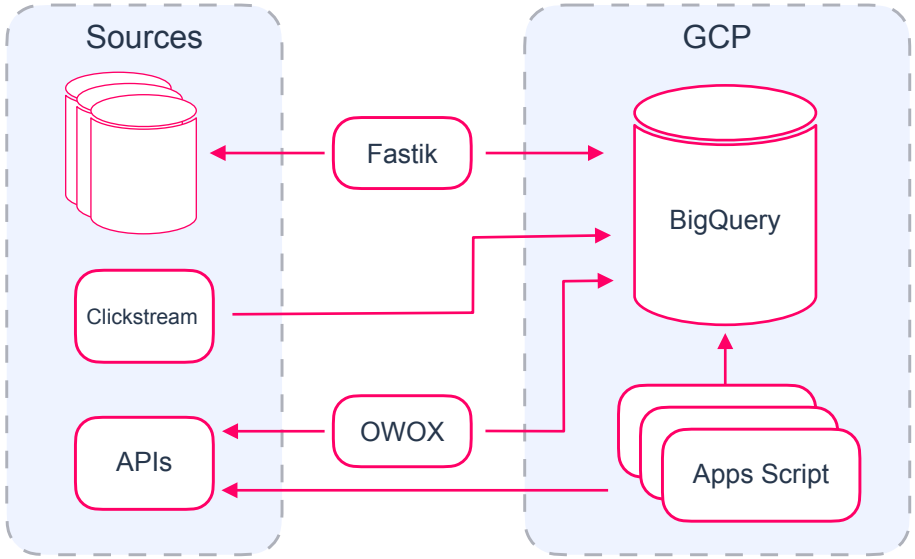
- Кратко про СберЗдоровье
- `NOW() - INTERVAL '1 YEAR'`
- Про архитектуру в общем и ingest слой частности
- Про ingest-as-code

Пара слов про СберЗдоровье

- Основаны в 2012 году
- Изначально звались DocDoc, продукт — запись в клиники
- В 2018 году → Телемедицина
- В 2020 → Мониторинг хронических больных
- В 2022 → Подключенные мед девайсы



Оригинальная архитектура и проблемки



- Google cloud
- Home grown ETL tool
  - Batch only
  - No historization
  - No support
- Персональные/чувствительные данные
  - Вообще не могли с ними работать, а очень надо иногда
- Хаос интеграции API источников

Мы всё переделали

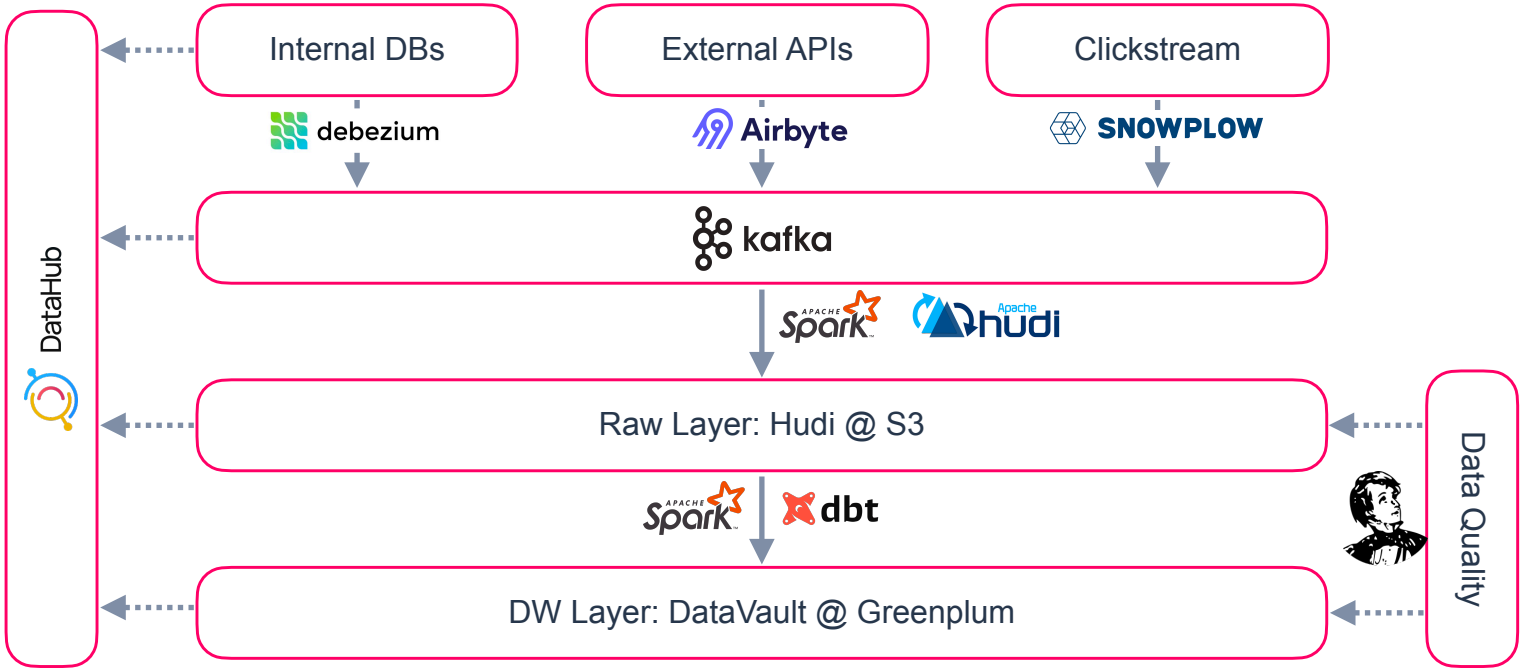
Что хотели

СБЕР ЗДОРОВЬЕ

- Переезд в РФ облако
- Минимум велосипедов
- Streaming first
- Историзация
- Respect-the-privacy
- Чем проще в использовании → тем лучше

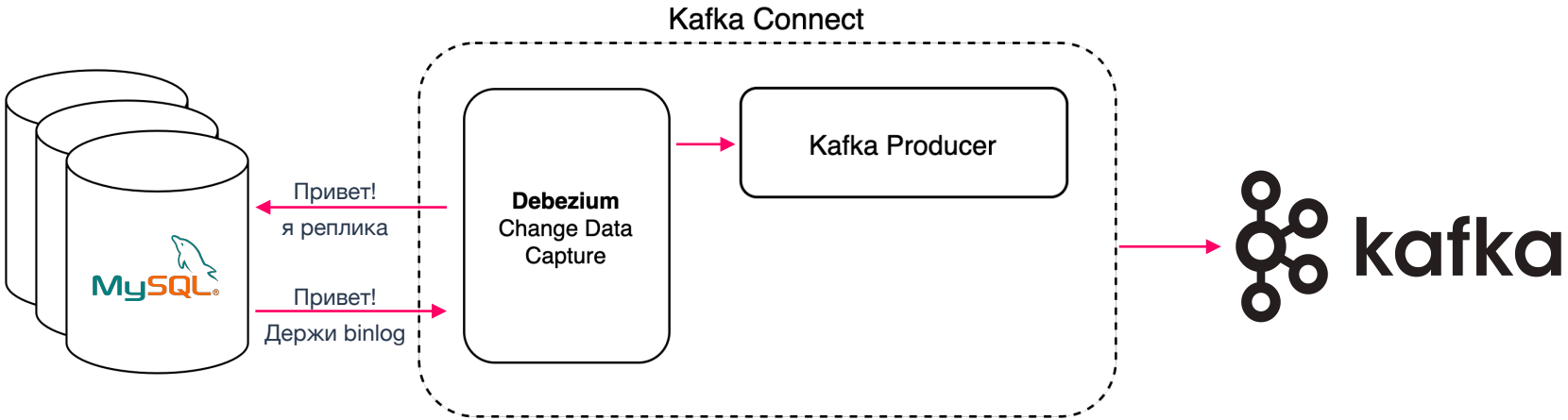
Что получилось

СБЕР ЗДОРОВЬЕ



Debezium

СБЕР ЗДОРОВЬЕ



Внешние API

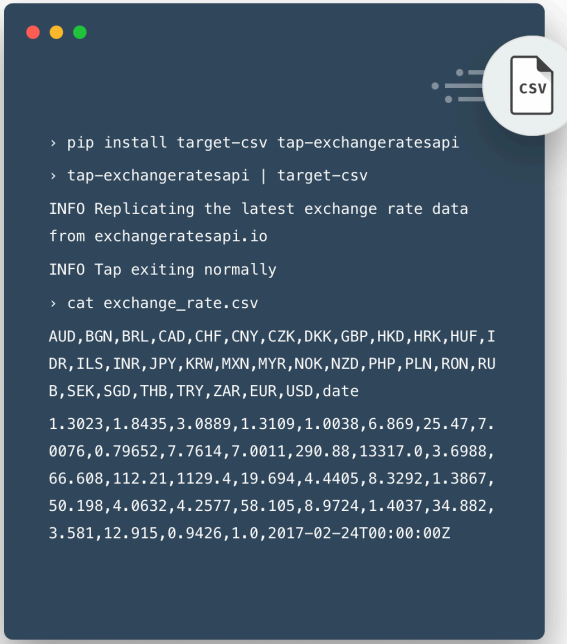
СБЕР ЗДОРОВЬЕ

- Можно нафигачить кучу велосипедов для взаимодействия с API и упаковать это в airflow
- Можно взять готовые интеграции singer.io, недостающие закодить и упаковать в airflow
- Можно взять Airbyte, недостающие интеграции закодить

Внешние API: singer.io + Airflow

СБЕР ЗДОРОВЬЕ

- JSON протокол stdout → stdin
- Много готовых коннекторов
- Конфиги никак не валидируются
- API отсутствует
- Надо как-то костылить работу с инкрементами
- Внутри airflow запуск через BashOperator

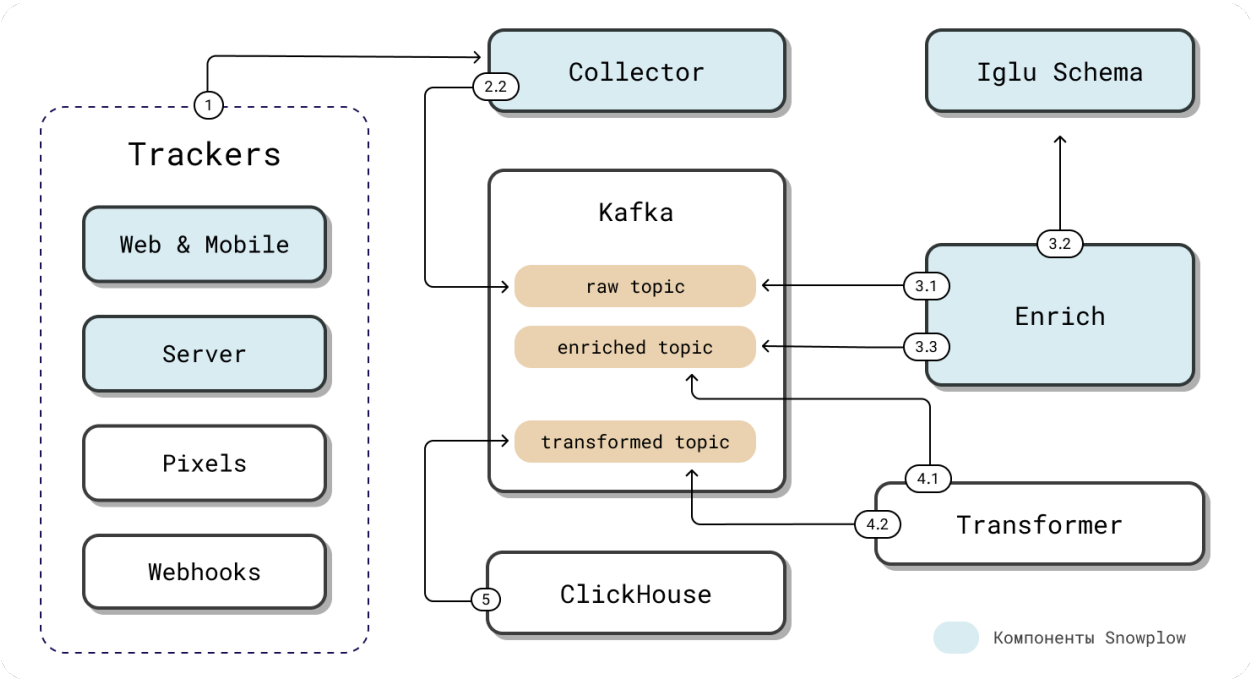


Внешние API: Airbyte

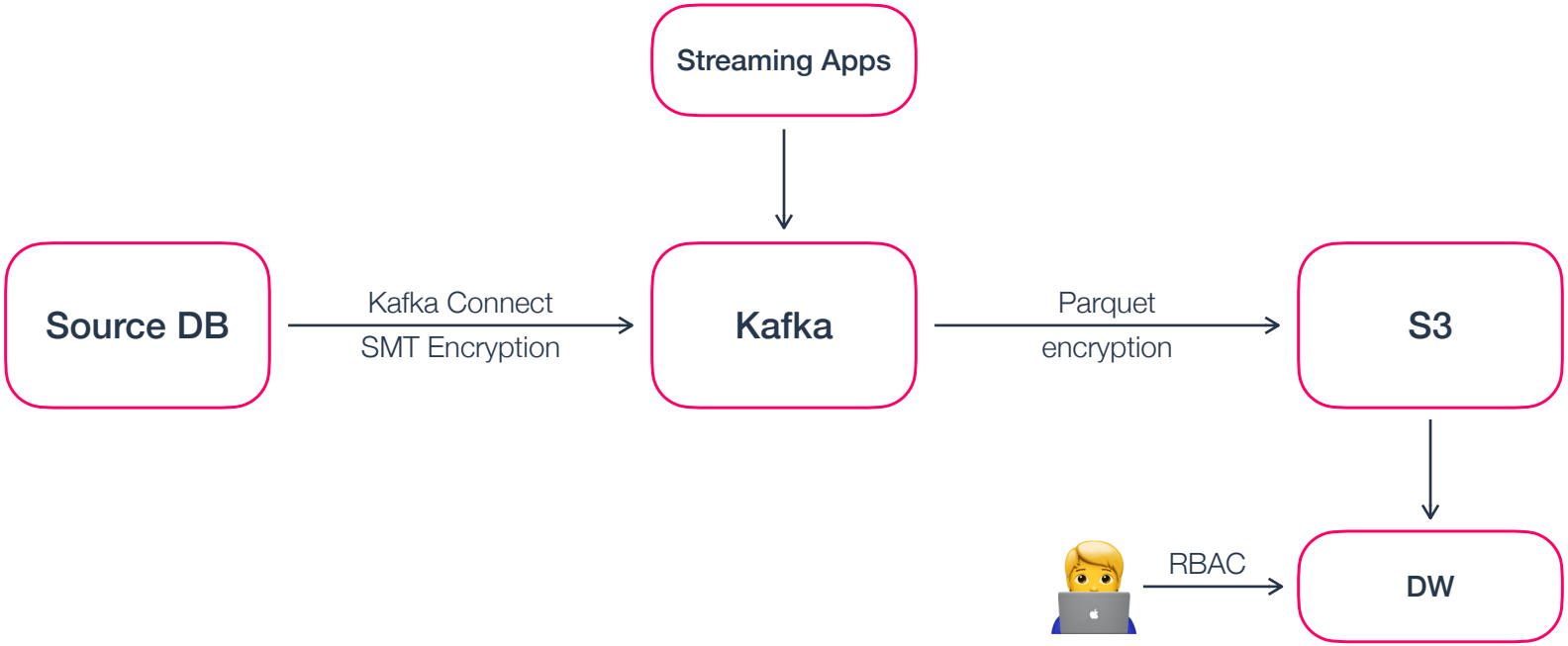
СБЕР ЗДОРОВЬЕ

- Open Core проект
- singer.io + Docker + Scheduler + API + UI
- Все коннекторы singer и даже больше, недостающие дописали под себя
- Конфиги строго типизированы
- Не надо велосипедить работу с инкрементами, всё уже готово

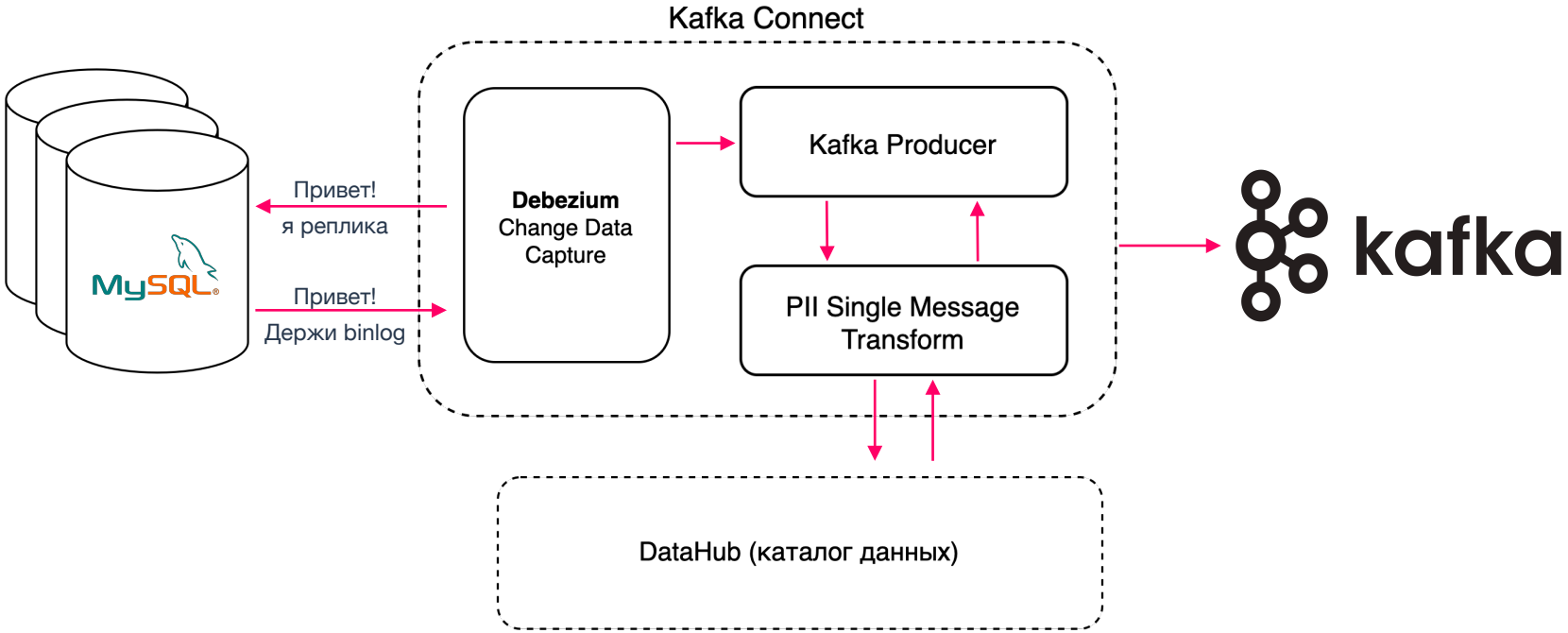
Кликстрим: Snowplow



Respect-the-privacy


Streaming Apps
Source DB
Kafka Connect
SMT Encryption
Kafka
Parquet
encryption
S3
RBAC
DW

Respect-the-privacy (Source DB → Kafka)

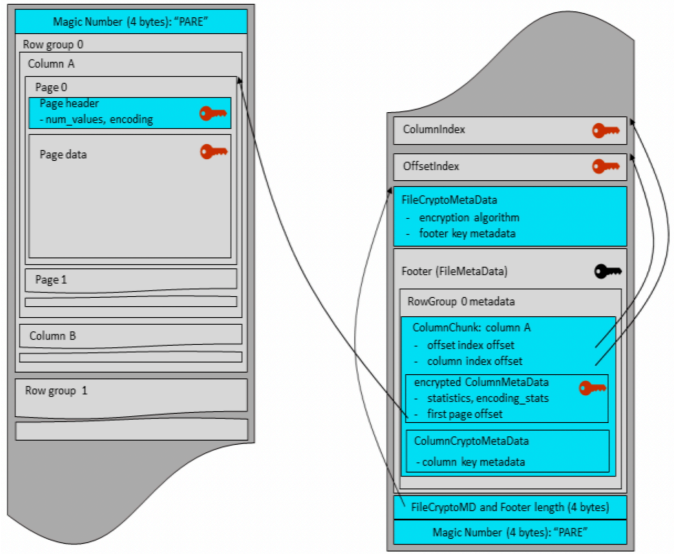


https://kafka.apache.org/documentation/#connect_transforms
https://github.com/aiven/transforms-for-apache-kafka-connect

Respect-the-privacy: Parquet Encryption

СБЕР ЗДОРОВЬЕ

- Encryption + Integrity
- Encryption
  - Full encryption
  - Separate keys for columns
  - Separate keys for footer / metadata
  - Plaintext footer для обратной совместимости



https://github.com/apache/parquet-format/blob/master/Encryption.md
https://www.youtube.com/watch?v=xfGkCqDKtwg

Respect-the-privacy: Hudi + Spark

СБЕР ЗДОРОВЬЕ

```
// Activate Parquet encryption, driven by Hadoop properties
jsc.hadoopConfiguration().set("parquet.crypto.factory.class", "org.apache.parquet.crypto.keytools.Prope
// Explicit master keys (base64 encoded) - required only for mock InMemoryKMS
jsc.hadoopConfiguration().set("parquet.encryption.kms.client.class" , "org.apache.parquet.crypto.keytoo
jsc.hadoopConfiguration().set("parquet.encryption.key.list", "k1:AAECAwQFBgcICQoLDA0ODw==, k2:AAECAAECA
// Write encrypted dataframe files.
// Column "rider" will be protected with master key "key2".
// Parquet file footers will be protected with master key "key1"
jsc.hadoopConfiguration().set("parquet.encryption.footer.key", "k1")
jsc.hadoopConfiguration().set("parquet.encryption.column.keys", "k2:rider")
```

Copy

https://hudi.apache.org/docs/encryption/

Сконфигурируй и поддерживай это

Config: Debezium

СБЕР ЗДОРОВЬЕ

```
$ curl -i -X POST -H "Content-Type:application/json" localhost:8083/connectors/ -d
{
  "name": "inventory-connector",
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "tasks.max": "1",
    "database.hostname": "mysql",
    "database.port": "3306",
    "database.user": "debezium",
    "database.password": "dbz",
    "database.server.id": "184054",
    "database.server.name": "dbserver1",
    "database.include.list": "inventory",
    "database.history.kafka.bootstrap.servers": "kafka:9092",
    "database.history.kafka.topic": "schema-changes.inventory"
  }
}
```

Config: Debezium

СБЕР ЗДОРОВЬЕ

```
$ curl -i -X POST -H "Content-Type:application/json" localhost:8083/connectors/ -d
{
  "name": "inventory-connector",
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "tasks.max": "1",
    "database.hostname": "mysql",
    "database.port": "3306",
    "database.user": "debezium",
    "database.password": "dbz",
    "database.server.id": "184054",
    "database.server.name": "dbserver1",
    "database.include.list": "inventory",
    "database.history.kafka.bootstrap.servers": "kafka:9092",
    "database.history.kafka.topic": "schema-changes.inventory"
  }
}
```

Config: Debezium

СБЕР ЗДОРОВЬЕ

```
$ curl -i -X POST -H "Content-Type:application/json" localhost:8083/connectors/ -d
{
  "name": "inventory-connector",
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "tasks.max": "1",
    "database.hostname": "mysql",
    "database.port": "3306",
    "database.user": "debezium",
    "database.password": "dbz",
    "database.server.id": "184054",
    "database.server.name": "dbserver1",
    "database.include.list": "inventory",
    "database.history.kafka.bootstrap.servers": "kafka:9092",
    "database.history.kafka.topic": "schema-changes.inventory"
  }
}
```

Config: Debezium

СБЕР ЗДОРОВЬЕ

```
$ curl -i -X POST -H "Content-Type:application/json" localhost:8083/connectors/ -d
{
  "name": "inventory-connector",
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "tasks.max": "1",
    "database.hostname": "mysql",
    "database.port": "3306",
    "database.user": "debezium",
    "database.password": "dbz",
    "database.server.id": "184054",
    "database.server.name": "dbserver1",
    "database.include.list": "inventory",
    "database.history.kafka.bootstrap.servers": "kafka:9092",
    "database.history.kafka.topic": "schema-changes.inventory"
  }
}
```

Config: Debezium snapshot modes

СБЕР ЗДОРОВЬЕ

Snapshot → состояние всех строк в таблице

- `initial` — делаем, но ток в первый раз
- `when_needed` — делаем по мере необходимости
- `schema_only` — только схемы таблиц

Config: Debezium big snapshots

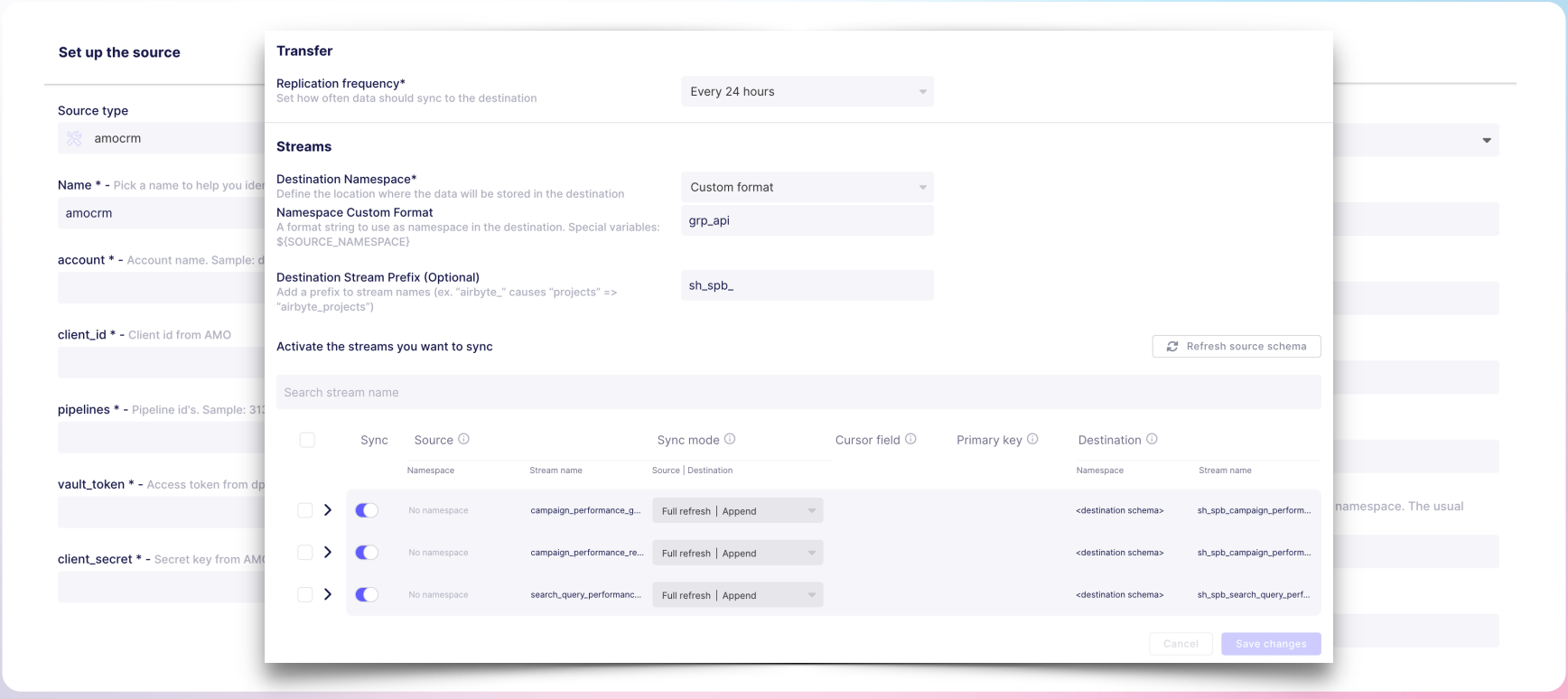
СБЕР ЗДОРОВЬЕ

- `SELECT *` во время первого старта = проблема на больших таблицах
- И нам тут нам поможет incremental snapshot
- Но для его работы нужны т.н. сигнальные таблицы

```
INSERT INTO myschema.debezium_signal (id, type, data) VALUES('ad-hoc-1',
'execute-snapshot', '{"data-collections": ["schema1.table1",
“schema2.table2"],"type":"incremental"}');
```

Config: Airbyte

СБЕР ЗДОРОВЬЕ



СБЕР ЗДОРОВЬЕ

Если просто всё развернуть и больше ничего не делать то для поддержки всего добра надо будет постоянно что-то делать, зная все нюансы, разумеется :)

DataHub → руками добавлять источники

Debezium → руками через api запускать/перезапускать коннекторы

Airbyte → руками через UI/API настраивать коннекторы и расписания

При этом, конечно, никакого ревью изменений и никакой версионности

А что если закрыть это всё за абстракцией

И так мы изобрели свой terraform на минималках

YAML + CLI = ❤️

Создаём источники в дата-каталоге + lineage

Управляем конфигурацией Debezium и Airbyte

Управляем Spark (Kafka → S3) джобами

YAML

СБЕР ЗДОРОВЬЕ

```
rgs_rpm:
  metadata:
    description: some cool and mission critical database
    domain: rgs
    tags: [‘blah’]

  connection:
    engine: postgres
    params:
      hostname: database_host
      port: 5432
      user: data_platform_user
      password: {$.rgs_rpm_password’}

  streams:
    - schema1:
       include_tables:
          - tbl1
          - tbl2
          - tbl3
```

YAML

СБЕР ЗДОРОВЬЕ

```
rgs_rpm:
  metadata:
    description: some cool and mission critical database
    domain: rgs
    tags: [‘blah’]

  connection:
    engine: postgres
    params:
      hostname: database_host
      port: 5432
      user: data_platform_user
      password: {$.rgs_rpm_password’}

  streams:
    - schema1:
       include_tables:
          - tbl1
          - tbl2
          - tbl3
```

YAML

СБЕР ЗДОРОВЬЕ

```
rgs_rpm:
  metadata:
    description: some cool and mission critical database
    domain: rgs
    tags: [‘blah’]

  connection:
    engine: postgres
    params:
      hostname: database_host
      port: 5432
      user: data_platform_user
      password: {$.rgs_rpm_password’}

  streams:
    - schema1:
       include_tables:
          - tbl1
          - tbl2
          - tbl3
```

YAML

СБЕР ЗДОРОВЬЕ

```
some_gitlab_account:
  engine: airbyte
  type: gitlab

  params:
    account_name: some_gitlab_account_name
    access_token: {$.some_gitlab_account_access_token}
    group_id: some_gitlab_group_id
    project_id: some_gitlab_project_id

    sources:
      - branches
      - commits
      - issues
      - pipelines
```

CLI

СБЕР ЗДОРОВЬЕ

```
$ ingestcli --help
Usage: ingestcli [command]

[command] is one of the following:
   validate          Checks if the configuration file is correct
   prepare           Prepare (if needed) cloud environment
   apply debezium    Apply configuration to Airbyte
   apply airbyte     Apply configuration to Debezium
   apply spark       Apply configuration to Spark
```

CLI

СБЕР ЗДОРОВЬЕ

```
$ ingestcli --help
Usage: ingestcli [command]

[command] is one of the following:
   validate          Checks if the configuration file is correct
   prepare           Prepare (if needed) cloud environment
   apply debezium    Apply configuration to Airbyte
   apply airbyte     Apply configuration to Debezium
   apply spark       Apply configuration to Spark
```

CLI

СБЕР ЗДОРОВЬЕ

```
$ ingestcli --help
Usage: ingestcli [command]

[command] is one of the following:
   validate          Checks if the configuration file is correct
   prepare           Prepare (if needed) cloud environment
   apply debezium    Apply configuration to Airbyte
   apply airbyte     Apply configuration to Debezium
   apply spark       Apply configuration to Spark
```

CLI

СБЕР ЗДОРОВЬЕ

```
$ ingestcli --help
Usage: ingestcli [command]

[command] is one of the following:
   validate          Checks if the configuration file is correct
   prepare           Prepare (if needed) cloud environment
   apply debezium    Apply configuration to Airbyte
   apply airbyte     Apply configuration to Debezium
   apply spark       Apply configuration to Spark
```

Итого

СБЕР ЗДОРОВЬЕ

Чего добились

- Из одного конфига управляем почти всем ingest пайплайном
- Полностью абстрагированы от конкретных технологий
- Минимизировали человеческий фактор, новые источники добавлять могут даже аналитики

Планы

- Автоматизировать incremental snapshots
- Реализовать шифрование JSON атрибутов, а не всего поля

Q&A