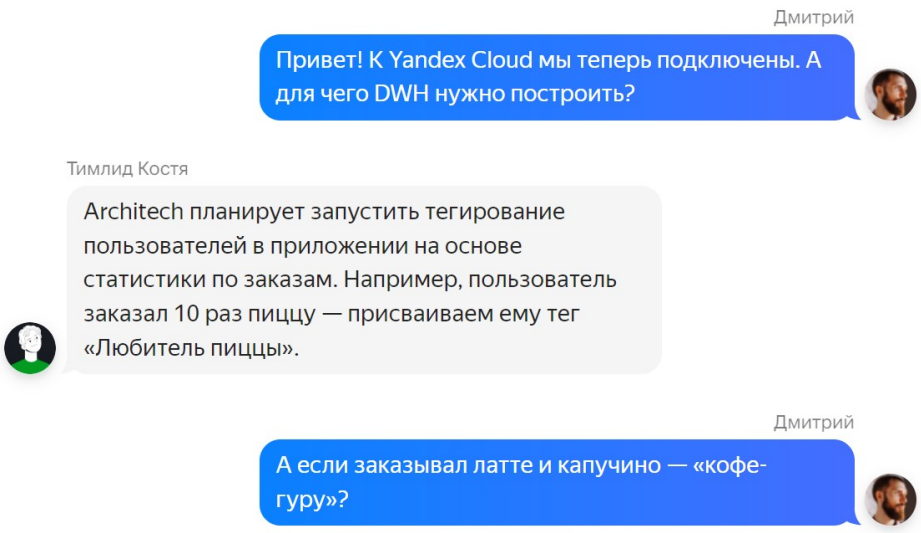
Облачные технологии `Yandex Cloud`

Оглавление

Сбор требований к DWH ▲

Бизнес-требования



Тимлид Костя

Смекаешь 😉

Потом маркетинг будет проводить кампании, заточенные под гостей с определённым тегом. Так рекламные акции станут эффективнее.

Дмитрий

А какие заказы считаем? Отменённые же заказы не будем считать?

Тимлид Костя

Совершенно верно. Все расчёты ведём только по закрытым заказам со статусом `CLOSED`.

Дмитрий

Хорошо. А как считаем?

Тимлид Костя

В каждом заказе есть категория — потом увидишь, когда данные из источника получишь. Для этих категорий заведём счётчики с конкретными порогами. При превышении порога будем назначать гостю тег. Значения порогов позднее у аналитиков уточнишь, сейчас на этапе проектирования это неважно.

Функциональные требования

Дмитрий

Формат входных данных уже известен?

Тимлид Костя

Да, по формату договорились. Это будет JSON, структуру зафиксировали. Я тебе позже описание скину.

Тимлид Костя

Какие слои данных организуешь в хранилище?

Дмитрий

По классике: STG, DDS, CDM. Только вопрос есть: какую модель на DDS взять?

Тимлид Костя

Data Vault.

Дмитрий

В STG храним данные as is?

Тимлид Костя

Конечно.

Дмитрий

А как должна выглядеть витрина?

Тимлид Костя

В CDM нужны две витрины. Первая витрина — счётчик заказов по блюдам; вторая — счётчик заказов по категориям товаров.

Дмитрий

Ок! Структуру витрин спроектирую позднее. Как дойду.

Нефункциональные требования

Дмитрий

Так, а что по нагрузке?

Тимлид Костя

Информация по заказам — это потоковые данные, которые будут передаваться через брокер сообщений. Нагрузка на систему заказов — 5 заказов в минуту. С одной стороны, немного. С другой — 7200 заказов в сутки. И на старте загрузят историю за неделю, т. е. почти 50 000 заказов сразу.

Дмитрий

Брокер сообщений же может прислать сообщение повторно. От дублей надо защищаться?

Тимлид Костя

Верно подмечаешь. Надо обеспечить идемпотентность обработки сообщений из брокера.

Дмитрий

А есть ещё какие-то специфичные требования?

Тимлид Костя

Суммарный бюджет на весь проект не должен превышать 5000 рублей в месяц.

Используем технологии, которые позволят легко масштабировать сервис в будущем. Причем, они будут универсальными, а не уникальными, которые есть только у одного облачного провайдера. Нужно оставить возможность переходить из одного облака в другое.

Подбор технологий

Дмитрий

Какой брокер сообщений будем использовать?

Тимлид Костя

В DWH данные будут передаваться через Kafka.

Дмитрий

А в чём будем хранить данные, полученные через Kafka?

Тимлид Костя

В PostgreSQL. У нас в команде по ней большая экспертиза, знакомая СУБД.

Дмитрий

Получается, обрабатываем поток данных из Kafka и записываем в PostgreSQL.

Тимлид Костя

Всё так. Я уже глянул исходную структуру сообщений. Сразу предупрежу, в сообщениях есть id, которые необходимо будет перевести в имена. Например, блюда, рестораны, пользователи. Для этого тебе понадобится Redis.

Дмитрий

Redis? Что это?

Тимлид Костя

Это key-value хранилище. Пообщайся с Ниной потом подробнее. Она расскажет.

Дмитрий

Понятно, позже напишу ей. А обрабатывать поток чем будем? С помощью Spark Streaming?

Тимлид Костя

Нет, нужно будет написать сервисы на Python и развернуть их в Kubernetes.

Дмитрий

Ещё одно новое слово.

Тимлид Костя

Ничего страшного, скоро со всем познакомишься. Пока считай, что сервис — это программа, которая обрабатывает данные.

Дмитрий

А сколько сервисов мне надо написать?

Тимлид Костя

Три. По одному сервису на слой. Каждый сервис будет читать поток данных из Kafka и формировать свой слой данных. Сервисы так и назови: STG-Service, DDS-Service и CDM-Service.

Аналитики ещё попросили сделать дашборды на основе данных из DWH. Для визуализаций можно DataLens использовать.

Дмитрий

Я так понимаю, все технологии мне придётся самостоятельно в Yandex Cloud поднять?

Тимлид Костя

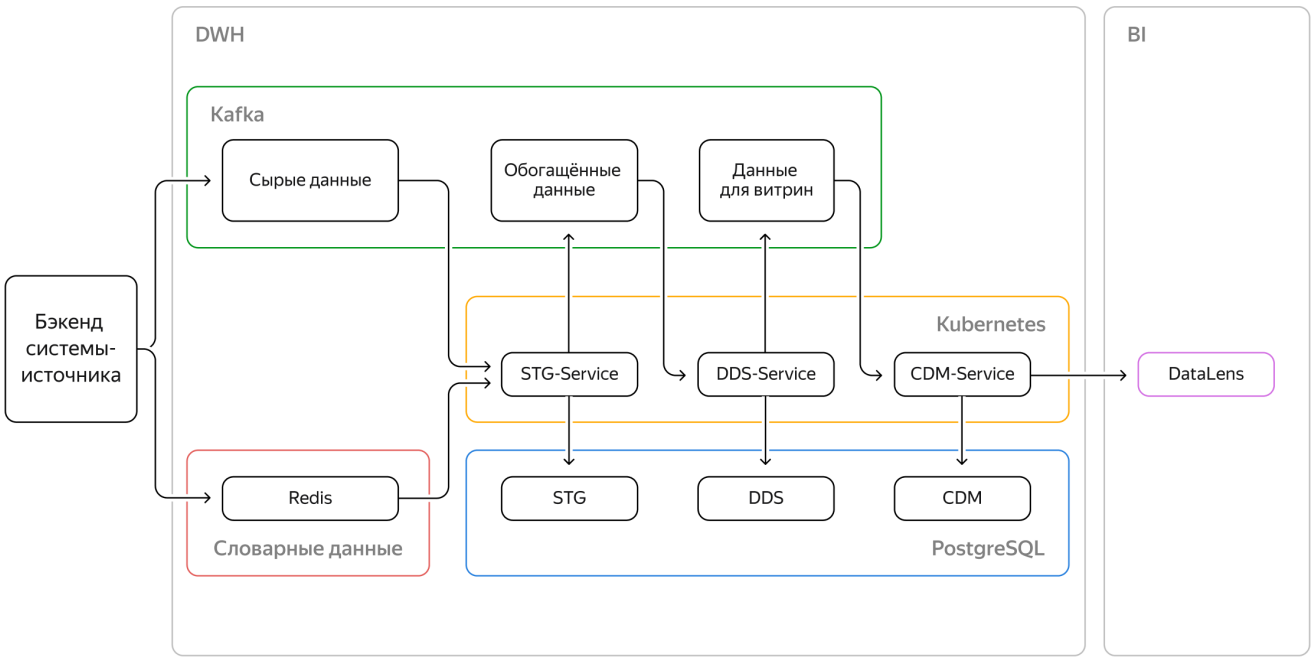
Ну почти. Kubernetes у нас общий с остальными командами. Так что заедем в него. А всё остальное надо будет поднять тебе.

Дмитрий

Вот это вызов! Принимаю!

Проектирование решения

Схема хранилища



Бизнес-задача — «тегирование гостей». Однако при проектировании и реализации нужно помнить, что строится хранилище данных, которое потом будет активно расширяться и развиваться.

С функциональной точки зрения структура хранилища стандартная: слои `STG`, `DDS`, `CDM`

Особенности слоёв:

- В `STG` — исходные данные `as is`
- В `CDM` — две витрины. Первая витрина — счётчик заказов по блюдам; вторая — счётчик заказов по категориям товаров.

- В `DDS` — модель данных `Data Vault`

Данные из системы-источника передаются по двум каналам:

- **Первый канал** — это поток заказов, который идёт в `Kafka` (5 заказов в минуту).
- **Второй канал** — это словарные данные (блюда, рестораны, пользователи), которые идут в `Redis`

В качестве БД используется `PostgreSQL`. Логику обработки данных нужно написать на `Python`, она будет разворачиваться в `Kubernetes`. Брокер сообщений как на вход, так и для обмена данными между сервисами — `Kafka`.

В конце нужно построить дашборды на основе данных в `DWH`

План действий:

- **Шаг 1. Поднять `Redis` и `Kafka` в облаке.**

  > Сначала нужно поднять сервисы, в которые поступают входные данные. Так мы можем сразу изучить оригинальную информацию.

- **Шаг 2. Поднять `PostgreSQL` в облаке.**

  > Продолжаем разворачивать инфраструктуру, чтобы все части `DWH` были готовы к разработке. Системы для получения данных подняты, значит, на очереди запуск базы под хранилище.

- **Шаг 3. Поднять Container `Registry` в облаке.**

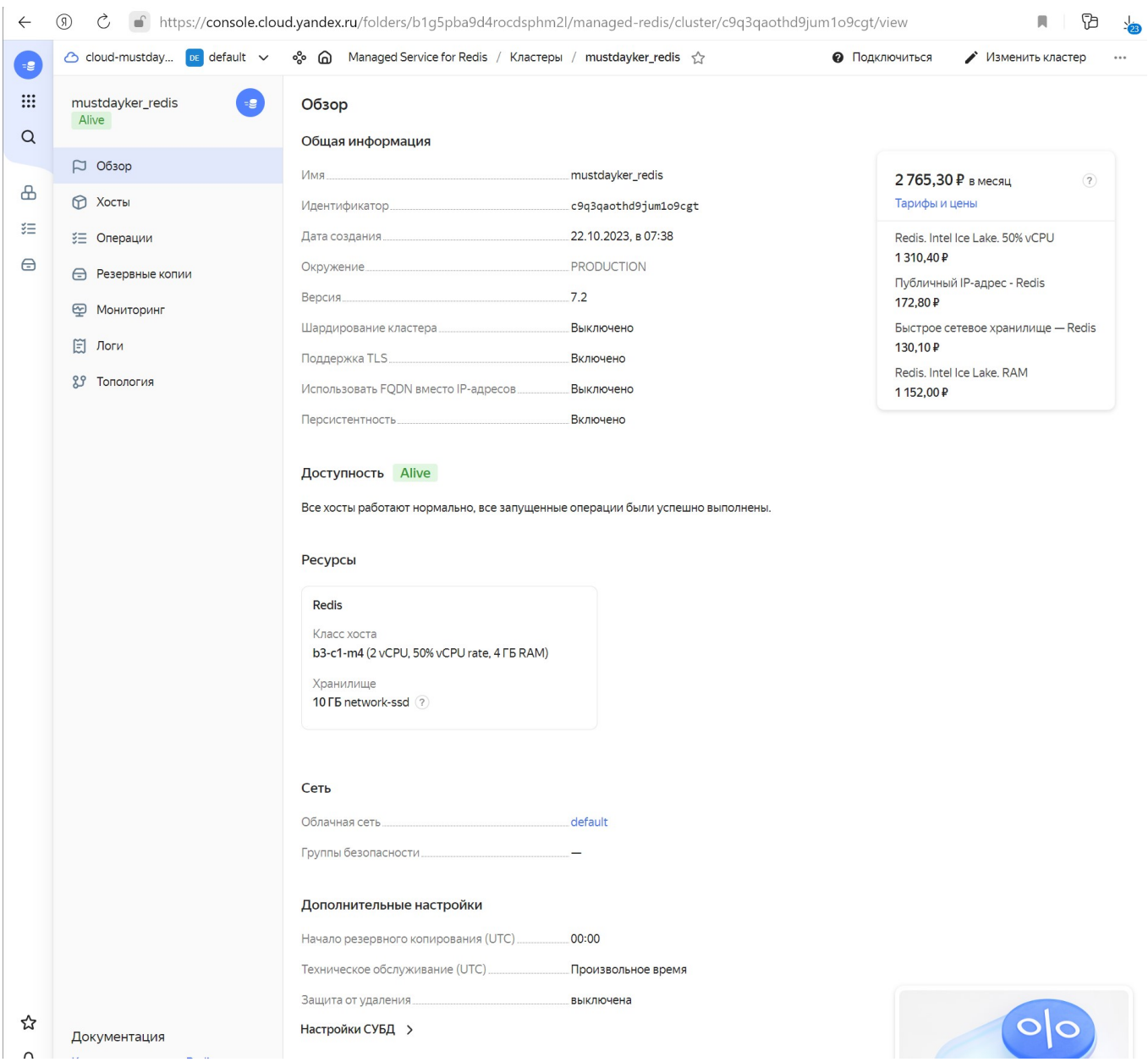
  > Создаем репозиторий контейнеров для наших сервисов.

- **Шаг 4. Собрать шаблон сервиса**
- **Шаг 5. Реализовать `STG`-сервис**
- **Шаг 6. Реализовать `DDS`-сервис**
- **Шаг 7. Реализовать `CDM`-сервис**
- **Шаг 8. Задплоить сервисы в `Kubernetes`**
- **Шаг 9. Визуализировать данные из `DWH` в `Datalens`**

1. Поднимаем `Redis` и `Kafka` в `Yandex Cloud` ▲

1.1 Поднимаем `Redis` в `Yandex Cloud` ▲

Параметр	Значение
Окружение	`PRODUCTION`
Поддержка `TLS`	`Есть`
Персистентность	`Есть`
Шардирование	`Нет`
Платформа	`Intel Ice Lake`
Тип ВМ	`burstable`
Хост	`b3-c1-m4`
Публичный доступ	`Разрешён`
Размер хранилища	`10 Гб`
Дополнительные настройки	`По умолчанию`



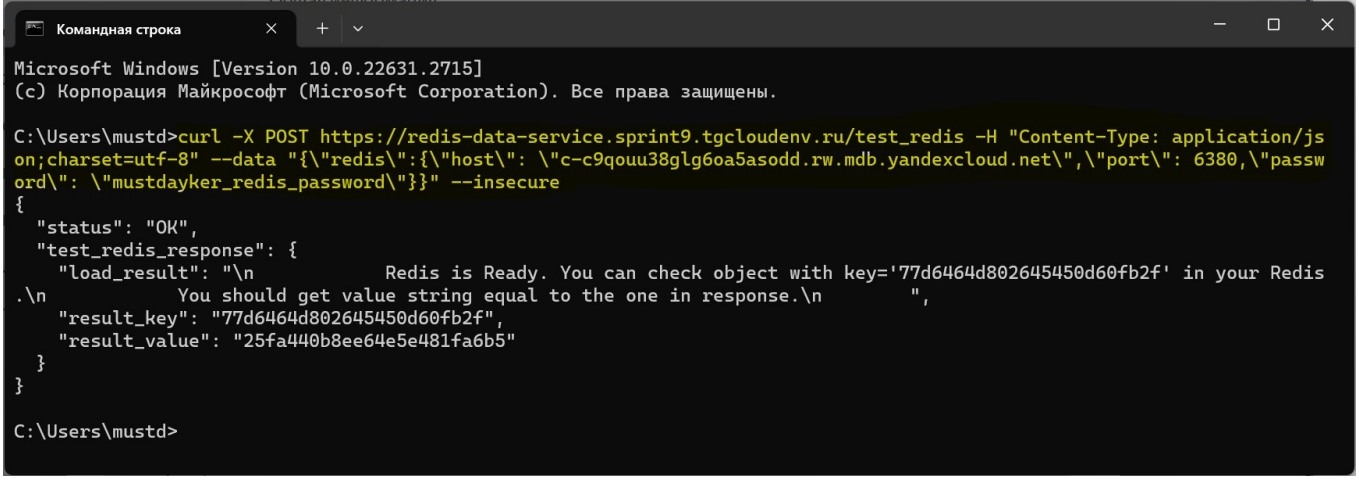
Параметры подключения

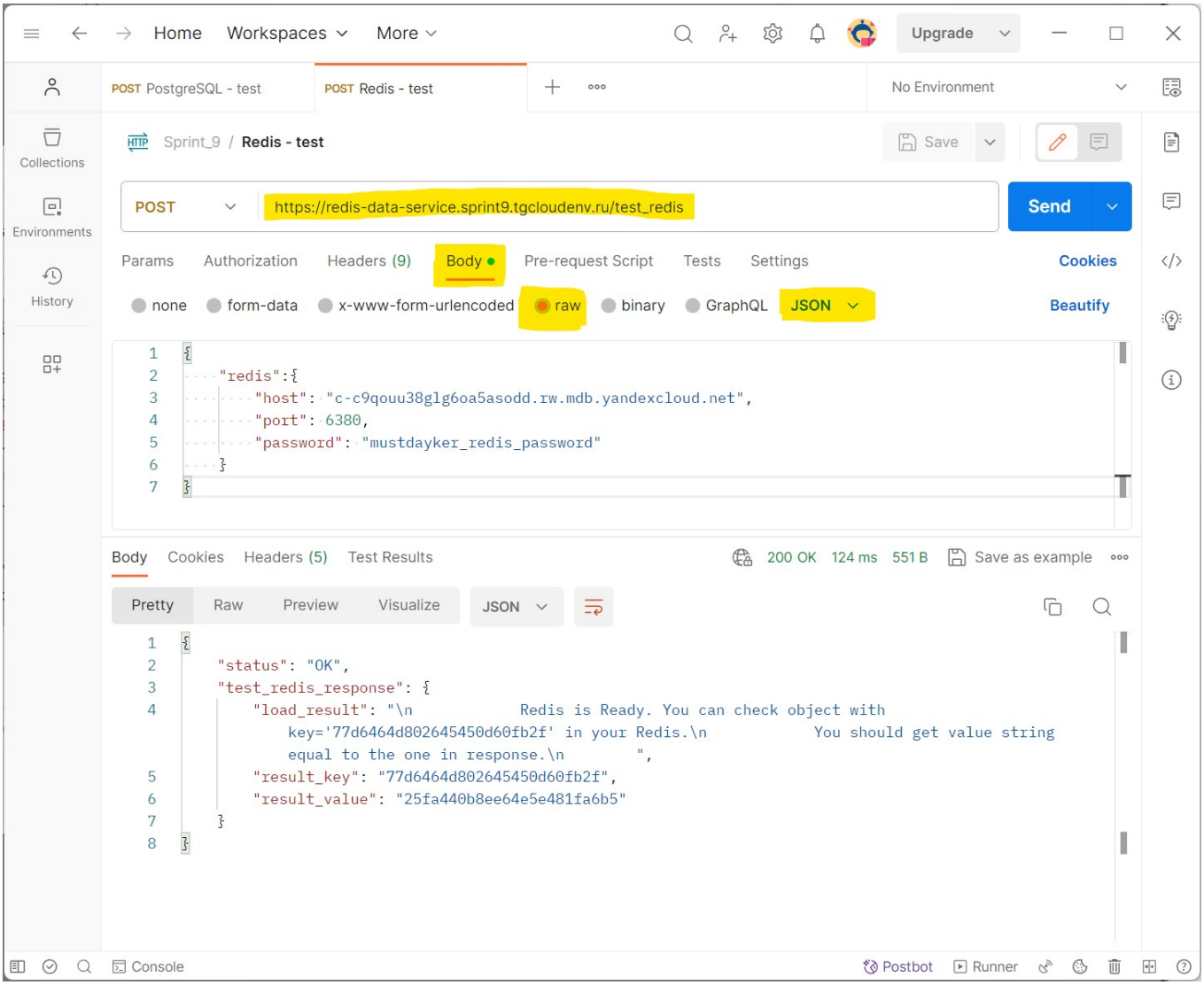
Параметр	Значение
Хост	`c-c9qouu38glg6oa5asodd.rw.mdb.yandexcloud.net`
Порт	`6380`
Пароль	`mustdayker_redis_password`
Путь до сертификата	`c:/GitHub/data_engineer/09_yandex_cloud/cert/.redis/YandexInternalRootCA.crt`

Проверяем доступность

```
curl -X POST https://redis-data-service.sprint9.tgcloudenv.ru/test_redis -H "Content-Type: application/json;charset=utf-8" --data "{\"redis\":{\"host\": \"c-c9qouu38glg6oa5asodd.rw.mdb.yandexcloud.net\",\"port\": 6380,\"password\": \"mustdayker_redis_password\"}}" --insecure
```

Результат

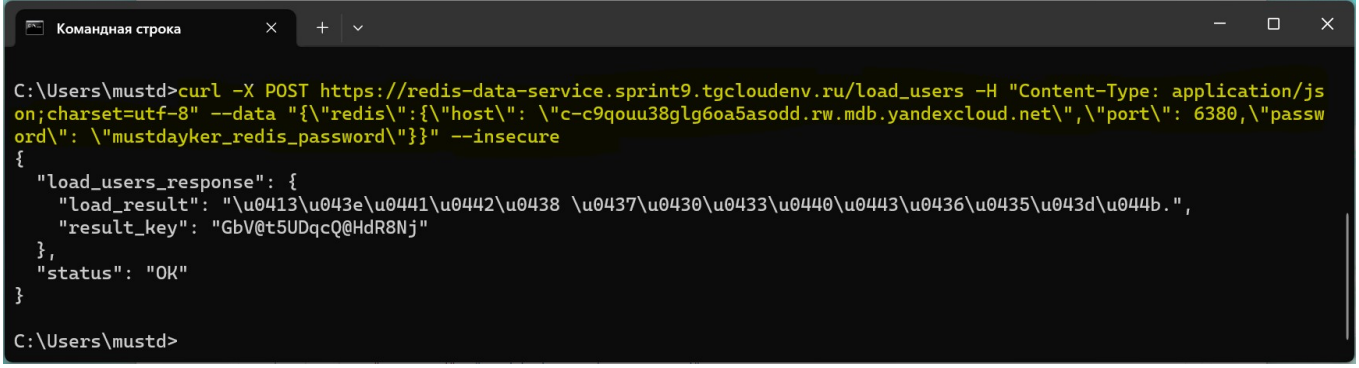


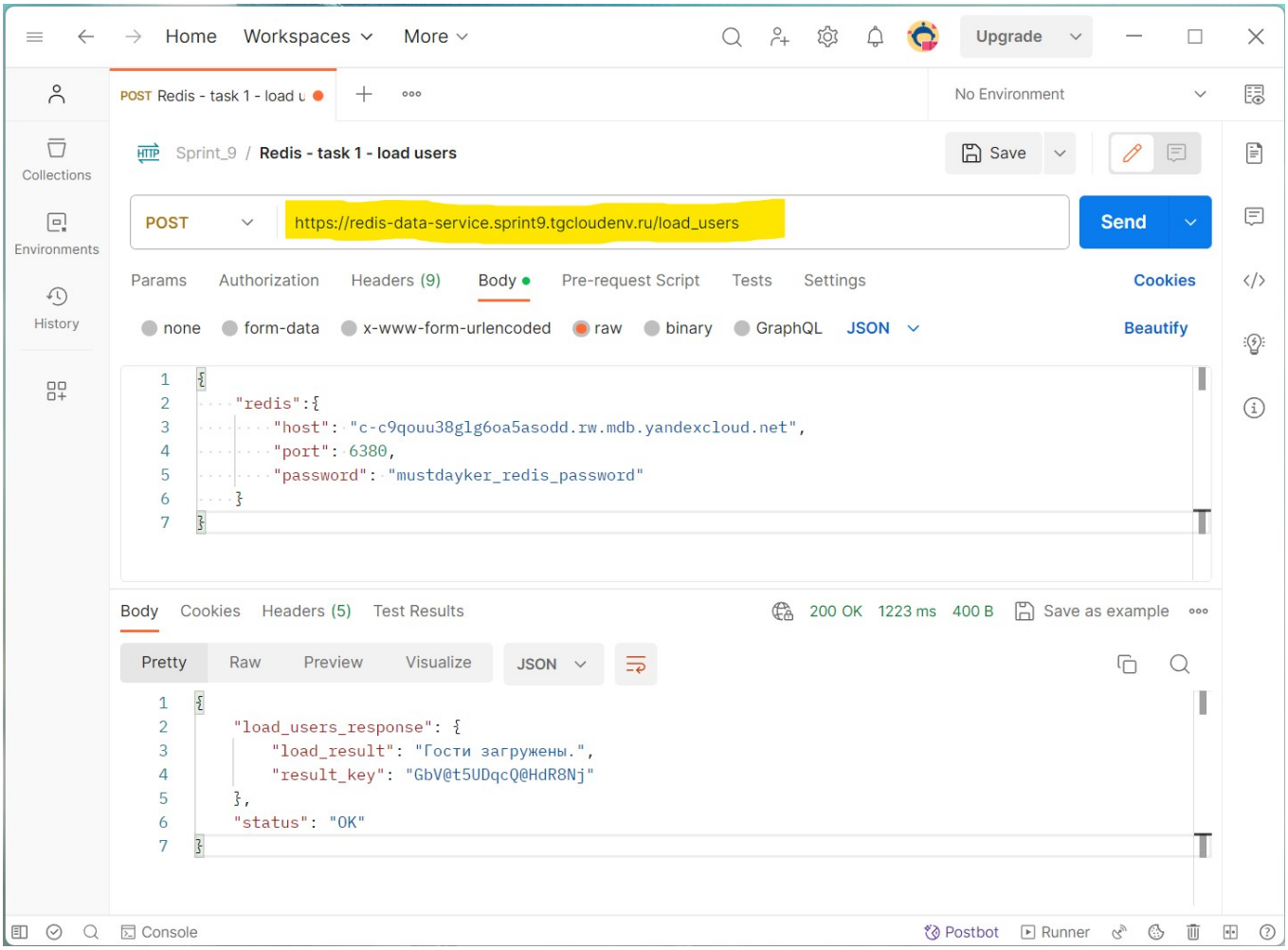


Загрузка пользователей

```
curl -X POST https://redis-data-service.sprint9.tgcloudenv.ru/load_users -H "Content-Type: application/json;charset=utf-8" --data "{\"redis\":{\"host\": \"c-c9qouu38glg6oa5asodd.rw.mdb.yandexcloud.net\",\"port\": 6380,\"password\": \"mustdayker_redis_password\"}}" --insecure
```

Результат

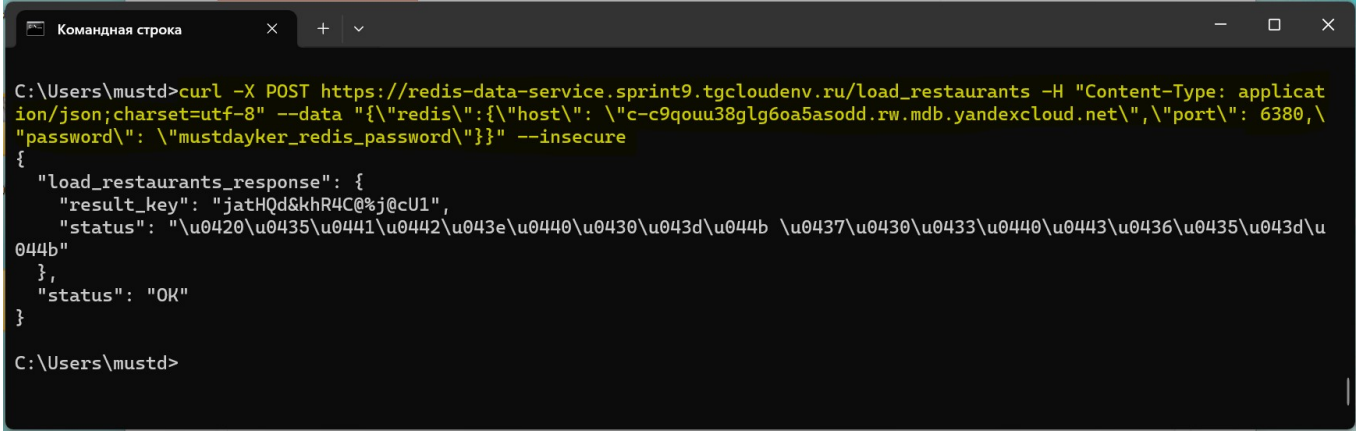


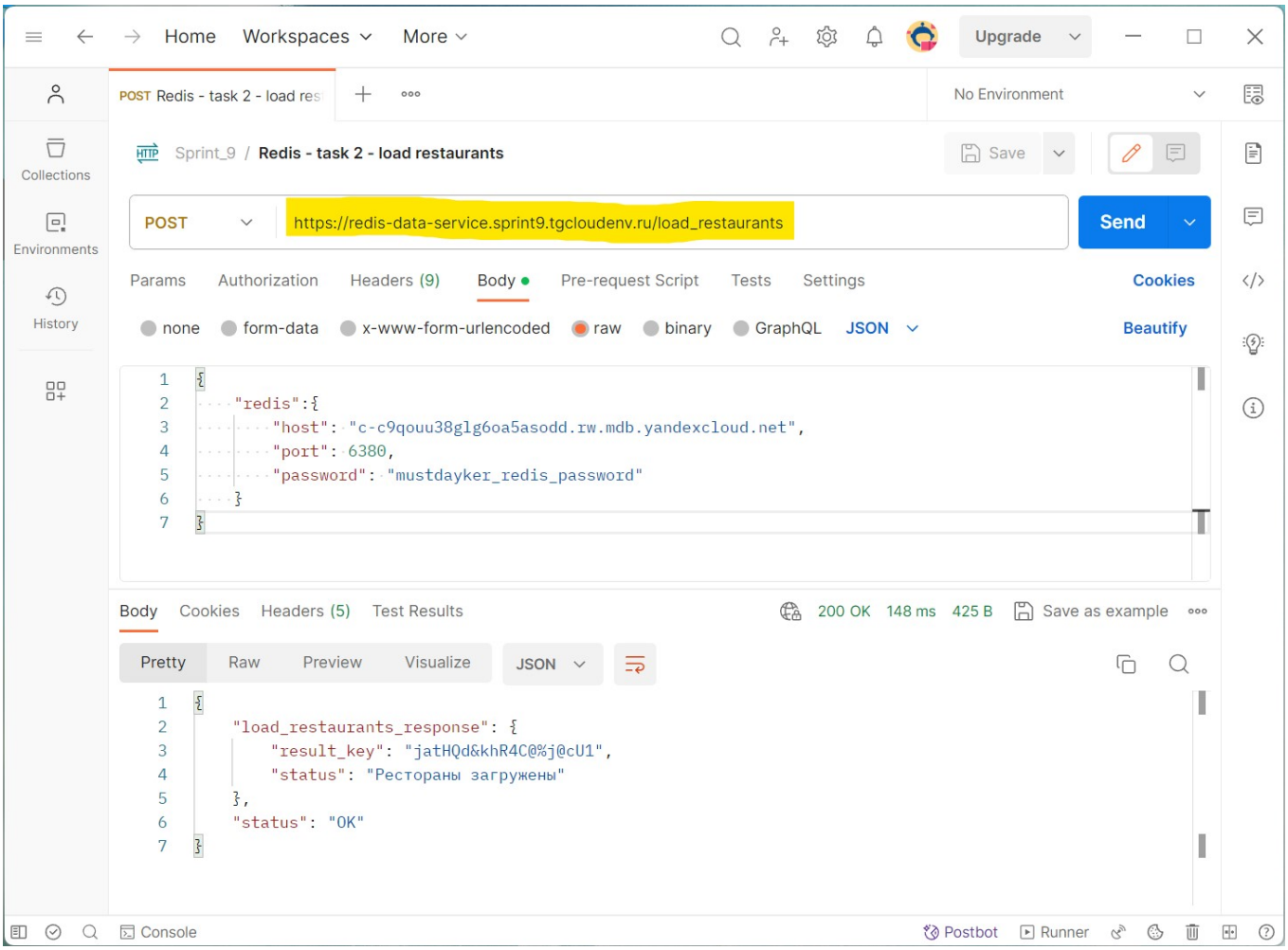


Загрузка ресторанов

```
curl -X POST https://redis-data-service.sprint9.tgcloudenv.ru/load_restaurants -H "Content-Type: application/json;charset=utf-8" --data "{\"redis\":{\"host\": \"c-c9qouu38glg6oa5asodd.rw.mdb.yandexcloud.net\",\"port\": 6380,\"password\": \"mustdayker_redis_password\"}}" --insecure
```

Результат

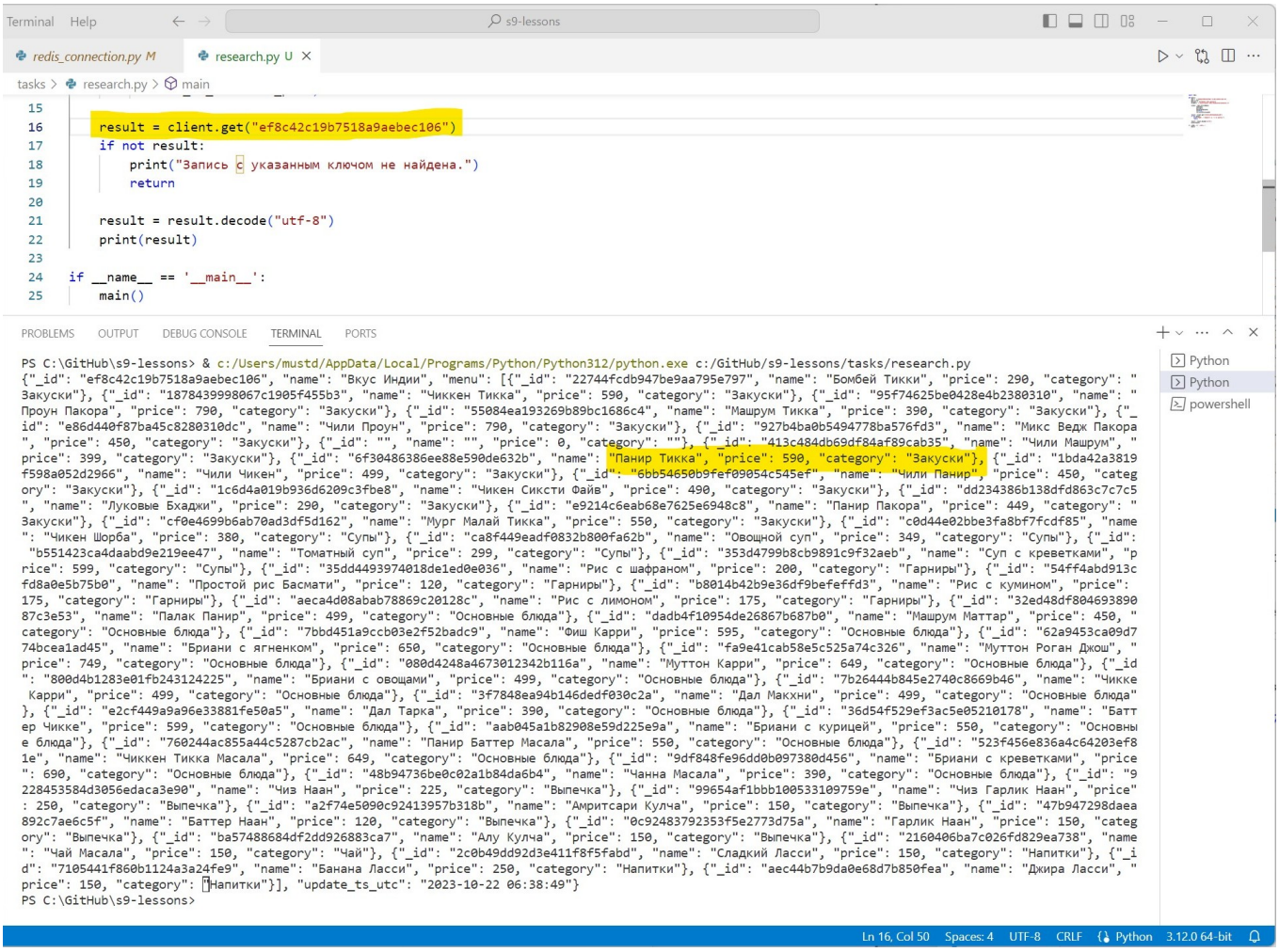


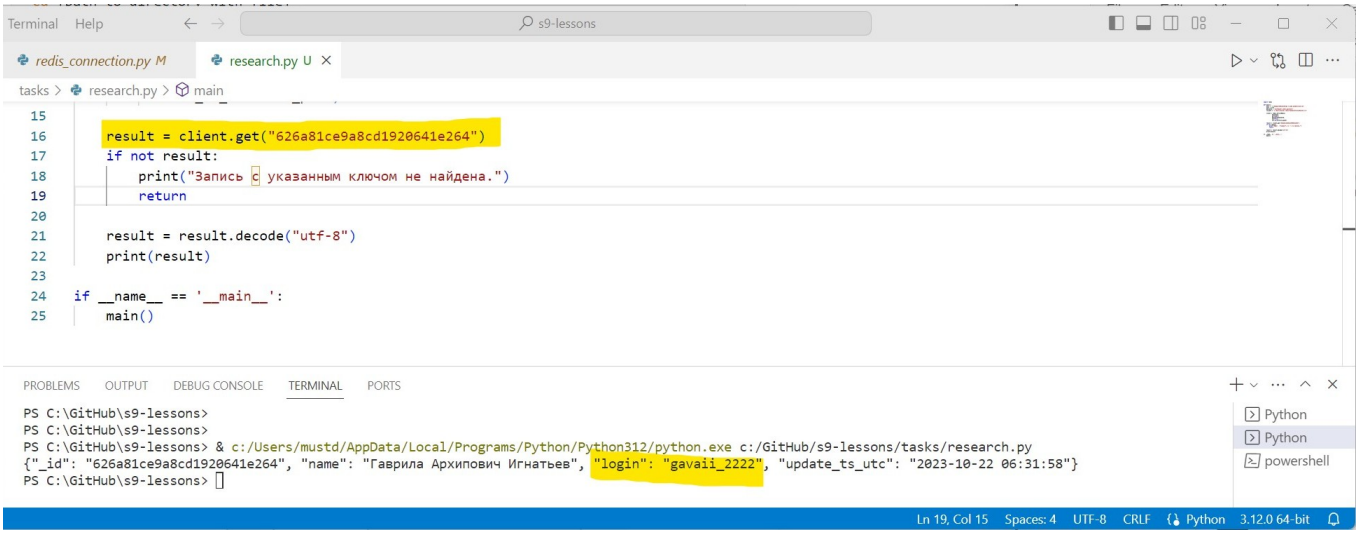


Изучение данных в `Redis`

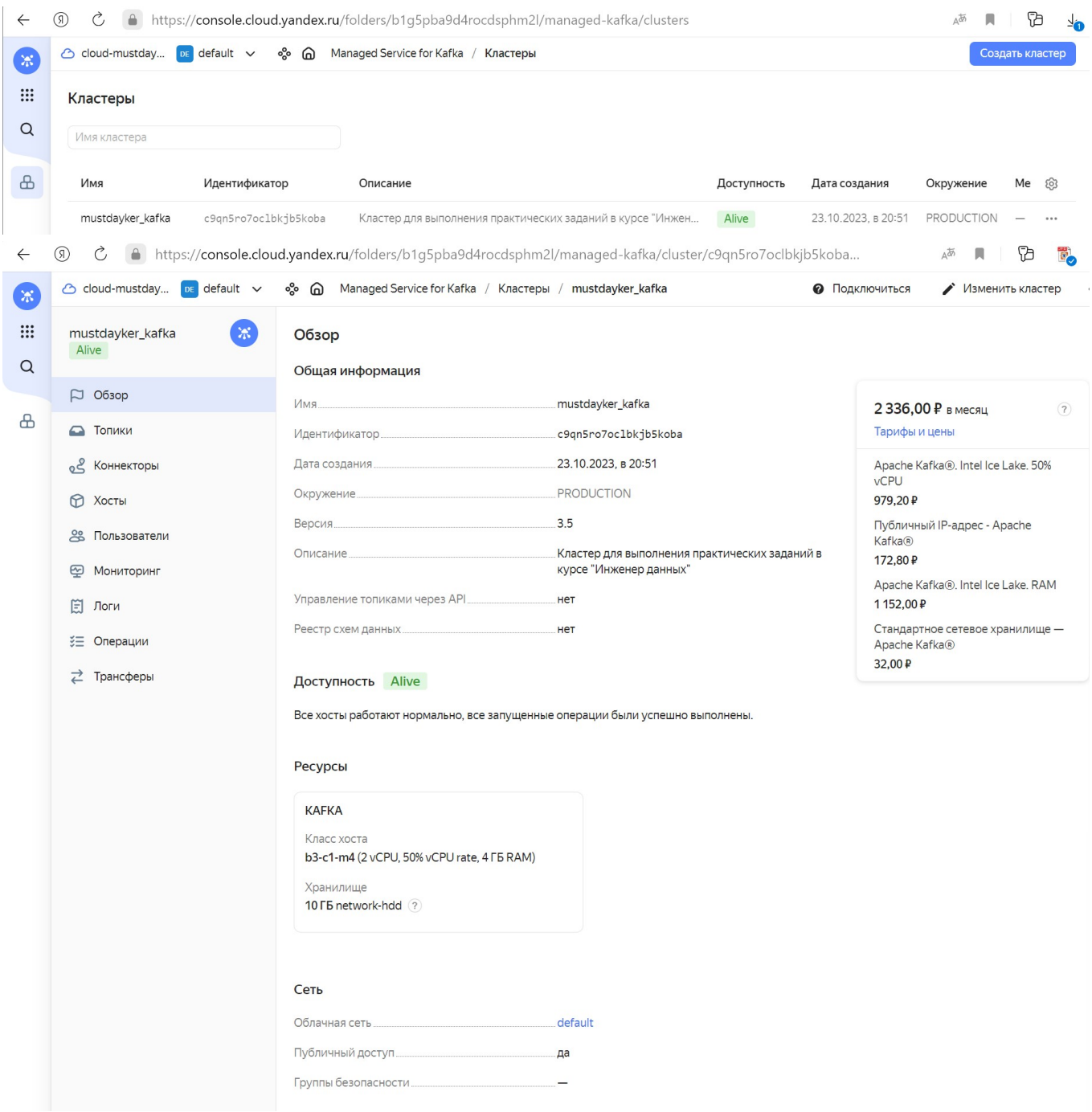
```
import redis

def main():
    host = 'c-c9qouu38glg6oa5asodd.rw.mdb.yandexcloud.net'
    port = 6380
    password = 'mustdayker_redis_password'
    ca_path = 'c:/GitHub/data_engineer/09_yandex_cloud/cert/.redis/YandexInternalRootCA.crt'

    client = redis.StrictRedis(
            host=host,
            port=port,
            password=password,
            ssl=True,
            ssl_ca_certs=ca_path)

    result = client.get("626a81ce9a8cd1920641e264")
    if not result:
        print("Запись с указанным ключом не найдена.")
        return

    result = result.decode("utf-8")
    print(result)
```

```python
if __name__ == '__main__':
    main()
```

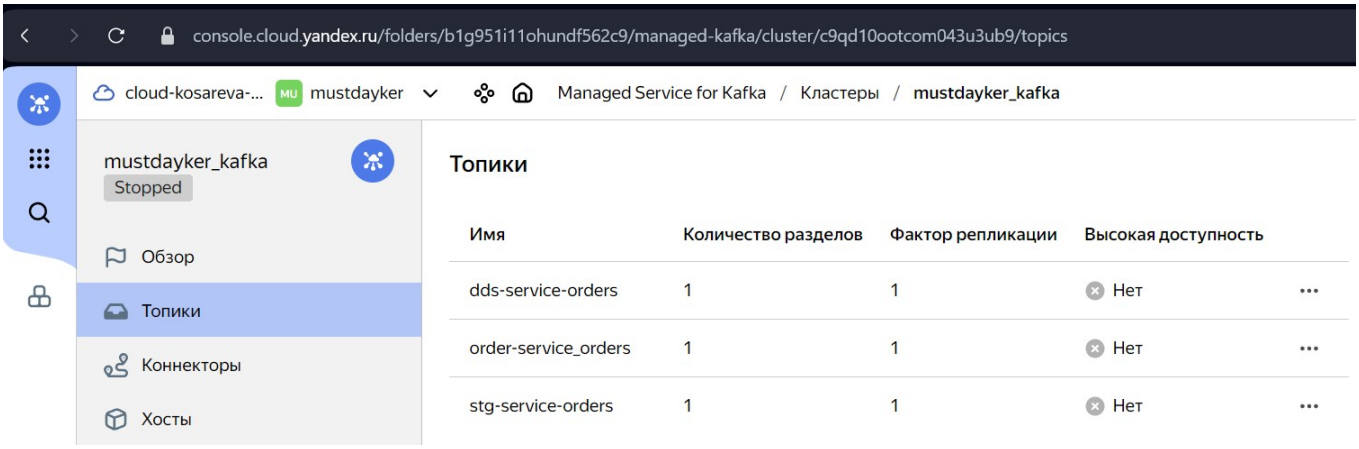




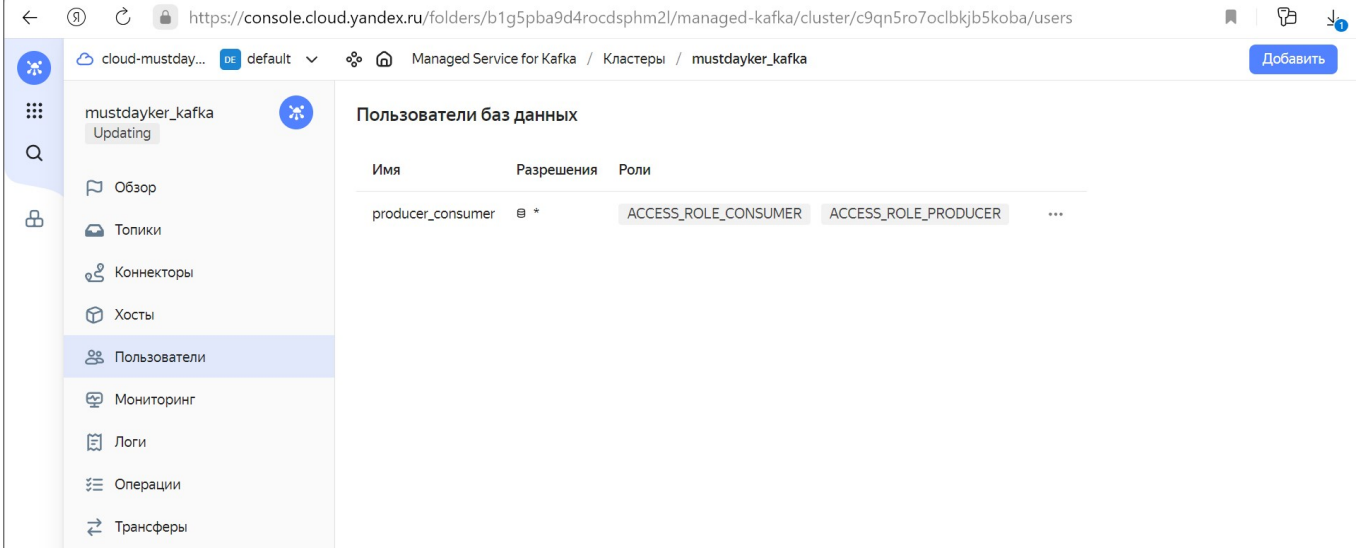
1.2 Поднимаем `Kafka` в `Yandex Cloud` ▲



Создание топиков



Добавление пользователя

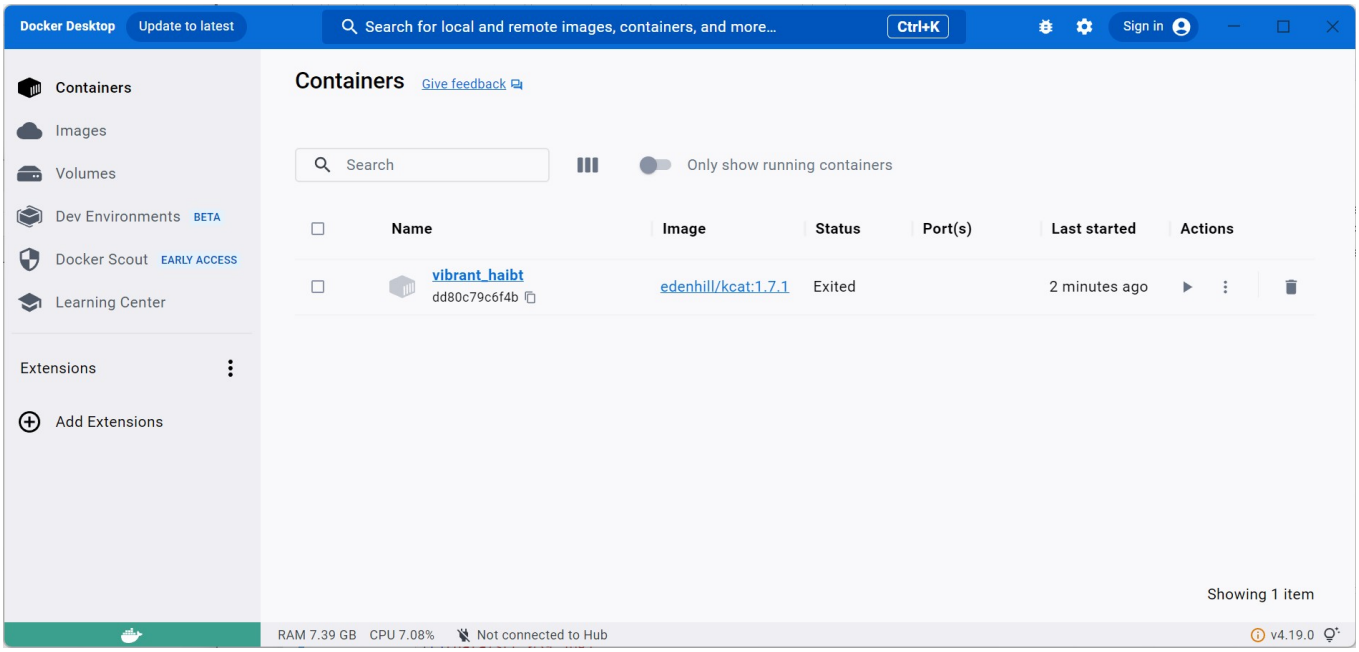


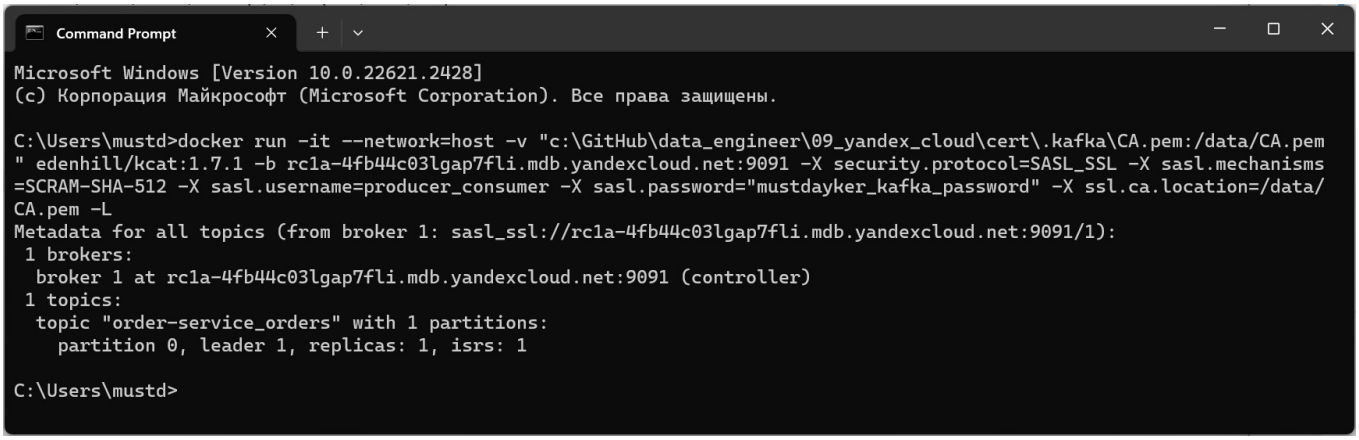
Параметры подключения

Параметр	Значение
Хост	`rc1a-93jto5vap3spn3u6.mdb.yandexcloud.net`
Порт	`9091`
Хост и порт	`rc1a-93jto5vap3spn3u6.mdb.yandexcloud.net:9091`
Пользователь	`producer_consumer`
Пароль	`mustdayker_kafka_password`
Путь до сертификата	`c:\GitHub\data_engineer\09_yandex_cloud\cert\.kafka\CA.pem`
Топик `order`	`order-service_orders`
Топик `stg`	`stg-service-orders`
Топик `dds`	`dds-service-orders`

Проверка доступности кластера

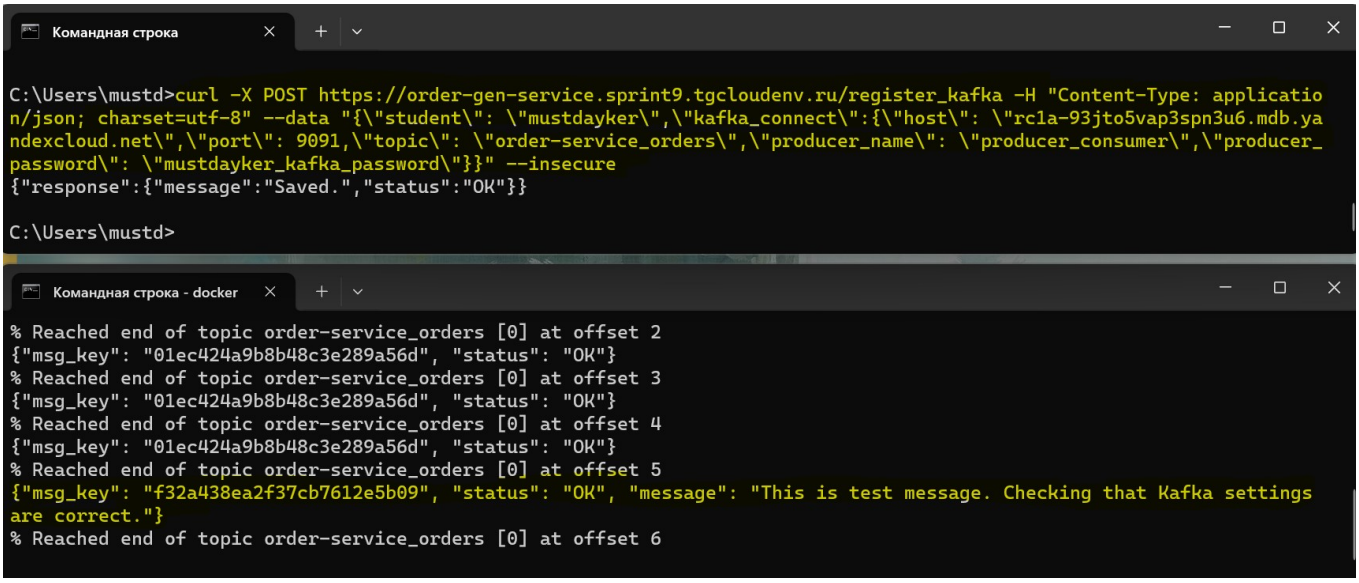
```
docker run -it --network=host -v "c:\GitHub\data_engineer\09_yandex_cloud\cert\.kafka\CA.pem:/data/CA.pem"
edenhill/kcat:1.7.1 -b rc1a-93jto5vap3spn3u6.mdb.yandexcloud.net:9091 -X security.protocol=SASL_SSL -X
sasl.mechanisms=SCRAM-SHA-512 -X sasl.username=producer_consumer -X
sasl.password="mustdayker_kafka_password" -X ssl.ca.location=/data/CA.pem -L
```





Инициируем отправку сообщений в топик `order-service_orders`

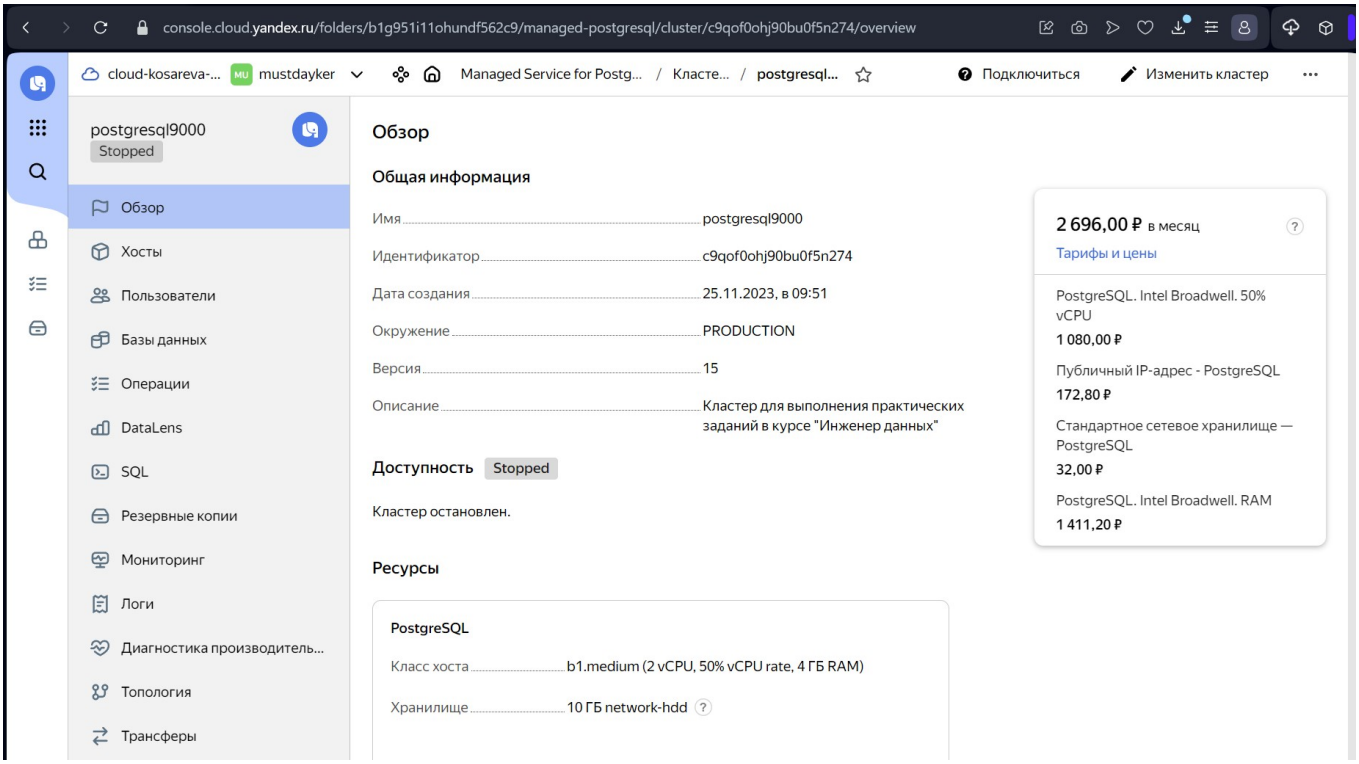
```
curl -X POST https://order-gen-service.sprint9.tgcloudenv.ru/register_kafka -H "Content-Type: application/json; charset=utf-8" --data "{\"student\": \"mustdayker\",\"kafka_connect\":{\"host\": \"rc1a-93jto5vap3spn3u6.mdb.yandexcloud.net\",\"port\": 9091,\"topic\": \"order-service_orders\",\"producer_name\": \"producer_consumer\",\"producer_password\": \"mustdayker_kafka_password\"}}" --insecure
```



Исследуем полученое сообщение


```
{
    "object_id": 3126984,
    "object_type": "order",
    "sent_dttm": "2023-11-25 17:39:51",
    "payload": {
        "restaurant": {"id": "626a81cfefa404208fe9abae"},
        "date": "2023-11-18 17:40:02",
        "user": {"id": "626a81ce9a8cd1920641e272"},
        "order_items": [
            {"id": "6276e8cd0cf48b4cded0086e", "name": "Ролл С ВЕТЧИНОЙ И ОМЛЕТОМ", "price": 120, "quantity": 3},
            {"id": "6276e8cd0cf48b4cded00879", "name": "Ролл С РЫБОЙ И СОУСОМ ТАРТАР", "price": 180, "quantity": 5},
            {"id": "6276e8cd0cf48b4cded00875", "name": "КАЛЬЦОНЕ МАРГАРИТА", "price": 120, "quantity": 4},
            {"id": "6276e8cd0cf48b4cded0087a", "name": "НОРВЕЖСКИЙ СЭНДВИЧ", "price": 180, "quantity": 4}
        ],
        "bonus_payment": 0,
        "cost": 2460,
        "payment": 2460,
        "bonus_grant": 0,
        "statuses": [
            {"status": "CLOSED", "dttm": "2023-11-18 17:40:02"},
            {"status": "DELIVERING", "dttm": "2023-11-18 16:56:24"},
            {"status": "COOKING", "dttm": "2023-11-18 16:39:28"},
            {"status": "OPEN", "dttm": "2023-11-18 16:02:44"}
        ],
        "final_status": "CLOSED", "update_ts": "2023-11-18 17:40:02"}
}
```

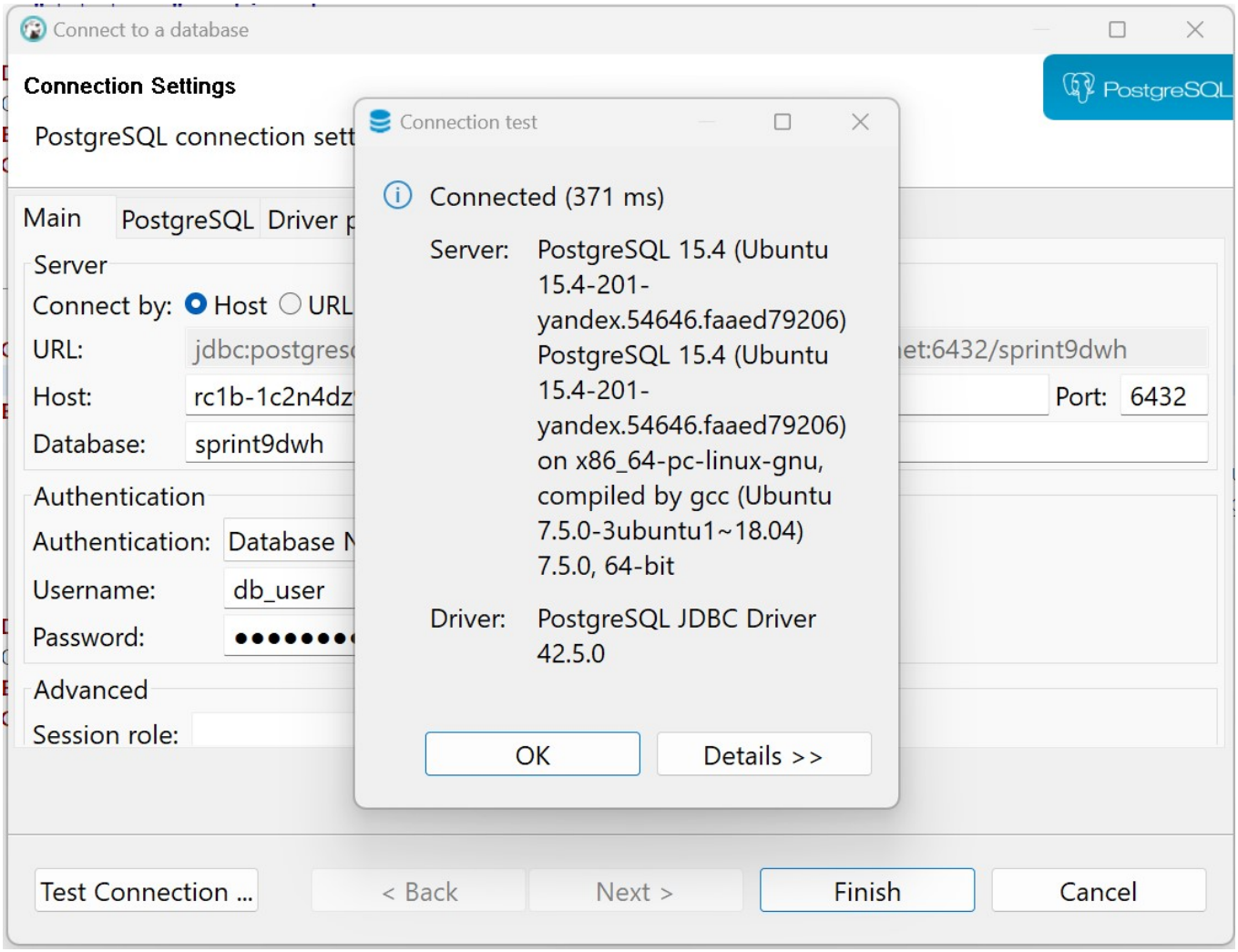

2. Поднимаем `PostgreSQL` в `Yandex Cloud` ▲



Параметры подключения к базе

Параметр	Значение
Хост	`rc1b-1c2n4dz99tf0temj.mdb.yandexcloud.net`
Порт	`6432`
Имя БД	`sprint9dwh`
Пользователь	`db_user`
Пароль	`db_user_password`
sslmode	`verify-full`

Проверка подключения



2.1 Наполняем БД

Создаем схемы

```
CREATE SCHEMA stg;
CREATE SCHEMA dds;
CREATE SCHEMA cdm;
```

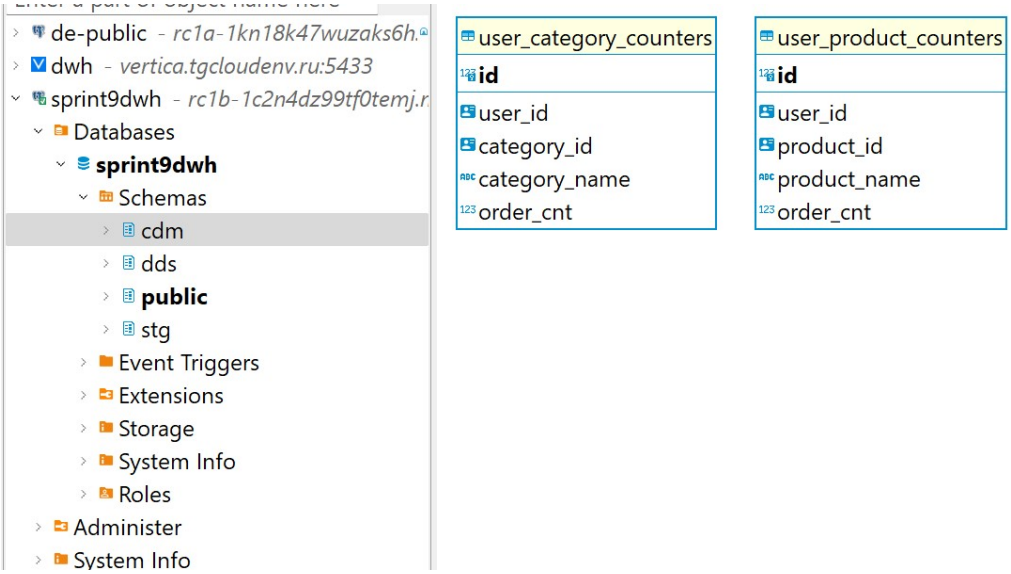
Заполняем слой `CDM`

`cdm.user_product_counters`

```
CREATE TABLE IF NOT EXISTS cdm.user_product_counters (
    id           SERIAL  PRIMARY KEY,
    user_id      UUID    NOT NULL,
    product_id   UUID    NOT NULL,
    product_name VARCHAR NOT NULL,
    order_cnt    INT     NOT NULL CHECK (order_cnt >= 0),
    UNIQUE (user_id, product_id)
);
```

`cdm.user_category_counters`

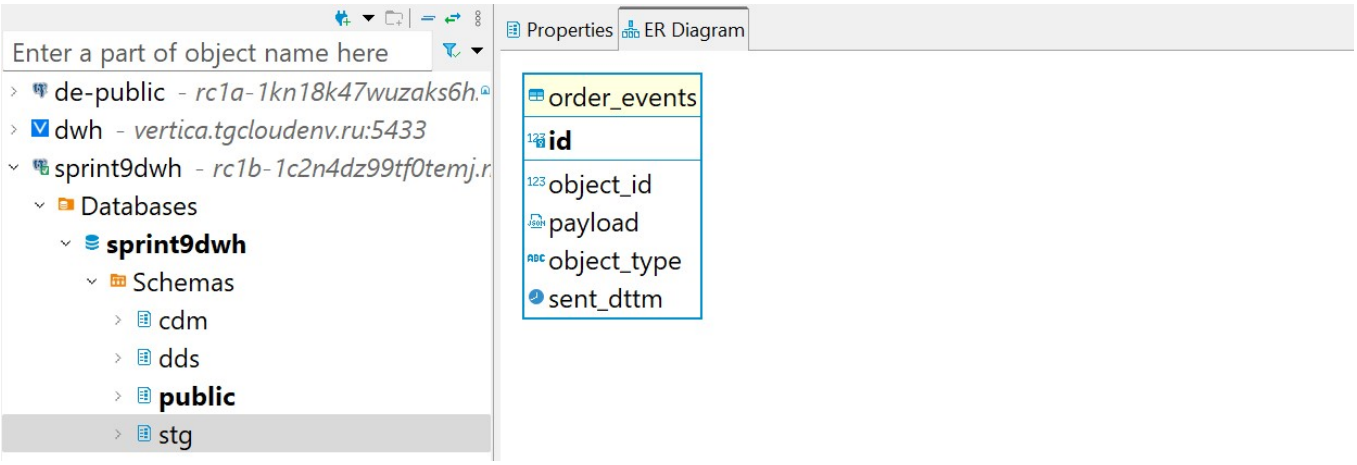
```
CREATE TABLE IF NOT EXISTS cdm.user_category_counters (
    id            SERIAL  PRIMARY KEY,
    user_id       UUID    NOT NULL,
    category_id   UUID    NOT NULL,
    category_name VARCHAR NOT NULL,
    order_cnt     INT     NOT NULL CHECK (order_cnt >= 0),
    UNIQUE (user_id, category_id)
);
```



Заполняем слой `STG`

`stg.order_events`

```
CREATE TABLE IF NOT EXISTS stg.order_events (
    id          SERIAL    PRIMARY KEY,
    object_id   integer   NOT NULL UNIQUE,
    payload     JSON      NOT NULL,
    object_type VARCHAR   NOT NULL,
    sent_dttm   timestamp NOT NULL
);
```



Заполняем слой `DDS`

Создаем хабы

`dds.h_user`

```
CREATE TABLE IF NOT EXISTS dds.h_user (
    h_user_pk UUID      NOT NULL PRIMARY KEY,
    user_id   VARCHAR   NOT NULL,
    load_dt   timestamp NOT NULL,
    load_src  VARCHAR   NOT NULL
);
```

`dds.h_product`

```
CREATE TABLE IF NOT EXISTS dds.h_product (
    h_product_pk UUID      NOT NULL PRIMARY KEY,
    product_id   VARCHAR   NOT NULL,
    load_dt      timestamp NOT NULL,
    load_src     VARCHAR   NOT NULL
);
```

`dds.h_category`

```
CREATE TABLE IF NOT EXISTS dds.h_category (
    h_category_pk UUID      NOT NULL PRIMARY KEY,
    category_name VARCHAR   NOT NULL,
    load_dt       timestamp NOT NULL,
    load_src      VARCHAR   NOT NULL
);
```

`dds.h_restaurant`

```
CREATE TABLE IF NOT EXISTS dds.h_restaurant (
    h_restaurant_pk UUID      NOT NULL PRIMARY KEY,
    restaurant_id   VARCHAR   NOT NULL,
    load_dt         timestamp NOT NULL,
    load_src        VARCHAR   NOT NULL
);
```

`dds.h_order`

```
CREATE TABLE IF NOT EXISTS dds.h_order (
    h_order_pk UUID      NOT NULL PRIMARY KEY,
    order_id   integer   NOT NULL,
    order_dt   timestamp NOT NULL,
    load_dt    timestamp NOT NULL,
    load_src   VARCHAR   NOT NULL
);
```

Создаем линки

`dds.l_order_product`

```
CREATE TABLE IF NOT EXISTS dds.l_order_product (
    hk_order_product_pk UUID      NOT NULL PRIMARY KEY,
    h_order_pk          UUID      NOT NULL,
    h_product_pk        UUID      NOT NULL,
    load_dt             timestamp NOT NULL,
    load_src            VARCHAR   NOT NULL,
    FOREIGN KEY (h_order_pk)   REFERENCES dds.h_order(h_order_pk),
    FOREIGN KEY (h_product_pk) REFERENCES dds.h_product(h_product_pk)
);
```

`dds.l_product_restaurant`

```
CREATE TABLE IF NOT EXISTS dds.l_product_restaurant (
    hk_product_restaurant_pk UUID      NOT NULL PRIMARY KEY,
    h_product_pk             UUID      NOT NULL,
    h_restaurant_pk          UUID      NOT NULL,
    load_dt                  timestamp NOT NULL,
    load_src                 VARCHAR   NOT NULL,
    FOREIGN KEY (h_product_pk)    REFERENCES dds.h_product(h_product_pk),
    FOREIGN KEY (h_restaurant_pk) REFERENCES dds.h_restaurant(h_restaurant_pk)
);
```

`dds.l_product_category`

```
CREATE TABLE IF NOT EXISTS dds.l_product_category (
    hk_product_category_pk UUID      NOT NULL PRIMARY KEY,
    h_product_pk           UUID      NOT NULL,
    h_category_pk          UUID      NOT NULL,
    load_dt                timestamp NOT NULL,
    load_src               VARCHAR   NOT NULL,
    FOREIGN KEY (h_product_pk)  REFERENCES dds.h_product(h_product_pk),
    FOREIGN KEY (h_category_pk) REFERENCES dds.h_category(h_category_pk)
);
```

`dds.l_order_user`

```
CREATE TABLE IF NOT EXISTS dds.l_order_user (
    hk_order_user_pk UUID      NOT NULL PRIMARY KEY,
    h_order_pk       UUID      NOT NULL,
    h_user_pk        UUID      NOT NULL,
    load_dt          timestamp NOT NULL,
    load_src         VARCHAR   NOT NULL,
    FOREIGN KEY (h_order_pk) REFERENCES dds.h_order(h_order_pk),
    FOREIGN KEY (h_user_pk)  REFERENCES dds.h_user(h_user_pk)
);
```

Создаем сателлиты

dds.s_user_names

```
CREATE TABLE IF NOT EXISTS dds.s_user_names (
    h_user_pk              UUID      NOT NULL,
    username               VARCHAR   NOT NULL,
    userlogin              VARCHAR   NOT NULL,
    load_dt                timestamp NOT NULL,
    load_src               VARCHAR   NOT NULL,
    hk_user_names_hashdiff UUID      NOT NULL,
    CONSTRAINT hk_user_names_pk PRIMARY KEY (h_user_pk, load_dt),
    FOREIGN KEY (h_user_pk) REFERENCES dds.h_user(h_user_pk)
);
```

dds.s_product_names

```
CREATE TABLE IF NOT EXISTS dds.s_product_names (
    h_product_pk              UUID      NOT NULL,
    "name"                    VARCHAR   NOT NULL,
    load_dt                   timestamp NOT NULL,
    load_src                  VARCHAR   NOT NULL,
    hk_product_names_hashdiff UUID      NOT NULL,
    CONSTRAINT hk_product_names_pk PRIMARY KEY (h_product_pk, load_dt),
    FOREIGN KEY (h_product_pk) REFERENCES dds.h_product(h_product_pk)
);
```

dds.s_restaurant_names

```
CREATE TABLE IF NOT EXISTS dds.s_restaurant_names (
    h_restaurant_pk              UUID      NOT NULL,
    "name"                       VARCHAR   NOT NULL,
    load_dt                      timestamp NOT NULL,
    load_src                     VARCHAR   NOT NULL,
    hk_restaurant_names_hashdiff UUID      NOT NULL,
    CONSTRAINT hk_restaurant_names_pk PRIMARY KEY (h_restaurant_pk, load_dt),
    FOREIGN KEY (h_restaurant_pk) REFERENCES dds.h_restaurant(h_restaurant_pk)
);
```

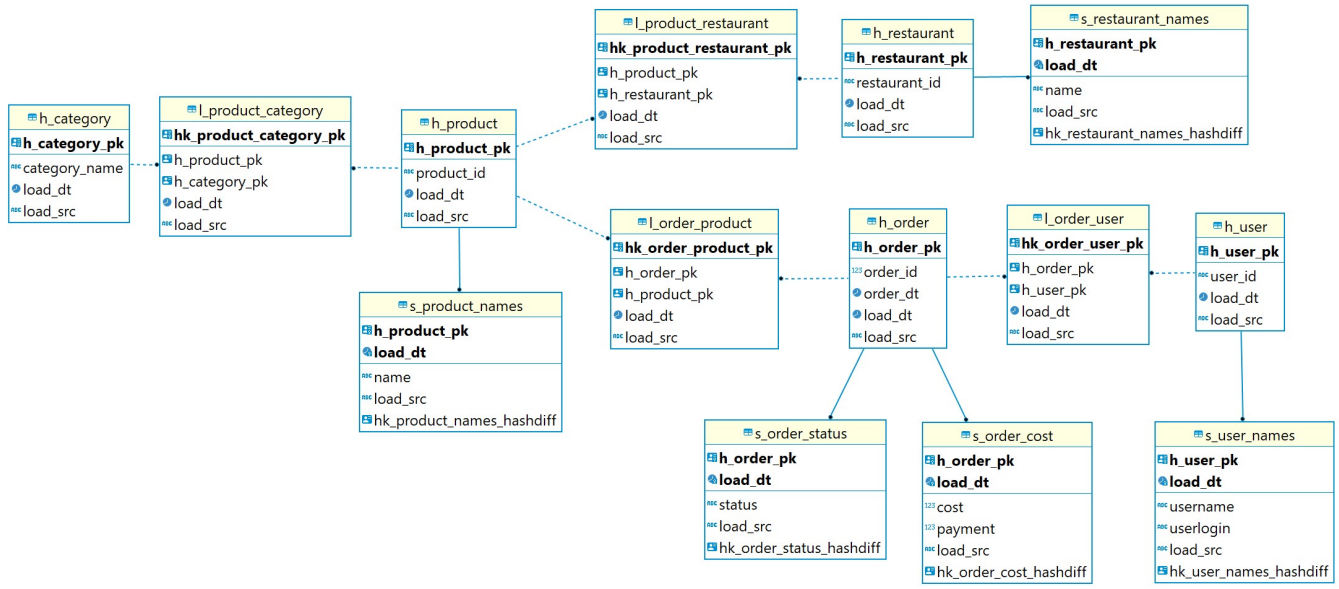
dds.s_order_cost

```
CREATE TABLE IF NOT EXISTS dds.s_order_cost (
    h_order_pk             UUID           NOT NULL,
    "cost"                 decimal(19, 5) NOT NULL,
    payment                decimal(19, 5) NOT NULL,
    load_dt                timestamp      NOT NULL,
    load_src               VARCHAR        NOT NULL,
    hk_order_cost_hashdiff UUID           NOT NULL,
    CONSTRAINT hk_order_cost_pk PRIMARY KEY (h_order_pk, load_dt),
    FOREIGN KEY (h_order_pk) REFERENCES dds.h_order(h_order_pk)
);
```

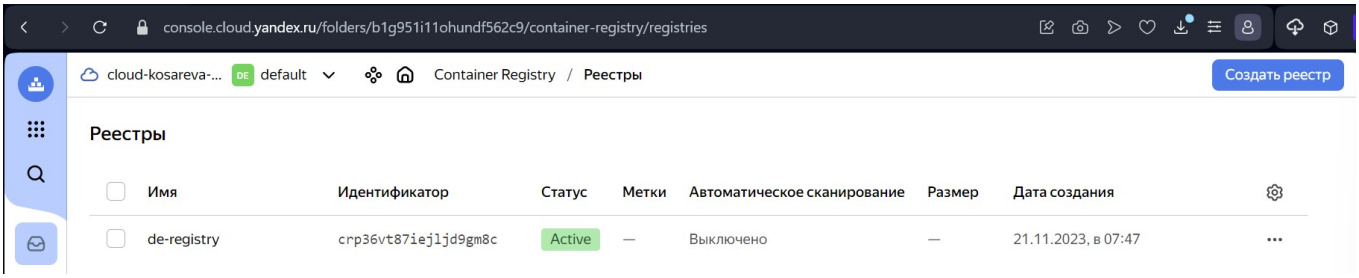
dds.s_order_status

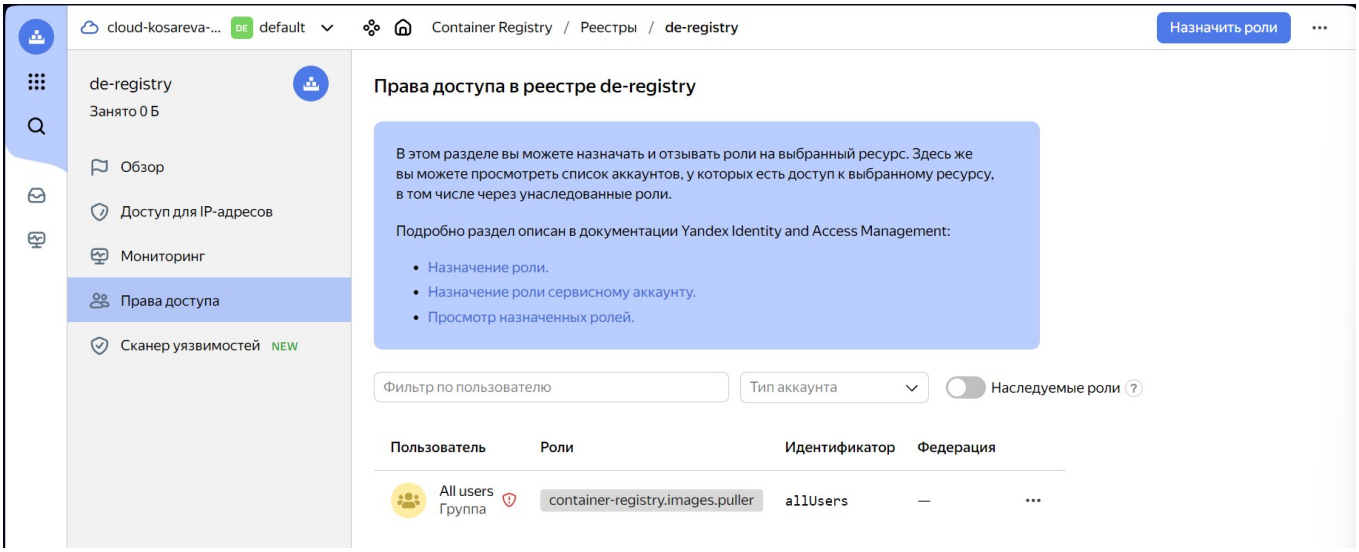
```
CREATE TABLE IF NOT EXISTS dds.s_order_status (
    h_order_pk             UUID      NOT NULL,
    status                 VARCHAR   NOT NULL,
    load_dt                timestamp NOT NULL,
    load_src               VARCHAR   NOT NULL,
    hk_order_status_hashdiff UUID    NOT NULL,
    CONSTRAINT hk_order_status_pk PRIMARY KEY (h_order_pk, load_dt),
    FOREIGN KEY (h_order_pk) REFERENCES dds.h_order(h_order_pk)
);
```

2.2 Итоговая схема слоя DDS



3. Поднимаем Container Registry в Yandex Cloud





4. Собираем шаблон сервиса

```
|   .env
|   docker-compose.yml
|
+---service_cdm
|   |   dockerfile
|   |   requirements.txt
|   |
|   +---app
|   |   |   Chart.yaml
|   |   |   values.yaml
```

```
|   |
|   \---src
|       |   app.py
|       |   app_config.py
|       |
|       +---cdm_loader
|           |   cdm_message_processor_job.py
|           |
|           \---repository
|                   cdm_repository.py
|
+---service_dds
|   |   dockerfile
|   |   requirements.txt
|   |
|   +---app
|   |   |   Chart.yaml
|   |   |   values.yaml
|   |
|   \---src
|       |   app.py
|       |   app_config.py
|       |
|       +---dds_loader
|       |   |   dds_message_processor_job.py
|       |   |
|       |   \---repository
|       |           dds_repository.py
|
\---service_stg
    |   dockerfile
    |   requirements.txt
    |
    +---app
    |   |   Chart.yaml
    |   |   values.yaml
    |
    \---src
        |   app.py
        |   app_config.py
        |
        \---stg_loader
            |   stg_message_processor_job.py
            |
            \---repository
                    stg_repository.py
```

.env

```
KAFKA_HOST="rc1a-93jto5vap3spn3u6.mdb.yandexcloud.net"
KAFKA_PORT="9091"
KAFKA_CONSUMER_USERNAME="producer_consumer"
KAFKA_CONSUMER_PASSWORD="mustdayker_kafka_password"
KAFKA_CONSUMER_GROUP="producer_consumer"
KAFKA_SOURCE_TOPIC="order-service_orders"
KAFKA_STG_SERVICE_ORDERS_TOPIC="stg-service-orders"
KAFKA_DDS_SERVICE_ORDERS_TOPIC="dds-service-orders"

PG_WAREHOUSE_HOST="rc1b-1c2n4dz99tf0temj.mdb.yandexcloud.net"
PG_WAREHOUSE_PORT="6432"
PG_WAREHOUSE_DBNAME="sprint9dwh"
PG_WAREHOUSE_USER="db_user"
PG_WAREHOUSE_PASSWORD="db_user_password"

REDIS_HOST="c-c9qouu38glg6oa5asodd.rw.mdb.yandexcloud.net"
REDIS_PORT="6380"
REDIS_PASSWORD="mustdayker_redis_password"
```

docker-compose.yml

```yaml
version: "3.9"

services:
  stg-service:
    build:
```

```
    context: ./service_stg
    network: host
  image: stg_service:local
  container_name: stg_service_container
  environment:
    FLASK_APP: ${STG_SERVICE_APP_NAME:-stg_service}
    DEBUG: ${STG_SERVICE_DEBUG:-True}

    KAFKA_HOST: ${KAFKA_HOST}
    KAFKA_PORT: ${KAFKA_PORT}
    KAFKA_CONSUMER_USERNAME: ${KAFKA_CONSUMER_USERNAME}
    KAFKA_CONSUMER_PASSWORD: ${KAFKA_CONSUMER_PASSWORD}
    KAFKA_CONSUMER_GROUP: ${KAFKA_CONSUMER_GROUP}
    KAFKA_SOURCE_TOPIC: ${KAFKA_SOURCE_TOPIC}
    KAFKA_DESTINATION_TOPIC: ${KAFKA_STG_SERVICE_ORDERS_TOPIC}

    PG_WAREHOUSE_HOST: ${PG_WAREHOUSE_HOST}
    PG_WAREHOUSE_PORT: ${PG_WAREHOUSE_PORT}
    PG_WAREHOUSE_DBNAME: ${PG_WAREHOUSE_DBNAME}
    PG_WAREHOUSE_USER: ${PG_WAREHOUSE_USER}
    PG_WAREHOUSE_PASSWORD: ${PG_WAREHOUSE_PASSWORD}

    REDIS_HOST: ${REDIS_HOST}
    REDIS_PORT: ${REDIS_PORT}
    REDIS_PASSWORD: ${REDIS_PASSWORD}
  network_mode: "bridge"
  ports:
    - "5011:5000"
  restart: unless-stopped

dds-service:
  build:
    context: ./service_dds
    network: host
  image: dds_service:local
  container_name: dds_service_container
  environment:
    FLASK_APP: ${DDS_APP:-dds_service}
    DEBUG: ${DDS_DEBUG:-True}

    KAFKA_HOST: ${KAFKA_HOST}
    KAFKA_PORT: ${KAFKA_PORT}
    KAFKA_CONSUMER_USERNAME: ${KAFKA_CONSUMER_USERNAME}
    KAFKA_CONSUMER_PASSWORD: ${KAFKA_CONSUMER_PASSWORD}
    KAFKA_CONSUMER_GROUP: ${KAFKA_CONSUMER_GROUP}
    KAFKA_SOURCE_TOPIC: ${KAFKA_STG_SERVICE_ORDERS_TOPIC}
    KAFKA_DESTINATION_TOPIC: ${KAFKA_DDS_SERVICE_ORDERS_TOPIC}

    PG_WAREHOUSE_HOST: ${PG_WAREHOUSE_HOST}
    PG_WAREHOUSE_PORT: ${PG_WAREHOUSE_PORT}
    PG_WAREHOUSE_DBNAME: ${PG_WAREHOUSE_DBNAME}
    PG_WAREHOUSE_USER: ${PG_WAREHOUSE_USER}
    PG_WAREHOUSE_PASSWORD: ${PG_WAREHOUSE_PASSWORD}

  network_mode: "bridge"
  ports:
    - "5012:5000"
  restart: unless-stopped

cdm-service:
  build:
    context: ./service_cdm
    network: host
  image: cdm_service:local
  container_name: cdm_service_container
  environment:
    FLASK_APP: ${CDM_SERVICE_APP:-cdm_service}
    DEBUG: ${CDM_SERVICE_DEBUG:-True}

    KAFKA_HOST: ${KAFKA_HOST}
    KAFKA_PORT: ${KAFKA_PORT}
    KAFKA_CONSUMER_USERNAME: ${KAFKA_CONSUMER_USERNAME}
    KAFKA_CONSUMER_PASSWORD: ${KAFKA_CONSUMER_PASSWORD}
    KAFKA_CONSUMER_GROUP: ${KAFKA_CONSUMER_GROUP}
    KAFKA_SOURCE_TOPIC: ${KAFKA_DDS_SERVICE_ORDERS_TOPIC}
```

```
      PG_WAREHOUSE_HOST: ${PG_WAREHOUSE_HOST}
      PG_WAREHOUSE_PORT: ${PG_WAREHOUSE_PORT}
      PG_WAREHOUSE_DBNAME: ${PG_WAREHOUSE_DBNAME}
      PG_WAREHOUSE_USER: ${PG_WAREHOUSE_USER}
      PG_WAREHOUSE_PASSWORD: ${PG_WAREHOUSE_PASSWORD}

    network_mode: "bridge"
    ports:
      - "5013:5000"
    restart: unless-stopped
```

Для тестирования сервисов используем `Docker Compose`

```
cd c:\GitHub\sprint-9-sample-service

docker compose up -d --build

docker compose down
```

5. Реализация `STG`-сервиса ▲

5.1 Код сервиса `STG` ▲

Проверяем поступающие сообщения из `order-service_orders`

```
docker run -it --name "order" --network=host --rm -v
"c:\GitHub\data_engineer\09_yandex_cloud\cert\.kafka\CA.pem:/data/CA.pem" edenhill/kcat:1.7.1 -b rc1a-
93jto5vap3spn3u6.mdb.yandexcloud.net:9091 -X security.protocol=SASL_SSL -X sasl.mechanisms=SCRAM-SHA-512 -X
sasl.username=producer_consumer -X sasl.password="mustdayker_kafka_password" -X ssl.ca.location=/data/CA.pem
-t order-service_orders -C -o beginning
```

Содержание:


```
{
    "object_id": 3126984,
    "object_type": "order",
    "sent_dttm": "2023-11-25 17:39:51",
    "payload": {
        "restaurant": {"id": "626a81cfefa404208fe9abae"},
        "date": "2023-11-18 17:40:02",
        "user": {"id": "626a81ce9a8cd1920641e272"},
        "order_items": [
            {"id": "6276e8cd0cf48b4cded0086e", "name": "Ролл С ВЕТЧИНОЙ И ОМЛЕТОМ", "price": 120,
"quantity": 3},
            {"id": "6276e8cd0cf48b4cded00879", "name": "Ролл С РЫБОЙ И СОУСОМ ТАРТАР", "price": 180,
"quantity": 5},
            {"id": "6276e8cd0cf48b4cded00875", "name": "КАЛЬЦОНЕ МАРГАРИТА", "price": 120, "quantity": 4},
            {"id": "6276e8cd0cf48b4cded0087a", "name": "НОРВЕЖСКИЙ СЭНДВИЧ", "price": 180, "quantity": 4}
        ],
        "bonus_payment": 0,
        "cost": 2460,
        "payment": 2460,
        "bonus_grant": 0,
        "statuses": [
            {"status": "CLOSED", "dttm": "2023-11-18 17:40:02"},
            {"status": "DELIVERING", "dttm": "2023-11-18 16:56:24"},
            {"status": "COOKING", "dttm": "2023-11-18 16:39:28"},
            {"status": "OPEN", "dttm": "2023-11-18 16:02:44"}
        ],
        "final_status": "CLOSED", "update_ts": "2023-11-18 17:40:02"}
}
```


`stg_message_processor_job.py`

```
import json
from logging import Logger
from typing import List, Dict
from datetime import datetime
```

```python
from lib.kafka_connect import KafkaConsumer, KafkaProducer
from lib.redis import RedisClient
from stg_loader.repository import StgRepository
import time

class StgMessageProcessor:
    def __init__(self,
                 consumer: KafkaConsumer,
                 producer: KafkaProducer,
                 redis_client: RedisClient,
                 stg_repository: StgRepository,
                 batch_size: int,
                 logger: Logger) -> None:
        self._consumer = consumer
        self._producer = producer
        self._redis = redis_client
        self._stg_repository = stg_repository
        self._logger = logger
        self._batch_size = 100



    # функция, которая будет вызываться по расписанию.
    def run(self) -> None:
        # Пишем в лог, что джоб был запущен.
        self._logger.info(f"{datetime.utcnow()}: START")

        for _ in range(self._batch_size):
            msg = self._consumer.consume()
            if not msg:
                break

            self._logger.info(f"{datetime.utcnow()}: Message received")

            order = msg['payload']

            # Загрузка данных в Staging слой PostgreSQL
            self._stg_repository.order_events_insert(
                msg["object_id"],
                msg["object_type"],
                msg["sent_dttm"],
                json.dumps(order))

            user_id = order["user"]["id"]
            user = self._redis.get(user_id)
            user_name = user["name"]
            user_login = user["login"]

            restaurant_id = order['restaurant']['id']
            restaurant = self._redis.get(restaurant_id)
            restaurant_name = restaurant["name"]

            dst_msg = {
                "object_id": msg["object_id"],
                "object_type": "order",
                "payload": {
                    "id": msg["object_id"],
                    "date": order["date"],
                    "cost": order["cost"],
                    "payment": order["payment"],
                    "status": order["final_status"],
                    "restaurant": self._format_restaurant(restaurant_id, restaurant_name),
                    "user": self._format_user(user_id, user_name, user_login),
                    "products": self._format_items(order["order_items"], restaurant)
                }
            }

            self._producer.produce(dst_msg)
            self._logger.info(f"{datetime.utcnow()}. Message Sent")


        # Пишем в лог, что джоб успешно завершен.
        self._logger.info(f"{datetime.utcnow()}: FINISH")

    def _format_restaurant(self, id, name) -> Dict[str, str]:
```

```
        return {
            "id": id,
            "name": name
        }

    def _format_user(self, id, name, login) -> Dict[str, str]:
        return {
            "id": id,
            "name": name,
            "login": login
        }

    def _format_items(self, order_items, restaurant) -> List[Dict[str, str]]:
        items = []

        menu = restaurant["menu"]
        for it in order_items:
            menu_item = next(x for x in menu if x["_id"] == it["id"])
            dst_it = {
                "id": it["id"],
                "price": it["price"],
                "quantity": it["quantity"],
                "name": menu_item["name"],
                "category": menu_item["category"]
            }
            items.append(dst_it)

        return items
```

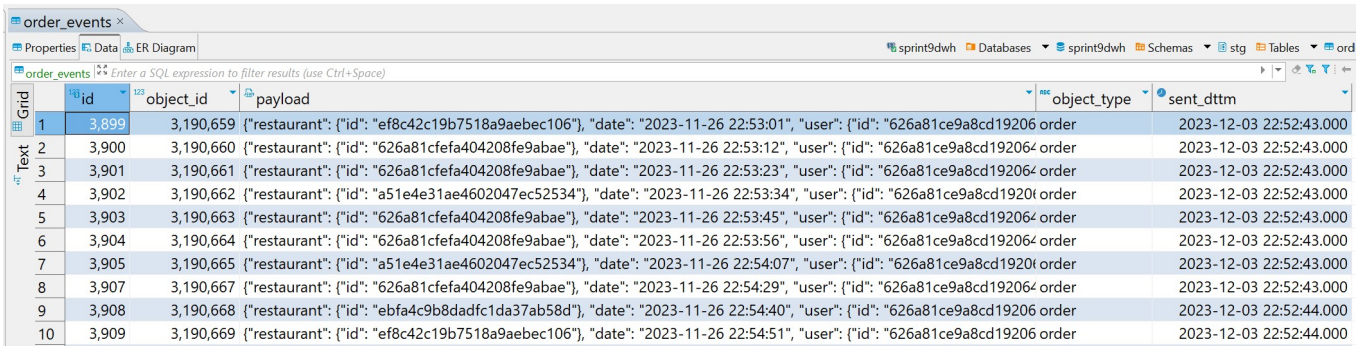
stg_repository.py

```
from datetime import datetime

from lib.pg import PgConnect

class StgRepository:
    def __init__(self, db: PgConnect) -> None:
        self._db = db

    def order_events_insert(self,
                            object_id: int,
                            object_type: str,
                            sent_dttm: datetime,
                            payload: str
                            ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO stg.order_events(
                        object_id,
                        object_type,
                        sent_dttm,
                        payload
                    )
                    VALUES(
                        %(object_id)s,
                        %(object_type)s,
                        %(sent_dttm)s,
                        %(payload)s
                    )
                    ON CONFLICT (object_id) DO UPDATE
                    SET
                        object_type = EXCLUDED.object_type,
                        sent_dttm = EXCLUDED.sent_dttm,
                        payload = EXCLUDED.payload
                    ;
                    """,
                    {
                        'object_id': object_id,
                        'object_type': object_type,
                        'sent_dttm': sent_dttm,
                        'payload': payload
```

```
        }
    )
```

5.2 Проверяем наполнение `STG`

`stg.order_events`



order_events

	id	object_id	payload	object_type	sent_dttm
1	3,899	3,190,659	{"restaurant": {"id": "ef8c42c19b7518a9aebec106"}, "date": "2023-11-26 22:53:01", "user": {"id": "626a81ce9a8cd19206	order	2023-12-03 22:52:43.000
2	3,900	3,190,660	{"restaurant": {"id": "626a81cfefa404208fe9abae"}, "date": "2023-11-26 22:53:12", "user": {"id": "626a81ce9a8cd192064	order	2023-12-03 22:52:43.000
3	3,901	3,190,661	{"restaurant": {"id": "626a81cfefa404208fe9abae"}, "date": "2023-11-26 22:53:23", "user": {"id": "626a81ce9a8cd192064	order	2023-12-03 22:52:43.000
4	3,902	3,190,662	{"restaurant": {"id": "a51e4e31ae4602047ec52534"}, "date": "2023-11-26 22:53:34", "user": {"id": "626a81ce9a8cd1920(	order	2023-12-03 22:52:43.000
5	3,903	3,190,663	{"restaurant": {"id": "626a81cfefa404208fe9abae"}, "date": "2023-11-26 22:53:45", "user": {"id": "626a81ce9a8cd192064	order	2023-12-03 22:52:43.000
6	3,904	3,190,664	{"restaurant": {"id": "626a81cfefa404208fe9abae"}, "date": "2023-11-26 22:53:56", "user": {"id": "626a81ce9a8cd192064	order	2023-12-03 22:52:43.000
7	3,905	3,190,665	{"restaurant": {"id": "a51e4e31ae4602047ec52534"}, "date": "2023-11-26 22:54:07", "user": {"id": "626a81ce9a8cd1920(	order	2023-12-03 22:52:43.000
8	3,907	3,190,667	{"restaurant": {"id": "626a81cfefa404208fe9abae"}, "date": "2023-11-26 22:54:29", "user": {"id": "626a81ce9a8cd192064	order	2023-12-03 22:52:43.000
9	3,908	3,190,668	{"restaurant": {"id": "ebfa4c9b8dadfc1da37ab58d"}, "date": "2023-11-26 22:54:40", "user": {"id": "626a81ce9a8cd19206	order	2023-12-03 22:52:44.000
10	3,909	3,190,669	{"restaurant": {"id": "ef8c42c19b7518a9aebec106"}, "date": "2023-11-26 22:54:51", "user": {"id": "626a81ce9a8cd19206	order	2023-12-03 22:52:44.000

6. Реализация `DDS`-сервиса

6.1 Код сервиса `DDS`

Проверяем поступающие сообщения из `stg-service-orders`

```
docker run -it --name "stg" --network=host --rm -v
"c:\GitHub\data_engineer\09_yandex_cloud\cert\.kafka\CA.pem:/data/CA.pem" edenhill/kcat:1.7.1 -b rc1a-
93jto5vap3spn3u6.mdb.yandexcloud.net:9091 -X security.protocol=SASL_SSL -X sasl.mechanisms=SCRAM-SHA-512 -X
sasl.username=producer_consumer -X sasl.password="mustdayker_kafka_password" -X ssl.ca.location=/data/CA.pem
-t stg-service-orders -C -o beginning
```

Содержание:

```
{
    "object_id": 3127394,
    "object_type": "order",
    "payload": {
        "id": 3127394,
        "date": "2023-11-18 18:55:11",
        "cost": 600,
        "payment": 600,
        "status": "CLOSED",
        "restaurant": {"id": "626a81cfefa404208fe9abae",
                       "name": "Кофейня №1"},
        "user": {"id": "626a81ce9a8cd1920641e2a0",
                 "name": "Исидор Дорофеевич Сысоев",
                 "login": "kolesnikovsilvestr"},
        "products": [
            {"id": "6276e8cd0cf48b4cded00870",
             "price": 120,
             "quantity": 1,
             "name": "РОЛЛ С ИНДЕЙКОЙ",
             "category": "Закуски"},
            {"id": "6276e8cd0cf48b4cded0086e",
             "price": 120,
             "quantity": 4,
             "name": "РОЛЛ С ВЕТЧИНОЙ И ОМЛЕТОМ",
             "category": "Закуски"}
        ]
    }
}
```

`dds_message_processor_job.py`

```python
import json
from logging import Logger
from typing import List, Dict
from datetime import datetime
from uuid import UUID, uuid5
from lib.kafka_connect import KafkaConsumer, KafkaProducer


from dds_loader.repository import DdsRepository
import time

class DdsMessageProcessor:
    def __init__(self,
                 consumer: KafkaConsumer,
                 producer: KafkaProducer,
                 dds_repository: DdsRepository,
                 batch_size: int,
                 logger: Logger
                 ) -> None:
        self._consumer = consumer
        self._producer = producer
        self._dds_repository = dds_repository
        self._logger = logger
        self._batch_size = 30

    def run(self) -> None:
        self._logger.info(f"{datetime.utcnow()}: START")

        # Цикл обработки сообщений
        for _ in range(self._batch_size):
            msg = self._consumer.consume()
            if not msg:
                break

            self._logger.info(f"{datetime.utcnow()}: Message received")

            order = msg['payload']

            load_src_var = "orders-system-kafka"

# __________ ЗАГРУЗКА ДАННЫХ POSTGESQL __________

# _____________________ ХАБЫ _____________________

            # h.1 ______ Загрузка dds.h_user ______
            self._dds_repository.dds_h_user_insert(
                self._uuid_gen(order["user"]["id"]),
                order["user"]["id"],
                datetime.utcnow(),
                load_src_var
                )


            # h.2 ______ Загрузка dds.h_product ______
            for product in order['products']:
                self._dds_repository.dds_h_product_insert(
                    self._uuid_gen(product["id"]),
                    product["id"],
                    datetime.utcnow(),
                    load_src_var
                    )


            # h.3 ______ Загрузка dds.h_category ______
            for product in order['products']:
                self._dds_repository.dds_h_category_insert(
                    self._uuid_gen(product["category"]),
                    product["category"],
                    datetime.utcnow(),
                    load_src_var
                    )

            # h.4 ______ Загрузка dds.h_restaurant ______
            self._dds_repository.dds_h_restaurant_insert(
                self._uuid_gen(order["restaurant"]["id"]),
                order["restaurant"]["id"],
```

```
                datetime.utcnow(),
                load_src_var
                )


            # h.5 _____ Загрузка dds.h_order _____
            self._dds_repository.dds_h_order_insert(
                self._uuid_gen(str(order["id"])),
                order["id"],
                order["date"],
                datetime.utcnow(),
                load_src_var
                )


    # _____________________ ЛИНКИ _____________________

            # l.1 _____ Загрузка dds.l_order_product _____
            for product in order['products']:
                self._dds_repository.dds_l_order_product_insert(
                    self._uuid_gen(f'{order["id"]}{product["id"]}'),
                    self._uuid_gen(str(order["id"])),
                    self._uuid_gen(product["id"]),
                    datetime.utcnow(),
                    load_src_var
                    )

            # l.2 _____ Загрузка dds.l_product_restaurant _____
            for product in order['products']:
                self._dds_repository.dds_l_product_restaurant_insert(
                    self._uuid_gen(f'{product["id"]}{order["restaurant"]["id"]}'),
                    self._uuid_gen(product["id"]),
                    self._uuid_gen(order["restaurant"]["id"]),
                    datetime.utcnow(),
                    load_src_var
                    )


            # l.3 _____ Загрузка dds.l_product_category _____
            for product in order['products']:
                self._dds_repository.dds_l_product_category_insert(
                    self._uuid_gen(f'{product["id"]}{product["category"]}'),
                    self._uuid_gen(product["id"]),
                    self._uuid_gen(product["category"]),
                    datetime.utcnow(),
                    load_src_var
                    )


            # l.4 _____ Загрузка dds.l_order_user _____
            self._dds_repository.dds_l_order_user_insert(
                self._uuid_gen(f'{order["id"]}{order["user"]["id"]}'),
                self._uuid_gen(str(order["id"])),
                self._uuid_gen(order["user"]["id"]),
                datetime.utcnow(),
                load_src_var
                )


    # _____________________ САТЕЛЛИТЫ _____________________

            # s.1 _____ Загрузка dds.s_user_names _____
            self._dds_repository.dds_s_user_names_insert(
                self._uuid_gen(order["user"]["id"]),
                order["user"]["name"],
                order["user"]["login"],
                datetime.utcnow(),
                load_src_var,
                self._uuid_gen(order["user"]["name"])
                )


            # s.2 _____ Загрузка dds.s_product_names _____
            for product in order['products']:
                self._dds_repository.dds_s_product_names_insert(
                    self._uuid_gen(product["id"]),
```

```
                    product["name"],
                    datetime.utcnow(),
                    load_src_var,
                    self._uuid_gen(product["name"])
                    )

            # s.3 ______ Загрузка dds.s_restaurant_names ______
            self._dds_repository.dds_s_restaurant_names_insert(
                self._uuid_gen(order["restaurant"]["id"]),
                order["restaurant"]["name"],
                datetime.utcnow(),
                load_src_var,
                self._uuid_gen(order["restaurant"]["name"])
                )


            # s.4 ______ Загрузка dds.s_order_cost ______
            self._dds_repository.dds_s_order_cost_insert(
                self._uuid_gen(str(order["id"])),
                order["cost"],
                order["payment"],
                datetime.utcnow(),
                load_src_var,
                self._uuid_gen(str(order["cost"]))
                )


            # s.5 ______ Загрузка dds.s_order_status ______
            self._dds_repository.dds_s_order_status_insert(
                self._uuid_gen(str(order["id"])),
                order["status"],
                datetime.utcnow(),
                load_src_var,
                self._uuid_gen(order["status"])
                )


            # Создание сообщения для топика dds-service-orders
            dst_msg = {
                "object_id": order["user"]["id"],
                "object_type": "order",
                "payload": {
                    "user": order['user'],
                    "products": order['products']
                }
            }



            # Отправка сообщения в топик dds-service-orders
            self._producer.produce(dst_msg)

            self._logger.info(f"{datetime.utcnow()}. Message Sent")

        self._logger.info(f"{datetime.utcnow()}: FINISH")

    # Функция для генерации uuid
    def _uuid_gen(self, keygen):
        return uuid5(UUID('7f288a2e-0ad0-4039-8e59-6c9838d87307'), keygen)
```

dds_repository.py

```
import uuid
from uuid import UUID, uuid5
from datetime import datetime
from typing import Any, Dict, List

from lib.pg import PgConnect
from pydantic import BaseModel




class DdsRepository:
    def __init__(self, db: PgConnect) -> None:
```

```
        self._db = db


    # ____________________ ХАБЫ ____________________

    # h.1 ______ Загрузка dds.h_user ______
    def dds_h_user_insert(self,
                          h_user_pk: uuid,
                          user_id: str,
                          load_dt: datetime,
                          load_src: str
                    ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.h_user(
                        h_user_pk,
                        user_id,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(h_user_pk)s,
                        %(user_id)s,
                        %(load_dt)s,
                        %(load_src)s
                    )
                    ON CONFLICT (h_user_pk) DO UPDATE
                    SET
                        load_dt = EXCLUDED.load_dt
                    ;
                    """,
                    {
                        'h_user_pk': h_user_pk,
                        'user_id': user_id,
                        'load_dt': load_dt,
                        'load_src': load_src
                    }
                )

    # h.2 ______ Загрузка dds.h_product ______
    def dds_h_product_insert(self,
                          h_product_pk: uuid,
                          product_id: str,
                          load_dt: datetime,
                          load_src: str
                    ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.h_product(
                        h_product_pk,
                        product_id,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(h_product_pk)s,
                        %(product_id)s,
                        %(load_dt)s,
                        %(load_src)s
                    )
                    ON CONFLICT (h_product_pk) DO UPDATE
                    SET
                        load_dt = EXCLUDED.load_dt
                    ;
                    """,
                    {
                        'h_product_pk': h_product_pk,
                        'product_id': product_id,
                        'load_dt': load_dt,
```

```
                    'load_src': load_src
                }
            )


    # h.3 ______ Загрузка dds.h_category ______
    def dds_h_category_insert(self,
                          h_category_pk: uuid,
                          category_name: str,
                          load_dt: datetime,
                          load_src: str
                ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.h_category(
                        h_category_pk,
                        category_name,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(h_category_pk)s,
                        %(category_name)s,
                        %(load_dt)s,
                        %(load_src)s
                    )
                    ON CONFLICT (h_category_pk) DO UPDATE
                    SET
                        load_dt = EXCLUDED.load_dt
                    ;
                    """,
                    {
                        'h_category_pk': h_category_pk,
                        'category_name': category_name,
                        'load_dt': load_dt,
                        'load_src': load_src
                    }
                )


    # h.4 ______ Загрузка dds.h_restaurant ______
    def dds_h_restaurant_insert(self,
                          h_restaurant_pk: uuid,
                          restaurant_id: str,
                          load_dt: datetime,
                          load_src: str
                ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.h_restaurant(
                        h_restaurant_pk,
                        restaurant_id,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(h_restaurant_pk)s,
                        %(restaurant_id)s,
                        %(load_dt)s,
                        %(load_src)s
                    )
                    ON CONFLICT (h_restaurant_pk) DO UPDATE
                    SET
                        load_dt = EXCLUDED.load_dt
                    ;
                    """,
                    {
                        'h_restaurant_pk': h_restaurant_pk,
                        'restaurant_id': restaurant_id,
                        'load_dt': load_dt,
```

```
                    'load_src': load_src
                }
            )


    # h.5 ______ Загрузка dds.h_order ______
    def dds_h_order_insert(self,
                           h_order_pk: uuid,
                           order_id: int,
                           order_dt: datetime,
                           load_dt: datetime,
                           load_src: str
                ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.h_order(
                        h_order_pk,
                        order_id,
                        order_dt,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(h_order_pk)s,
                        %(order_id)s,
                        %(order_dt)s,
                        %(load_dt)s,
                        %(load_src)s
                    )
                    ON CONFLICT (h_order_pk) DO UPDATE
                    SET
                        order_dt = EXCLUDED.order_dt,
                        load_dt = EXCLUDED.load_dt

                    ;
                    """,
                    {
                        'h_order_pk': h_order_pk,
                        'order_id': order_id,
                        'order_dt': order_dt,
                        'load_dt': load_dt,
                        'load_src': load_src
                    }
                )


# ____________________ ЛИНКИ ____________________


    # l.1 ______ Загрузка dds.l_order_product ______
    def dds_l_order_product_insert(self,
                                   hk_order_product_pk: uuid,
                                   h_order_pk: uuid,
                                   h_product_pk: uuid,
                                   load_dt: datetime,
                                   load_src: str
                                   ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.l_order_product(
                        hk_order_product_pk,
                        h_order_pk,
                        h_product_pk,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(hk_order_product_pk)s,
                        %(h_order_pk)s,
                        %(h_product_pk)s,
```

```
                    %(load_dt)s,
                    %(load_src)s
                )
                ON CONFLICT (hk_order_product_pk) DO UPDATE
                SET
                    load_dt = EXCLUDED.load_dt
                ;
                """,
                {
                    'hk_order_product_pk': hk_order_product_pk,
                    'h_order_pk': h_order_pk,
                    'h_product_pk': h_product_pk,
                    'load_dt': load_dt,
                    'load_src': load_src
                }
            )


    # l.2 ______ Загрузка dds.l_product_restaurant ______
    def dds_l_product_restaurant_insert(self,
                                hk_product_restaurant_pk: uuid,
                                h_product_pk: uuid,
                                h_restaurant_pk: uuid,
                                load_dt: datetime,
                                load_src: str
                                ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.l_product_restaurant(
                        hk_product_restaurant_pk,
                        h_product_pk,
                        h_restaurant_pk,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(hk_product_restaurant_pk)s,
                        %(h_product_pk)s,
                        %(h_restaurant_pk)s,
                        %(load_dt)s,
                        %(load_src)s
                    )
                    ON CONFLICT (hk_product_restaurant_pk) DO UPDATE
                    SET
                        load_dt = EXCLUDED.load_dt
                    ;
                    """,
                    {
                        'hk_product_restaurant_pk': hk_product_restaurant_pk,
                        'h_product_pk': h_product_pk,
                        'h_restaurant_pk': h_restaurant_pk,
                        'load_dt': load_dt,
                        'load_src': load_src
                    }
                )


    # l.3 ______ Загрузка dds.l_product_category ______
    def dds_l_product_category_insert(self,
                                hk_product_category_pk: uuid,
                                h_product_pk: uuid,
                                h_category_pk: uuid,
                                load_dt: datetime,
                                load_src: str
                                ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.l_product_category(
                        hk_product_category_pk,
                        h_product_pk,
```

```
                    h_category_pk,
                    load_dt,
                    load_src
                )
                VALUES(
                    %(hk_product_category_pk)s,
                    %(h_product_pk)s,
                    %(h_category_pk)s,
                    %(load_dt)s,
                    %(load_src)s
                )
                ON CONFLICT (hk_product_category_pk) DO UPDATE
                SET
                    load_dt = EXCLUDED.load_dt
                ;
                """,
                {
                    'hk_product_category_pk': hk_product_category_pk,
                    'h_product_pk': h_product_pk,
                    'h_category_pk': h_category_pk,
                    'load_dt': load_dt,
                    'load_src': load_src
                }
            )



    # l.4 ______ Загрузка dds.l_order_user ______
    def dds_l_order_user_insert(self,
                                hk_order_user_pk: uuid,
                                h_order_pk: uuid,
                                h_user_pk: uuid,
                                load_dt: datetime,
                                load_src: str
                                ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO dds.l_order_user(
                        hk_order_user_pk,
                        h_order_pk,
                        h_user_pk,
                        load_dt,
                        load_src
                    )
                    VALUES(
                        %(hk_order_user_pk)s,
                        %(h_order_pk)s,
                        %(h_user_pk)s,
                        %(load_dt)s,
                        %(load_src)s
                    )
                    ON CONFLICT (hk_order_user_pk) DO UPDATE
                    SET
                        load_dt = EXCLUDED.load_dt
                    ;
                    """,
                    {
                        'hk_order_user_pk': hk_order_user_pk,
                        'h_order_pk': h_order_pk,
                        'h_user_pk': h_user_pk,
                        'load_dt': load_dt,
                        'load_src': load_src
                    }
                )

# ____________________ САТЕЛЛИТЫ ____________________

    # s.1 ______ Загрузка dds.s_user_names ______
    def dds_s_user_names_insert(self,
                                h_user_pk: uuid,
                                username: str,
```

```
                                userlogin: str,
                                load_dt: datetime,
                                load_src: str,
                                hk_user_names_hashdiff: uuid
                                ) -> None:

    with self._db.connection() as conn:
        with conn.cursor() as cur:
            cur.execute(
                """
                INSERT INTO dds.s_user_names(
                    h_user_pk,
                    username,
                    userlogin,
                    load_dt,
                    load_src,
                    hk_user_names_hashdiff
                )
                VALUES(
                    %(h_user_pk)s,
                    %(username)s,
                    %(userlogin)s,
                    %(load_dt)s,
                    %(load_src)s,
                    %(hk_user_names_hashdiff)s
                )
                ON CONFLICT (h_user_pk, load_dt) DO UPDATE
                SET
                    username = EXCLUDED.username,
                    userlogin = EXCLUDED.userlogin,
                    hk_user_names_hashdiff = EXCLUDED.hk_user_names_hashdiff
                ;
                """,
                {
                    'h_user_pk': h_user_pk,
                    'username': username,
                    'userlogin': userlogin,
                    'load_dt': load_dt,
                    'load_src': load_src,
                    'hk_user_names_hashdiff': hk_user_names_hashdiff
                }
            )



# s.2 ______ Загрузка dds.s_product_names ______
def dds_s_product_names_insert(self,
                                h_product_pk: uuid,
                                name: str,
                                load_dt: datetime,
                                load_src: str,
                                hk_product_names_hashdiff: uuid
                                ) -> None:

    with self._db.connection() as conn:
        with conn.cursor() as cur:
            cur.execute(
                """
                INSERT INTO dds.s_product_names(
                    h_product_pk,
                    name,
                    load_dt,
                    load_src,
                    hk_product_names_hashdiff
                )
                VALUES(
                    %(h_product_pk)s,
                    %(name)s,
                    %(load_dt)s,
                    %(load_src)s,
                    %(hk_product_names_hashdiff)s
                )
                ON CONFLICT (h_product_pk, load_dt) DO UPDATE
                SET
                    name = EXCLUDED.name,
                    hk_product_names_hashdiff = EXCLUDED.hk_product_names_hashdiff
```

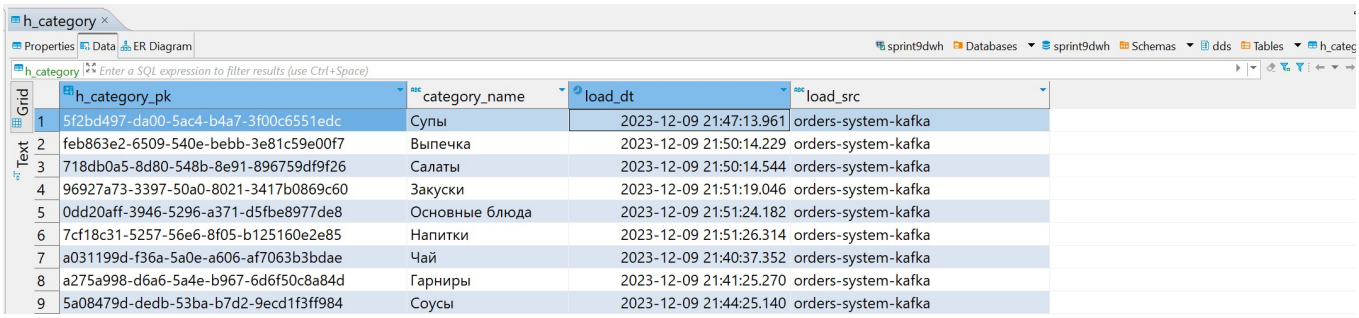
```
                ;
                """,
                {
                    'h_product_pk': h_product_pk,
                    'name': name,
                    'load_dt': load_dt,
                    'load_src': load_src,
                    'hk_product_names_hashdiff': hk_product_names_hashdiff
                }
            )



# s.3 ______ Загрузка dds.s_restaurant_names ______
def dds_s_restaurant_names_insert(self,
                                  h_restaurant_pk: uuid,
                                  name: str,
                                  load_dt: datetime,
                                  load_src: str,
                                  hk_restaurant_names_hashdiff: uuid
                                  ) -> None:

    with self._db.connection() as conn:
        with conn.cursor() as cur:
            cur.execute(
                """
                INSERT INTO dds.s_restaurant_names(
                    h_restaurant_pk,
                    name,
                    load_dt,
                    load_src,
                    hk_restaurant_names_hashdiff
                )
                VALUES(
                    %(h_restaurant_pk)s,
                    %(name)s,
                    %(load_dt)s,
                    %(load_src)s,
                    %(hk_restaurant_names_hashdiff)s
                )
                ON CONFLICT (h_restaurant_pk, load_dt) DO UPDATE
                SET
                    name = EXCLUDED.name,
                    hk_restaurant_names_hashdiff = EXCLUDED.hk_restaurant_names_hashdiff
                ;
                """,
                {
                    'h_restaurant_pk': h_restaurant_pk,
                    'name': name,
                    'load_dt': load_dt,
                    'load_src': load_src,
                    'hk_restaurant_names_hashdiff': hk_restaurant_names_hashdiff
                }
            )



# s.4 ______ Загрузка dds.s_order_cost ______
def dds_s_order_cost_insert(self,
                                  h_order_pk: uuid,
                                  cost: float,
                                  payment: float,
                                  load_dt: datetime,
                                  load_src: str,
                                  hk_order_cost_hashdiff: uuid
                                  ) -> None:

    with self._db.connection() as conn:
        with conn.cursor() as cur:
            cur.execute(
                """
                INSERT INTO dds.s_order_cost(
                    h_order_pk,
```

```
                    cost,
                    payment,
                    load_dt,
                    load_src,
                    hk_order_cost_hashdiff
                )
                VALUES(
                    %(h_order_pk)s,
                    %(cost)s,
                    %(payment)s,
                    %(load_dt)s,
                    %(load_src)s,
                    %(hk_order_cost_hashdiff)s
                )
                ON CONFLICT (h_order_pk, load_dt) DO UPDATE
                SET
                    cost = EXCLUDED.cost,
                    payment = EXCLUDED.payment,
                    hk_order_cost_hashdiff = EXCLUDED.hk_order_cost_hashdiff
                ;
                """,
                {
                    'h_order_pk': h_order_pk,
                    'cost': cost,
                    'payment': payment,
                    'load_dt': load_dt,
                    'load_src': load_src,
                    'hk_order_cost_hashdiff': hk_order_cost_hashdiff
                }
            )



# s.5 ______ Загрузка dds.s_order_status ______
def dds_s_order_status_insert(self,
                              h_order_pk: uuid,
                              status: str,
                              load_dt: datetime,
                              load_src: str,
                              hk_order_status_hashdiff: uuid
                              ) -> None:

    with self._db.connection() as conn:
        with conn.cursor() as cur:
            cur.execute(
                """
                INSERT INTO dds.s_order_status(
                    h_order_pk,
                    status,
                    load_dt,
                    load_src,
                    hk_order_status_hashdiff
                )
                VALUES(
                    %(h_order_pk)s,
                    %(status)s,
                    %(load_dt)s,
                    %(load_src)s,
                    %(hk_order_status_hashdiff)s
                )
                ON CONFLICT (h_order_pk, load_dt) DO UPDATE
                SET
                    status = EXCLUDED.status,
                    hk_order_status_hashdiff = EXCLUDED.hk_order_status_hashdiff
                ;
                """,
                {
                    'h_order_pk': h_order_pk,
                    'status': status,
                    'load_dt': load_dt,
                    'load_src': load_src,
                    'hk_order_status_hashdiff': hk_order_status_hashdiff
                }
            )
```

6.2 Проверяем наполнение DDS

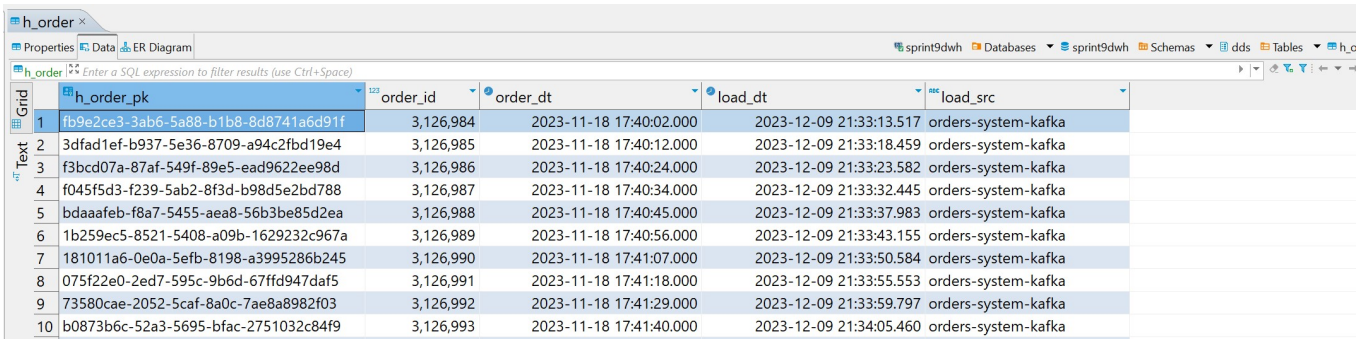
Хабы

dds.h_category



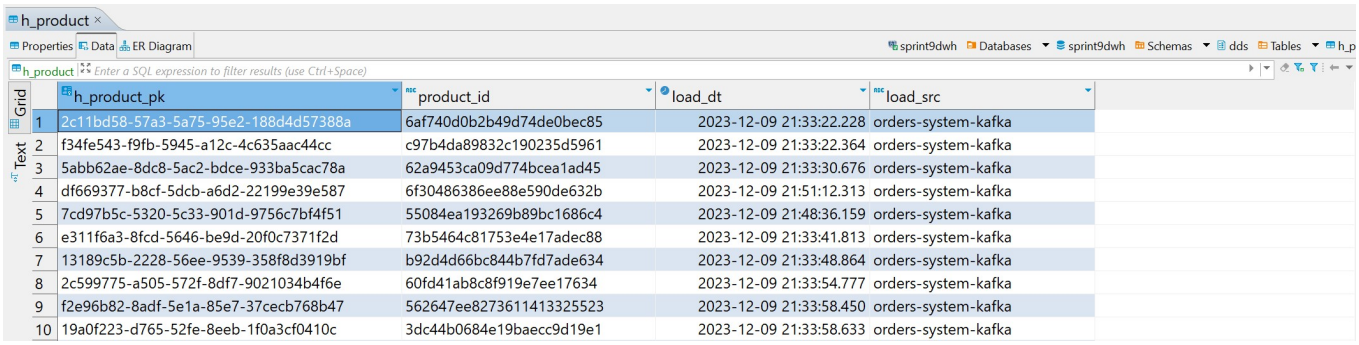
	h_category_pk	category_name	load_dt	load_src
1	5f2bd497-da00-5ac4-b4a7-3f00c6551edc	Супы	2023-12-09 21:47:13.961	orders-system-kafka
2	feb863e2-6509-540e-bebb-3e81c59e00f7	Выпечка	2023-12-09 21:50:14.229	orders-system-kafka
3	718db0a5-8d80-548b-8e91-896759df9f26	Салаты	2023-12-09 21:50:14.544	orders-system-kafka
4	96927a73-3397-50a0-8021-3417b0869c60	Закуски	2023-12-09 21:51:19.046	orders-system-kafka
5	0dd20aff-3946-5296-a371-d5fbe8977de8	Основные блюда	2023-12-09 21:51:24.182	orders-system-kafka
6	7cf18c31-5257-56e6-8f05-b125160e2e85	Напитки	2023-12-09 21:51:26.314	orders-system-kafka
7	a031199d-f36a-5a0e-a606-af7063b3bdae	Чай	2023-12-09 21:40:37.352	orders-system-kafka
8	a275a998-d6a6-5a4e-b967-6d6f50c8a84d	Гарниры	2023-12-09 21:41:25.270	orders-system-kafka
9	5a08479d-dedb-53ba-b7d2-9ecd1f3ff984	Соусы	2023-12-09 21:44:25.140	orders-system-kafka

dds.h_order



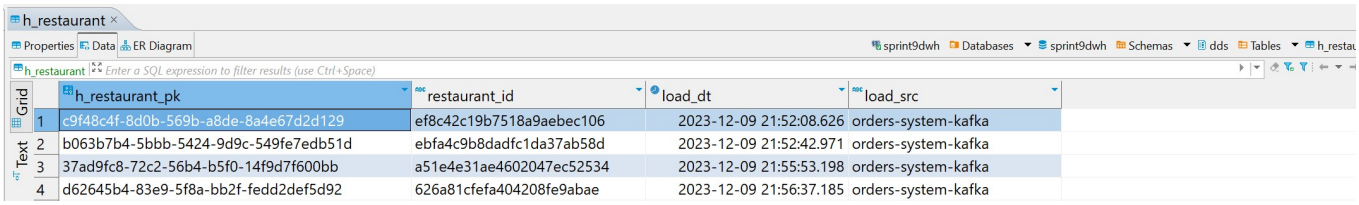
	h_order_pk	order_id	order_dt	load_dt	load_src
1	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	3,126,984	2023-11-18 17:40:02.000	2023-12-09 21:33:13.517	orders-system-kafka
2	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	3,126,985	2023-11-18 17:40:12.000	2023-12-09 21:33:18.459	orders-system-kafka
3	f3bcd07a-87af-549f-89e5-ead9622ee98d	3,126,986	2023-11-18 17:40:24.000	2023-12-09 21:33:23.582	orders-system-kafka
4	f045f5d3-f239-5ab2-8f3d-b98d5e2bd788	3,126,987	2023-11-18 17:40:34.000	2023-12-09 21:33:32.445	orders-system-kafka
5	bdaaafeb-f8a7-5455-aea8-56b3be85d2ea	3,126,988	2023-11-18 17:40:45.000	2023-12-09 21:33:37.983	orders-system-kafka
6	1b259ec5-8521-5408-a09b-1629232c967a	3,126,989	2023-11-18 17:40:56.000	2023-12-09 21:33:43.155	orders-system-kafka
7	181011a6-0e0a-5efb-8198-a3995286b245	3,126,990	2023-11-18 17:41:07.000	2023-12-09 21:33:50.584	orders-system-kafka
8	075f22e0-2ed7-595c-9b6d-67ffd947daf5	3,126,991	2023-11-18 17:41:18.000	2023-12-09 21:33:55.553	orders-system-kafka
9	73580cae-2052-5caf-8a0c-7ae8a8982f03	3,126,992	2023-11-18 17:41:29.000	2023-12-09 21:33:59.797	orders-system-kafka
10	b0873b6c-52a3-5695-bfac-2751032c84f9	3,126,993	2023-11-18 17:41:40.000	2023-12-09 21:34:05.460	orders-system-kafka

dds.h_product



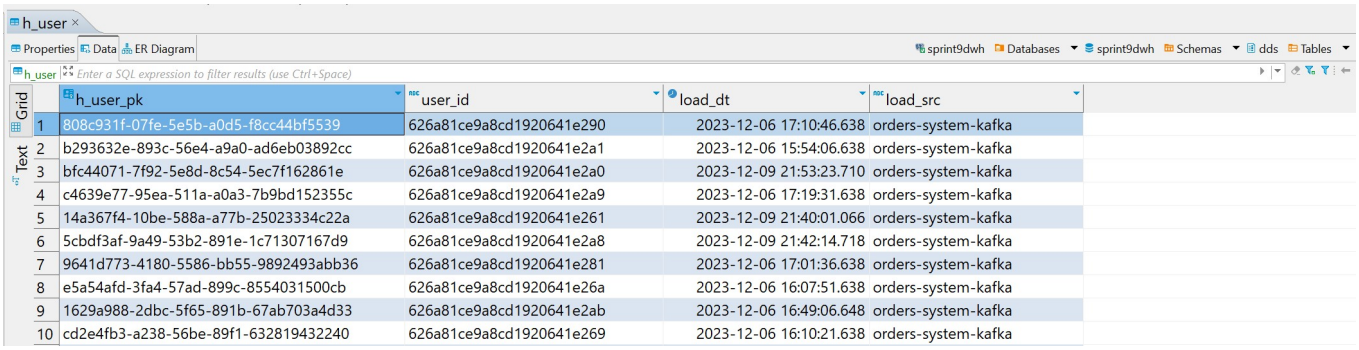
	h_product_pk	product_id	load_dt	load_src
1	2c11bd58-57a3-5a75-95e2-188d4d57388a	6af740d0b2b49d74de0bec85	2023-12-09 21:33:22.228	orders-system-kafka
2	f34fe543-f9fb-5945-a12c-4c635aac44cc	c97b4da89832c190235d5961	2023-12-09 21:33:22.364	orders-system-kafka
3	5abb62ae-8dc8-5ac2-bdce-933ba5cac78a	62a9453ca09d774bcea1ad45	2023-12-09 21:33:30.676	orders-system-kafka
4	df669377-b8cf-5dcb-a6d2-22199e39e587	6f30486386ee88e590de632b	2023-12-09 21:51:12.313	orders-system-kafka
5	7cd97b5c-5320-5c33-901d-9756c7bf4f51	55084ea193269b89bc1686c4	2023-12-09 21:48:36.159	orders-system-kafka
6	e311f6a3-8fcd-5646-be9d-20f0c7371f2d	73b5464c81753e4e17adec88	2023-12-09 21:33:41.813	orders-system-kafka
7	13189c5b-2228-56ee-9539-358f8d3919bf	b92d4d66bc844b7fd7ade634	2023-12-09 21:33:48.864	orders-system-kafka
8	2c599775-a505-572f-8df7-9021034b4f6e	60fd41ab8c8f919e7ee17634	2023-12-09 21:33:54.777	orders-system-kafka
9	f2e96b82-8adf-5e1a-85e7-37cecb768b47	562647ee8273611413325523	2023-12-09 21:33:58.450	orders-system-kafka
10	19a0f223-d765-52fe-8eeb-1f0a3cf0410c	3dc44b0684e19baecc9d19e1	2023-12-09 21:33:58.633	orders-system-kafka

dds.h_restaraurant



	h_restaurant_pk	restaurant_id	load_dt	load_src
1	c9f48c4f-8d0b-569b-a8de-8a4e67d2d129	ef8c42c19b7518a9aebec106	2023-12-09 21:52:08.626	orders-system-kafka
2	b063b7b4-5bbb-5424-9d9c-549fe7edb51d	ebfa4c9b8dadfc1da37ab58d	2023-12-09 21:52:42.971	orders-system-kafka
3	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	a51e4e31ae4602047ec52534	2023-12-09 21:55:53.198	orders-system-kafka
4	d62645b4-83e9-5f8a-bb2f-fedd2def5d92	626a81cfefa404208fe9abae	2023-12-09 21:56:37.185	orders-system-kafka

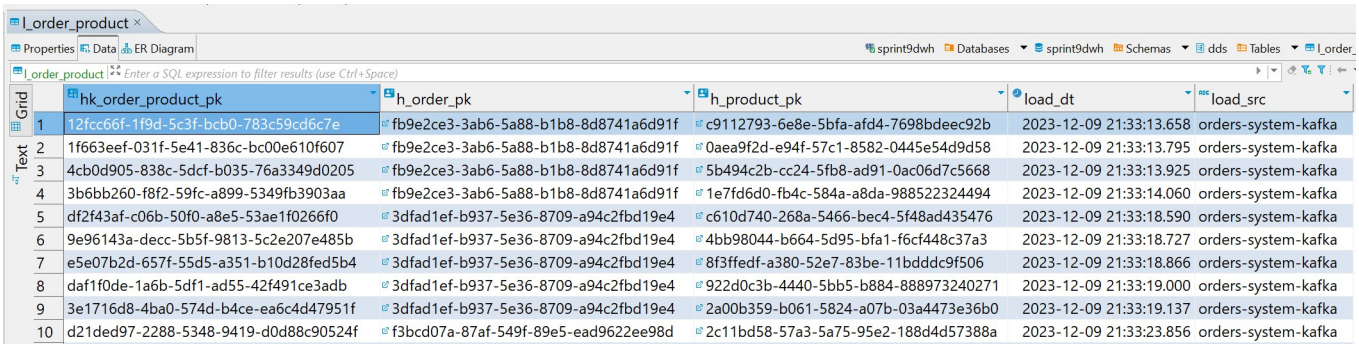
dds.h_user



	h_user_pk	user_id	load_dt	load_src
1	808c931f-07fe-5e5b-a0d5-f8cc44bf5539	626a81ce9a8cd1920641e290	2023-12-06 17:10:46.638	orders-system-kafka
2	b293632e-893c-56e4-a9a0-ad6eb03892cc	626a81ce9a8cd1920641e2a1	2023-12-06 15:54:06.638	orders-system-kafka
3	bfc44071-7f92-5e8d-8c54-5ec7f162861e	626a81ce9a8cd1920641e2a0	2023-12-09 21:53:23.710	orders-system-kafka
4	c4639e77-95ea-511a-a0a3-7b9bd152355c	626a81ce9a8cd1920641e2a9	2023-12-06 17:19:31.638	orders-system-kafka
5	14a367f4-10be-588a-a77b-25023334c22a	626a81ce9a8cd1920641e261	2023-12-09 21:40:01.066	orders-system-kafka
6	5cbdf3af-9a49-53b2-891e-1c71307167d9	626a81ce9a8cd1920641e2a8	2023-12-09 21:42:14.718	orders-system-kafka
7	9641d773-4180-5586-bb55-9892493abb36	626a81ce9a8cd1920641e281	2023-12-06 17:01:36.638	orders-system-kafka
8	e5a54afd-3fa4-57ad-899c-8554031500cb	626a81ce9a8cd1920641e26a	2023-12-06 16:07:51.638	orders-system-kafka
9	1629a988-2dbc-5f65-891b-67ab703a4d33	626a81ce9a8cd1920641e2ab	2023-12-06 16:49:06.648	orders-system-kafka
10	cd2e4fb3-a238-56be-89f1-632819432240	626a81ce9a8cd1920641e269	2023-12-06 16:10:21.638	orders-system-kafka

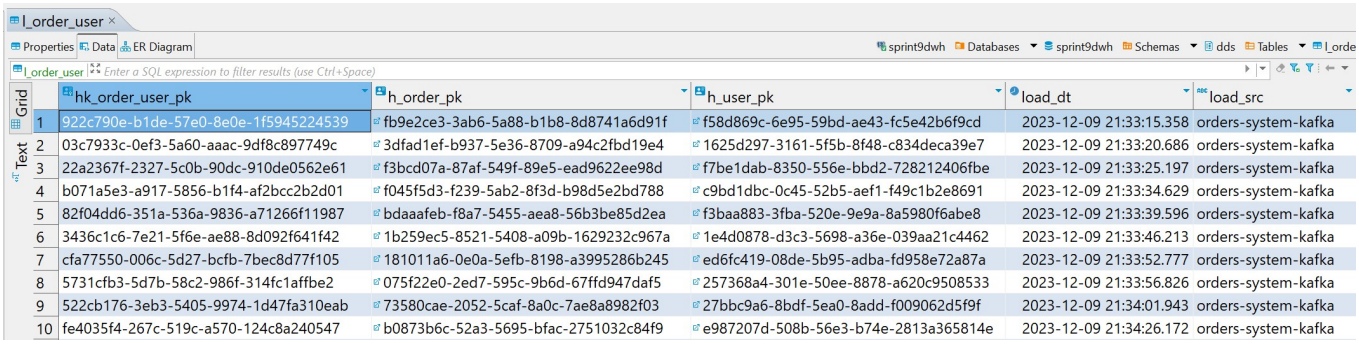
Линки

dds.l_order_product



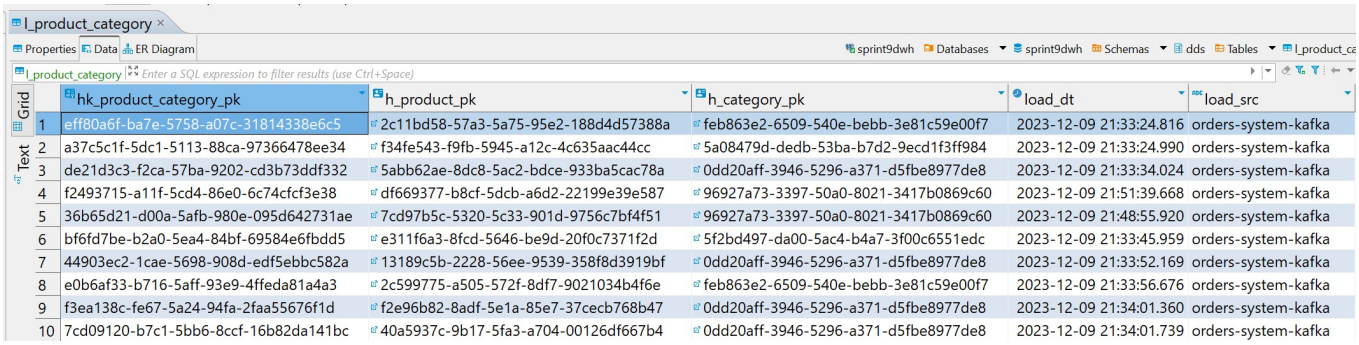
	hk_order_product_pk	h_order_pk	h_product_pk	load_dt	load_src
1	12fcc66f-1f9d-5c3f-bcb0-783c59cd6c7e	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	c9112793-6e8e-5bfa-afd4-7698bdeec92b	2023-12-09 21:33:13.658	orders-system-kafka
2	1f663eef-031f-5e41-836c-bc00e610f607	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	0aea9f2d-e94f-57c1-8582-0445e54d9d58	2023-12-09 21:33:13.795	orders-system-kafka
3	4cb0d905-838c-5dcf-b035-76a3349d0205	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	5b494c2b-cc24-5fb8-ad91-0ac06d7c5668	2023-12-09 21:33:13.925	orders-system-kafka
4	3b6bb260-f8f2-59fc-a899-5349fb3903aa	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	1e7fd6d0-fb4c-584a-a8da-988522324494	2023-12-09 21:33:14.060	orders-system-kafka
5	df2f43af-c06b-50f0-a8e5-53ae1f0266f0	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	c610d740-268a-5466-bec4-5f48ad435476	2023-12-09 21:33:18.590	orders-system-kafka
6	9e96143a-decc-5b5f-9813-5c2e207e485b	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	4bb98044-b664-5d95-bfa1-f6cf448c37a3	2023-12-09 21:33:18.727	orders-system-kafka
7	e5e07b2d-657f-55d5-a351-b10d28fed5b4	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	8f3ffedf-a380-52e7-83be-11bdddc9f506	2023-12-09 21:33:18.866	orders-system-kafka
8	daf1f0de-1a6b-5df1-ad55-42f491ce3adb	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	922d0c3b-4440-5bb5-b884-888973240271	2023-12-09 21:33:19.000	orders-system-kafka
9	3e1716d8-4ba0-574d-b4ce-ea6c4d47951f	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	2a00b359-b061-5824-a07b-03a4473e36b0	2023-12-09 21:33:19.137	orders-system-kafka
10	d21ded97-2288-5348-9419-d0d88c90524f	f3bcd07a-87af-549f-89e5-ead9622ee98d	2c11bd58-57a3-5a75-95e2-188d4d57388a	2023-12-09 21:33:23.856	orders-system-kafka

dds.l_order_user



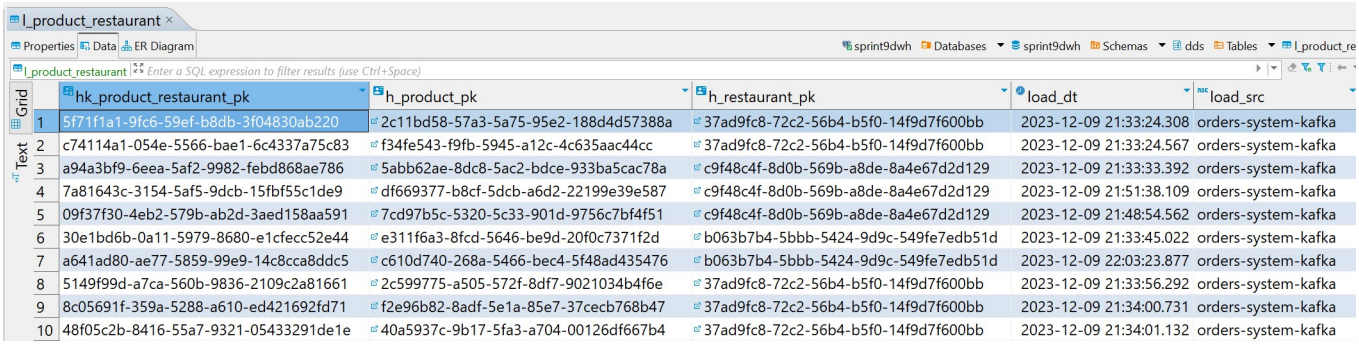
	hk_order_user_pk	h_order_pk	h_user_pk	load_dt	load_src
1	922c790e-b1de-57e0-8e0e-1f5945224539	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	f58d869c-6e95-59bd-ae43-fc5e42b6f9cd	2023-12-09 21:33:15.358	orders-system-kafka
2	03c7933c-0ef3-5a60-aaac-9df8c897749c	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	1625d297-3161-5f5b-8f48-c834deca39e7	2023-12-09 21:33:20.686	orders-system-kafka
3	22a2367f-2327-5c0b-90dc-910de0562e61	f3bcd07a-87af-549f-89e5-ead9622ee98d	f7be1dab-8350-556e-bbd2-728212406fbe	2023-12-09 21:33:25.197	orders-system-kafka
4	b071a5e3-a917-5856-b1f4-af2bcc2b2d01	f045f5d3-f239-5ab2-8f3d-b98d5e2bd788	c9bd1dbc-0c45-52b5-aef1-f49c1b2e8691	2023-12-09 21:33:34.629	orders-system-kafka
5	82f04dd6-351a-536a-9836-a71266f11987	bdaaafeb-f8a7-5455-aea8-56b3be85d2ea	f3baa883-3fba-520e-9e9a-8a5980f6abe8	2023-12-09 21:33:39.596	orders-system-kafka
6	3436c1c6-7e21-5f6e-ae88-8d092f641f42	1b259ec5-8521-5408-a09b-1629232c967a	1e4d0878-d3c3-5698-a36e-039aa21c4462	2023-12-09 21:33:46.213	orders-system-kafka
7	cfa77550-006c-5d27-bcfb-7bec8d77f105	181011a6-0e0a-5efb-8198-a3995286b245	ed6fc419-08de-5b95-adba-fd958e72a87a	2023-12-09 21:33:52.777	orders-system-kafka
8	5731cfb3-5d7b-58c2-986f-314fc1affbe2	075f22e0-2ed7-595c-9b6d-67ffd947daf5	257368a4-301e-50ee-8878-a620c9508533	2023-12-09 21:33:56.826	orders-system-kafka
9	522cb176-3eb3-5405-9974-1d47fa310eab	73580cae-2052-5caf-8a0c-7ae8a8982f03	27bbc9a6-8bdf-5ea0-8add-f009062d5f9f	2023-12-09 21:34:01.943	orders-system-kafka
10	fe4035f4-267c-519c-a570-124c8a240547	b0873b6c-52a3-5695-bfac-2751032c84f9	e987207d-508b-56e3-b74e-2813a365814e	2023-12-09 21:34:26.172	orders-system-kafka

dds.l_product_category



	hk_product_category_pk	h_product_pk	h_category_pk	load_dt	load_src
1	eff80a6f-ba7e-5758-a07c-31814338e6c5	2c11bd58-57a3-5a75-95e2-188d4d57388a	feb863e2-6509-540e-bebb-3e81c59e00f7	2023-12-09 21:33:24.816	orders-system-kafka
2	a37c5c1f-5dc1-5113-88ca-97366478ee34	f34fe543-f9fb-5945-a12c-4c635aac44cc	5a08479d-dedb-53ba-b7d2-9ecd1f3ff984	2023-12-09 21:33:24.990	orders-system-kafka
3	de21d3c3-f2ca-57ba-9202-cd3b73ddf332	5abb62ae-8dc8-5ac2-bdce-933ba5cac78a	0dd20aff-3946-5296-a371-d5fbe8977de8	2023-12-09 21:33:34.024	orders-system-kafka
4	f2493715-a11f-5cd4-86e0-6c74cfcf3e38	df669377-b8cf-5dcb-a6d2-22199e39e587	96927a73-3397-50a0-8021-3417b0869c60	2023-12-09 21:51:39.668	orders-system-kafka
5	36b65d21-d00a-5afb-980e-095d642731ae	7cd97b5c-5320-5c33-901d-9756c7bf4f51	96927a73-3397-50a0-8021-3417b0869c60	2023-12-09 21:48:55.920	orders-system-kafka
6	bf6fd7be-b2a0-5ea4-84bf-69584e6fbdd5	e311f6a3-8fcd-5646-be9d-20f0c7371f2d	5f2bd497-da00-5ac4-b4a7-3f00c6551edc	2023-12-09 21:33:45.959	orders-system-kafka
7	44903ec2-1cae-5698-908d-edf5ebbc582a	13189c5b-2228-56ee-9539-358f8d3919bf	0dd20aff-3946-5296-a371-d5fbe8977de8	2023-12-09 21:33:52.169	orders-system-kafka
8	e0b6af33-b716-5aff-93e9-4ffeda81a4a3	2c599775-a505-572f-8df7-9021034b4f6e	feb863e2-6509-540e-bebb-3e81c59e00f7	2023-12-09 21:33:56.676	orders-system-kafka
9	f3ea138c-fe67-5a24-94fa-2faa55676f1d	f2e96b82-8adf-5e1a-85e7-37cecb768b47	0dd20aff-3946-5296-a371-d5fbe8977de8	2023-12-09 21:34:01.360	orders-system-kafka
10	7cd09120-b7c1-5bb6-8ccf-16b82da141bc	40a5937c-9b17-5fa3-a704-00126df667b4	0dd20aff-3946-5296-a371-d5fbe8977de8	2023-12-09 21:34:01.739	orders-system-kafka

dds.l_product_restaurant



	hk_product_restaurant_pk	h_product_pk	h_restaurant_pk	load_dt	load_src
1	5f71f1a1-9fc6-59ef-b8db-3f04830ab220	2c11bd58-57a3-5a75-95e2-188d4d57388a	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	2023-12-09 21:33:24.308	orders-system-kafka
2	c74114a1-054e-5566-bae1-6c4337a75c83	f34fe543-f9fb-5945-a12c-4c635aac44cc	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	2023-12-09 21:33:24.567	orders-system-kafka
3	a94a3bf9-6eea-5af2-9982-febd868ae786	5abb62ae-8dc8-5ac2-bdce-933ba5cac78a	c9f48c4f-8d0b-569b-a8de-8a4e67d2d129	2023-12-09 21:33:33.392	orders-system-kafka
4	7a81643c-3154-5af5-9dcb-15fbf55c1de9	df669377-b8cf-5dcb-a6d2-22199e39e587	c9f48c4f-8d0b-569b-a8de-8a4e67d2d129	2023-12-09 21:51:38.109	orders-system-kafka
5	09f37f30-4eb2-579b-ab2d-3aed158aa591	7cd97b5c-5320-5c33-901d-9756c7bf4f51	c9f48c4f-8d0b-569b-a8de-8a4e67d2d129	2023-12-09 21:48:54.562	orders-system-kafka
6	30e1bd6b-0a11-5979-8680-e1cfecc52e44	e311f6a3-8fcd-5646-be9d-20f0c7371f2d	b063b7b4-5bbb-5424-9d9c-549fe7edb51d	2023-12-09 21:33:45.022	orders-system-kafka
7	a641ad80-ae77-5859-99e9-14c8cca8ddc5	c610d740-268a-5466-bec4-5f48ad435476	b063b7b4-5bbb-5424-9d9c-549fe7edb51d	2023-12-09 22:03:23.877	orders-system-kafka
8	5149f99d-a7ca-560b-9836-2109c2a81661	2c599775-a505-572f-8df7-9021034b4f6e	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	2023-12-09 21:33:56.292	orders-system-kafka
9	8c05691f-359a-5288-a610-ed421692fd71	f2e96b82-8adf-5e1a-85e7-37cecb768b47	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	2023-12-09 21:34:00.731	orders-system-kafka
10	48f05c2b-8416-55a7-9321-05433291de1e	40a5937c-9b17-5fa3-a704-00126df667b4	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	2023-12-09 21:34:01.132	orders-system-kafka

Сателлиты

dds.s_order_cost

s_order_cost

	h_order_pk	cost	payment	load_dt	load_src	hk_order_cost_hashdiff
1	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	2,460	2,460	2023-12-09 21:33:16.372	orders-system-kafka	7648cb45-00fe-538e-9ff1-ae674b11aa93
2	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	6,050	6,050	2023-12-09 21:33:21.806	orders-system-kafka	95a53f25-f43e-5ea8-8769-b7ce65794d50
3	f3bcd07a-87af-549f-89e5-ead9622ee98d	1,370	1,370	2023-12-09 21:33:29.119	orders-system-kafka	dd5a3366-e6f5-5dfc-9d28-26b6670cbb04
4	f045f5d3-f239-5ab2-8f3d-b98d5e2bd788	2,147	2,147	2023-12-09 21:33:36.129	orders-system-kafka	ff3688e6-831a-526d-9e06-5ef41162e13c
5	bdaaafeb-f8a7-5455-aea8-56b3be85d2ea	720	720	2023-12-09 21:33:40.482	orders-system-kafka	22a4079f-5118-5f0c-bd54-a7c341bcb96d
6	1b259ec5-8521-5408-a09b-1629232c967a	6,760	6,760	2023-12-09 21:33:48.036	orders-system-kafka	00cf40f5-68a9-559a-978c-8f20f68ac9a2
7	181011a6-0e0a-5efb-8198-a3995286b245	3,150	3,150	2023-12-09 21:33:53.935	orders-system-kafka	2f1ee3e8-7e40-58cd-8296-a8850c68db1d
8	075f22e0-2ed7-595c-9b6d-67ffd947daf5	3,370	3,370	2023-12-09 21:33:57.850	orders-system-kafka	bf7f1ed2-bd0e-5dfb-aae8-4b26d9d38d6f
9	73580cae-2052-5caf-8a0c-7ae8a8982f03	2,670	2,670	2023-12-09 21:34:03.245	orders-system-kafka	32d7425b-79ea-5008-a52f-d376b17ccf0b
10	b0873b6c-52a3-5695-bfac-2751032c84f9	1,920	1,920	2023-12-09 21:34:38.263	orders-system-kafka	31de6f44-8d70-546b-886f-15acd970a241

dds.s_order_status

s_order_status

	h_order_pk	status	load_dt	load_src	hk_order_status_hashdiff
1	fb9e2ce3-3ab6-5a88-b1b8-8d8741a6d91f	CLOSED	2023-12-09 21:33:16.532	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
2	3dfad1ef-b937-5e36-8709-a94c2fbd19e4	CLOSED	2023-12-09 21:33:21.937	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
3	f3bcd07a-87af-549f-89e5-ead9622ee98d	CLOSED	2023-12-09 21:33:29.504	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
4	f045f5d3-f239-5ab2-8f3d-b98d5e2bd788	CLOSED	2023-12-09 21:33:36.420	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
5	bdaaafeb-f8a7-5455-aea8-56b3be85d2ea	CLOSED	2023-12-09 21:33:40.671	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
6	1b259ec5-8521-5408-a09b-1629232c967a	CLOSED	2023-12-09 21:33:48.286	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
7	181011a6-0e0a-5efb-8198-a3995286b245	CLOSED	2023-12-09 21:33:54.153	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
8	075f22e0-2ed7-595c-9b6d-67ffd947daf5	CANCELLED	2023-12-09 21:33:58.009	orders-system-kafka	19ea1da0-a52a-5d78-bed0-b187d95d3d34
9	73580cae-2052-5caf-8a0c-7ae8a8982f03	CLOSED	2023-12-09 21:34:03.458	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe
10	b0873b6c-52a3-5695-bfac-2751032c84f9	CLOSED	2023-12-09 21:34:39.205	orders-system-kafka	62e0268c-cb65-506c-bdc5-c40400a00dfe

dds.s_product_names

s_product_names

	h_product_pk	name	load_dt	load_src	hk_product_names_hashdiff
1	c9112793-6e8e-5bfa-afd4-7698bdeec92b	РОЛЛ С ВЕТЧИНОЙ И ОМЛЕТОМ	2023-12-09 21:33:15.651	orders-system-kafka	c7eed645-16dc-5683-a29a-7cf8a73095b1
2	0aea9f2d-e94f-57c1-8582-0445e54d9d58	РОЛЛ С РЫБОЙ И СОУСОМ ТАРТАР	2023-12-09 21:33:15.807	orders-system-kafka	6f3d4b46-c42d-5c36-93fe-67f2ed96e291
3	5b494c2b-cc24-5fb8-ad91-0ac06d7c5668	КАЛЬЦОНЕ МАРГАРИТА	2023-12-09 21:33:15.951	orders-system-kafka	5c19b655-6c57-5d98-9160-71ce3a3e9cfb
4	1e7fd6d0-fb4c-584a-a8da-988522324494	НОРВЕЖСКИЙ СЭНДВИЧ С РЫБОЙ И ЯЙЦОМ	2023-12-09 21:33:16.099	orders-system-kafka	0b9fede1-386b-5c34-8e54-dd160e6a7a8b
5	c610d740-268a-5466-bec4-5f48ad435476	Манты с бараниной 3шт	2023-12-09 21:33:20.975	orders-system-kafka	21f8bb26-5fc5-5e2a-84dd-b9e075b09a2d
6	4bb98044-b664-5d95-bfa1-f6cf448c37a3	Казан кебаб с говядиной	2023-12-09 21:33:21.152	orders-system-kafka	6dae8f28-724f-5bb3-99ae-cb11fd480bd1
7	8f3ffedf-a380-52e7-83be-11bdddc9f506	Манты с говядиной 3шт	2023-12-09 21:33:21.286	orders-system-kafka	aa3e9ee7-264c-5c48-b551-2a6d749ba750
8	922d0c3b-4440-5bb5-b884-888973240271	Лагман	2023-12-09 21:33:21.419	orders-system-kafka	65188369-2ada-58cd-9d69-ffcdef69007c
9	2a00b359-b061-5824-a07b-03a4473e36b0	Шурпа из говядины	2023-12-09 21:33:21.550	orders-system-kafka	18a4ec43-a114-5932-88a7-3e2a6f6c81af
10	2c11bd58-57a3-5a75-95e2-188d4d57388a	Хачапури по-аджарски	2023-12-09 21:33:28.432	orders-system-kafka	809d1f72-c4e9-5077-b391-339be0a8fdeb

dds.s_restaurant_names

s_restaurant_names

	h_restaurant_pk	name	load_dt	load_src	hk_restaurant_names_hashdiff
1	d62645b4-83e9-5f8a-bb2f-fedd2def5d92	Кофейня №1	2023-12-09 21:33:16.239	orders-system-kafka	0a6852f8-0ca3-5786-968e-77b8a020aa50
2	b063b7b4-5bbb-5424-9d9c-549fe7edb51d	PLove	2023-12-09 21:33:21.679	orders-system-kafka	f018cd46-bada-5215-9cf9-d0aba226604f
3	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	Кубдари	2023-12-09 21:33:28.888	orders-system-kafka	33b04147-ba55-561b-98af-444cac9f0562
4	c9f48c4f-8d0b-569b-a8de-8a4e67d2d129	Вкус Индии	2023-12-09 21:33:35.850	orders-system-kafka	1747d512-39df-5873-bbc1-ab2ec38f813f
5	d62645b4-83e9-5f8a-bb2f-fedd2def5d92	Кофейня №1	2023-12-09 21:33:40.309	orders-system-kafka	0a6852f8-0ca3-5786-968e-77b8a020aa50
6	b063b7b4-5bbb-5424-9d9c-549fe7edb51d	PLove	2023-12-09 21:33:47.759	orders-system-kafka	f018cd46-bada-5215-9cf9-d0aba226604f
7	b063b7b4-5bbb-5424-9d9c-549fe7edb51d	PLove	2023-12-09 21:33:53.702	orders-system-kafka	f018cd46-bada-5215-9cf9-d0aba226604f
8	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	Кубдари	2023-12-09 21:33:57.638	orders-system-kafka	33b04147-ba55-561b-98af-444cac9f0562
9	37ad9fc8-72c2-56b4-b5f0-14f9d7f600bb	Кубдари	2023-12-09 21:34:03.049	orders-system-kafka	33b04147-ba55-561b-98af-444cac9f0562
10	d62645b4-83e9-5f8a-bb2f-fedd2def5d92	Кофейня №1	2023-12-09 21:34:33.129	orders-system-kafka	0a6852f8-0ca3-5786-968e-77b8a020aa50

dds.s_user_names

s_user_names

	h_user_pk	username	userlogin	load_dt	load_src	hk_user_names_hashdiff
1	f58d869c-6e95-59bd-ae43-fc5e42b6f9cd	Рябов Федосий Терентьевич	budimir_63	2023-12-09 21:33:15.518	orders-system-kafka	6f77082d-b16f-5327-8d01-391f9c63f
2	1625d297-3161-5f5b-8f48-c834deca39e7	Селиверстов Никанор Адамович	trofim_66	2023-12-09 21:33:20.842	orders-system-kafka	4357c3b8-a5c1-5a51-9073-90122a8c
3	f7be1dab-8350-556e-bbd2-728212406fbe	Матвеева Марина Оскаровна	burovmitofan	2023-12-09 21:33:28.252	orders-system-kafka	0de6be1b-ae68-52b5-beca-7bf1780a
4	c9bd1dbc-0c45-52b5-aef1-f49c1b2e8691	Клавдия Викторовна Дмитриева	afanasilobanov	2023-12-09 21:33:34.922	orders-system-kafka	2fb393b6-854c-55e6-b571-795cb118
5	f3baa883-3fba-520e-9e9a-8a5980f6abe8	Светлана Валериевна Корнилова	zribakov	2023-12-09 21:33:39.797	orders-system-kafka	cf6359d8-8776-568e-9d63-2f4c3d73
6	1e4d0878-d3c3-5698-a36e-039aa21c4462	Елена Игоревна Ефимова	dobromisl_2009	2023-12-09 21:33:46.456	orders-system-kafka	1522d509-432b-51d5-9afd-08b73b7f
7	ed6fc419-08de-5b95-adba-fd958e72a87a	Комиссарова Акулина Кузьминична	ukononova	2023-12-09 21:33:53.076	orders-system-kafka	b055c95f-1e05-513a-84b4-f07a7088
8	257368a4-301e-50ee-8878-a620c9508533	Гаврила Архипович Игнатьев	gavaii_2222	2023-12-09 21:33:57.024	orders-system-kafka	edb95e73-7960-5d73-911e-68997b3
9	27bbc9a6-8bdf-5ea0-8add-f009062d5f9f	Мина Антонович Моисеев	sergeevaglafira	2023-12-09 21:34:02.180	orders-system-kafka	28793fec-42f7-5502-a6d3-8fefaaf008
10	e987207d-508b-56e3-b74e-2813a365814e	Воронцова Агафья Валентиновна	gorbunovajulija	2023-12-09 21:34:26.344	orders-system-kafka	b59e96e5-6c5a-5fae-9e83-0a474fd9c

7. Реализация CDM-сервиса ▲

7.1 Код сервиса CDM ▲

Проверяем поступающие сообщения из dds-service-orders

```
docker run -it --name "dds" --network=host --rm -v
"c:\GitHub\data_engineer\09_yandex_cloud\cert\.kafka\CA.pem:/data/CA.pem" edenhill/kcat:1.7.1 -b rc1a-
93jto5vap3spn3u6.mdb.yandexcloud.net:9091 -X security.protocol=SASL_SSL -X sasl.mechanisms=SCRAM-SHA-512 -X
sasl.username=producer_consumer -X sasl.password="mustdayker_kafka_password" -X ssl.ca.location=/data/CA.pem
-t dds-service-orders -C -o beginning
```

Содержание:


```
{
    "object_id": "626a81ce9a8cd1920641e294",
    "object_type": "order",
    "payload": {
        "user": {"id": "626a81ce9a8cd1920641e294",
                 "name": "Воронцова Агафья Валентиновна",
                 "login": "gorbunovajulija"},
        "products": [
            {"id": "6276e8cd0cf48b4cded0087b",
             "price": 180,
             "quantity": 4,
             "name": "Сэндвич с беконом и огурцом",
             "category": "выпечка"},
            {"id": "6276e8cd0cf48b4cded00870",
             "price": 120,
             "quantity": 1,
             "name": "Ролл С Индейкой",
             "category": "закуски"},
            {"id": "6276e8cd0cf48b4cded0086e",
             "price": 120,
             "quantity": 5,
             "name": "Ролл с ветчиной и омлетом",
             "category": "закуски"},
            {"id": "6276e8cd0cf48b4cded00872",
             "price": 120,
             "quantity": 4,
             "name": "Сэндвич с ветчиной и сыром",
             "category": "выпечка"}
        ]
    }
}
```


cdm_message_processor_job.py

```
import json
from logging import Logger
from typing import List, Dict
from datetime import datetime
from uuid import UUID, uuid5
from lib.kafka_connect import KafkaConsumer

from cdm_loader.repository import CdmRepository
import time

class CdmMessageProcessor:
    def __init__(self,
                 consumer: KafkaConsumer,
                 cdm_repository: CdmRepository,
                 batch_size: int,
                 logger: Logger
                 ) -> None:
        self._consumer = consumer
        self._cdm_repository = cdm_repository
        self._logger = logger
        self._batch_size = 30

    def run(self) -> None:
```

```
        self._logger.info(f"{datetime.utcnow()}: START")

        for _ in range(self._batch_size):
            msg = self._consumer.consume()
            if not msg:
                break

            self._logger.info(f"{datetime.utcnow()}: Message received")

            order = msg['payload']

            # ______ Загрузка cdm.user_product_counters ______
            for product in order['products']:
                self._cdm_repository.cdm_user_product_counters_insert(
                    self._uuid_gen(order["user"]["id"]),
                    self._uuid_gen(product["id"]),
                    product["name"],
                    product["quantity"]
                    )

            # ______ Загрузка cdm.user_category_counters ______
            for product in order['products']:
                self._cdm_repository.cdm_user_category_counters_insert(
                    self._uuid_gen(order["user"]["id"]),
                    self._uuid_gen(product["category"]),
                    product["category"],
                    product["quantity"]
                    )


            self._logger.info(f"{datetime.utcnow()}. Message Sent")

        self._logger.info(f"{datetime.utcnow()}: FINISH")

    # Функция для генерации uuid
    def _uuid_gen(self, keygen):
        return uuid5(UUID('7f288a2e-0ad0-4039-8e59-6c9838d87307'), keygen)
```

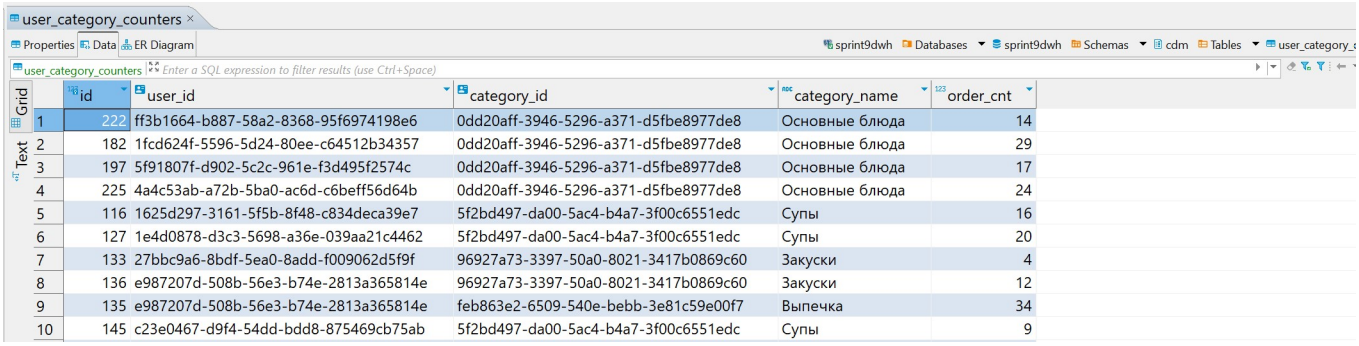
cdm_repository.py

```
import uuid
from datetime import datetime
from typing import Any, Dict, List

from lib.pg import PgConnect
from pydantic import BaseModel




class CdmRepository:
    def __init__(self, db: PgConnect) -> None:
        self._db = db



    # ______ Загрузка cdm.user_product_counters ______
    def cdm_user_product_counters_insert(self,
                                         user_id: uuid,
                                         product_id: uuid,
                                         product_name: str,
                                         order_cnt: int
                                         ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO cdm.user_product_counters(
                        user_id,
                        product_id,
                        product_name,
                        order_cnt
                    )
                    VALUES(
```

```
                    %(user_id)s,
                    %(product_id)s,
                    %(product_name)s,
                    %(order_cnt)s
                )
                ON CONFLICT (user_id, product_id) DO UPDATE
                SET
                    order_cnt = user_product_counters.order_cnt + EXCLUDED.order_cnt
                ;
                """,
                {
                    'user_id': user_id,
                    'product_id': product_id,
                    'product_name': product_name,
                    'order_cnt': order_cnt
                }
            )


    # ______ Загрузка cdm.user_category_counters ______
    def cdm_user_category_counters_insert(self,
                                          user_id: uuid,
                                          category_id: uuid,
                                          category_name: str,
                                          order_cnt: int
                                          ) -> None:

        with self._db.connection() as conn:
            with conn.cursor() as cur:
                cur.execute(
                    """
                    INSERT INTO cdm.user_category_counters(
                        user_id,
                        category_id,
                        category_name,
                        order_cnt
                    )
                    VALUES(
                        %(user_id)s,
                        %(category_id)s,
                        %(category_name)s,
                        %(order_cnt)s
                    )
                    ON CONFLICT (user_id, category_id) DO UPDATE
                    SET
                        order_cnt = user_category_counters.order_cnt + EXCLUDED.order_cnt
                    ;
                    """,
                    {
                        'user_id': user_id,
                        'category_id': category_id,
                        'category_name': category_name,
                        'order_cnt': order_cnt
                    }
                )
```

7.2 Проверяем наполнение `CDM` ▲

`cdm.user_category_counters`



user_category_counters

	id	user_id	category_id	category_name	order_cnt
1	222	ff3b1664-b887-58a2-8368-95f6974198e6	0dd20aff-3946-5296-a371-d5fbe8977de8	Основные блюда	14
2	182	1fcd624f-5596-5d24-80ee-c64512b34357	0dd20aff-3946-5296-a371-d5fbe8977de8	Основные блюда	29
3	197	5f91807f-d902-5c2c-961e-f3d495f2574c	0dd20aff-3946-5296-a371-d5fbe8977de8	Основные блюда	17
4	225	4a4c53ab-a72b-5ba0-ac6d-c6beff56d64b	0dd20aff-3946-5296-a371-d5fbe8977de8	Основные блюда	24
5	116	1625d297-3161-5f5b-8f48-c834deca39e7	5f2bd497-da00-5ac4-b4a7-3f00c6551edc	Супы	16
6	127	1e4d0878-d3c3-5698-a36e-039aa21c4462	5f2bd497-da00-5ac4-b4a7-3f00c6551edc	Супы	20
7	133	27bbc9a6-8bdf-5ea0-8add-f009062d5f9f	96927a73-3397-50a0-8021-3417b0869c60	Закуски	4
8	136	e987207d-508b-56e3-b74e-2813a365814e	96927a73-3397-50a0-8021-3417b0869c60	Закуски	12
9	135	e987207d-508b-56e3-b74e-2813a365814e	feb863e2-6509-540e-bebb-3e81c59e00f7	Выпечка	34
10	145	c23e0467-d9f4-54dd-bdd8-875469cb75ab	5f2bd497-da00-5ac4-b4a7-3f00c6551edc	Супы	9

`cdm.user_product_counters`

user_product_counters

	id	user_id	product_id	product_name	order_cnt
1	293	f58d869c-6e95-59bd-ae43-fc5e42b6f9cd	c9112793-6e8e-5bfa-afd4-7698bdeec92b	РОЛЛ С ВЕТЧИНОЙ И ОМЛЕТОМ	3
2	294	f58d869c-6e95-59bd-ae43-fc5e42b6f9cd	0aea9f2d-e94f-57c1-8582-0445e54d9d58	РОЛЛ С РЫБОЙ И СОУСОМ ТАРТАР	5
3	296	f58d869c-6e95-59bd-ae43-fc5e42b6f9cd	1e7fd6d0-fb4c-584a-a8da-988522324494	НОРВЕЖСКИЙ СЭНДВИЧ С РЫБОЙ И ЯЙЦОМ	4
4	327	c23e0467-d9f4-54dd-bdd8-875469cb75ab	c3174ea2-9d69-55a5-a3f8-dbf028f5ed86	Рис с лимоном	4
5	312	ed6fc419-08de-5b95-adba-fd958e72a87a	13189c5b-2228-56ee-9539-358f8d3919bf	Плов чайханский	10
6	297	1625d297-3161-5f5b-8f48-c834deca39e7	c610d740-268a-5466-bec4-5f48ad435476	Манты с бараниной 3шт	6
7	298	1625d297-3161-5f5b-8f48-c834deca39e7	4bb98044-b664-5d95-bfa1-f6cf448c37a3	Казан кебаб с говядиной	6
8	299	1625d297-3161-5f5b-8f48-c834deca39e7	8f3ffedf-a380-52e7-83be-11bdddc9f506	Манты с говядиной 3шт	8
9	300	1625d297-3161-5f5b-8f48-c834deca39e7	922d0c3b-4440-5bb5-b884-888973240271	Лагман	6
10	301	1625d297-3161-5f5b-8f48-c834deca39e7	2a00b359-b061-5824-a07b-03a4473e36b0	Шурпа из говядины	10

8. Деплоим сервисы в `Kubernetes`

8.1 Собираем контейнеры

`STG`

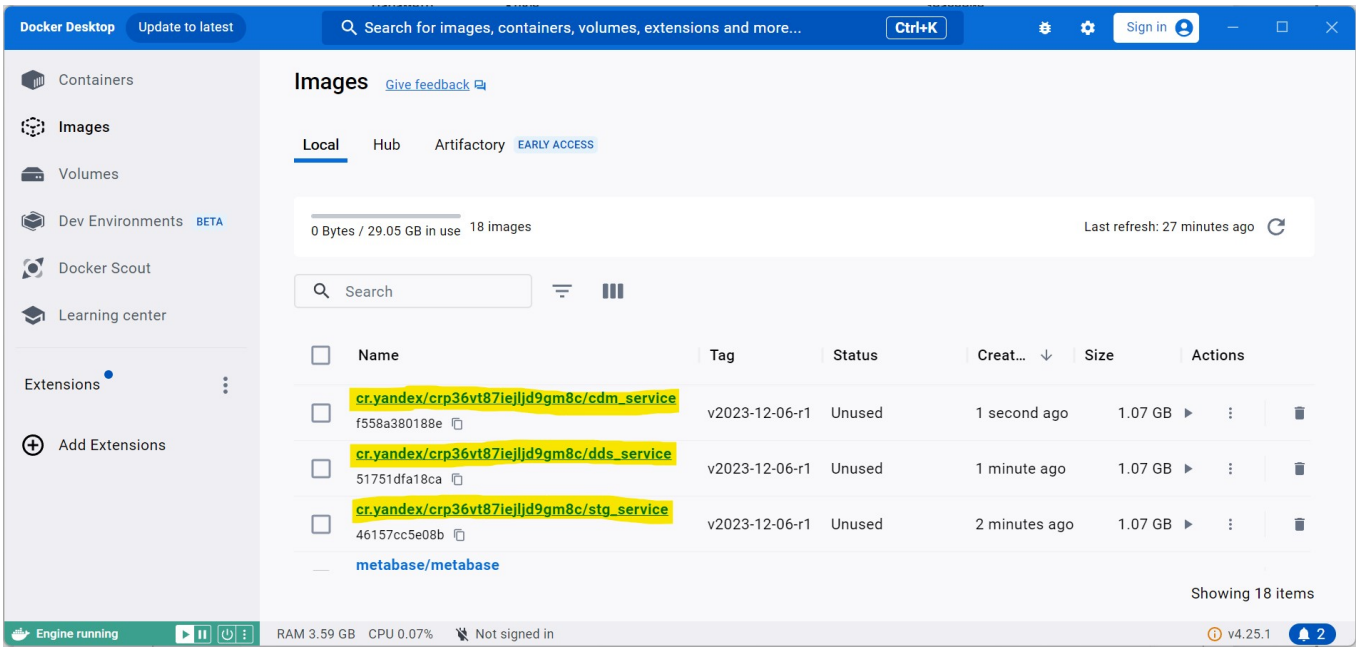
```
cd c:\GitHub\sprint-9-sample-service\service_stg
docker build . -t cr.yandex/crp36vt87iejljd9gm8c/stg_service:v2023-12-06-r1
```

`DDS`

```
cd c:\GitHub\sprint-9-sample-service\service_dds
docker build . -t cr.yandex/crp36vt87iejljd9gm8c/dds_service:v2023-12-06-r1
```

`CDM`

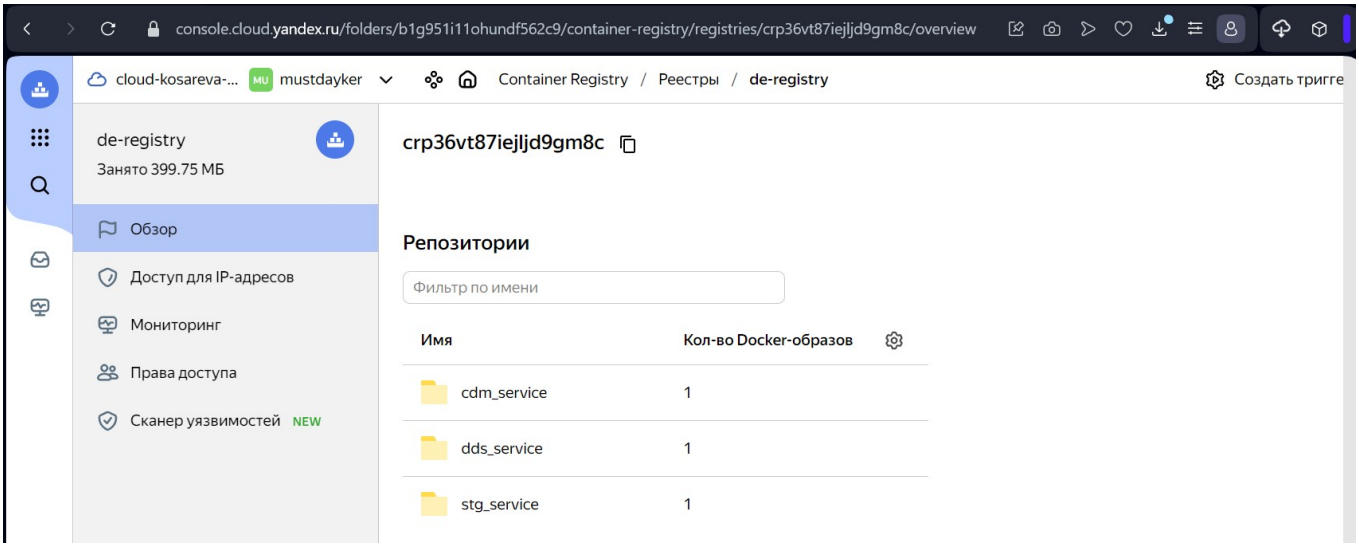
```
cd c:\GitHub\sprint-9-sample-service\service_cdm
docker build . -t cr.yandex/crp36vt87iejljd9gm8c/cdm_service:v2023-12-06-r1
```



8.2 Пушим образы в реджистри

```
docker push cr.yandex/crp36vt87iejljd9gm8c/stg_service:v2023-12-06-r1
docker push cr.yandex/crp36vt87iejljd9gm8c/dds_service:v2023-12-06-r1
docker push cr.yandex/crp36vt87iejljd9gm8c/cdm_service:v2023-12-06-r1
```

```
Командная строка | Windows PowerShell

Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS C:\Users\mustd> docker push cr.yandex/crp36vt87iejljd9gm8c/stg_service:v2023-12-06-r1
The push refers to repository [cr.yandex/crp36vt87iejljd9gm8c/stg_service]
5f70bf18a086: Layer already exists
034955842e23: Pushed
14921753d866: Pushed
ef44fa26cd33: Pushed
fa95e0d0b7bb: Pushed
52a0ca3c63f0: Pushed
dc10b1b84020: Pushed
8cc1d5d30a90: Pushed
9e2bd5ccd050: Pushed
70de58e39b72: Pushed
a04a14a911a5: Pushed
80bd043d4663: Layer already exists
30f5cd833236: Layer already exists
7c32e0608151: Layer already exists
7cea17427f83: Layer already exists
v2023-12-06-r1: digest: sha256:93e6b769d76add2a6e4e442f33faba733aedb60c49e9418a42c31a6b63fcf559 size: 3469
PS C:\Users\mustd>
```



8.3 Релизим сервис

STG

```
cd c:\GitHub\sprint-9-sample-service\service_stg
helm upgrade --install --atomic stg-service app -n c14-dmitrij-kosarev
```

DDS

```
cd c:\GitHub\sprint-9-sample-service\service_dds
helm upgrade --install --atomic dds-service app -n c14-dmitrij-kosarev
```

CDM

```
cd c:\GitHub\sprint-9-sample-service\service_cdm
helm upgrade --install --atomic cdm-service app -n c14-dmitrij-kosarev
```

```
Командная строка | Windows PowerShell | Command Prompt
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\mustd>cd c:\GitHub\sprint-9-sample-service\service_stg

c:\GitHub\sprint-9-sample-service\service_stg>helm upgrade --install --atomic stg-service app -n c14-dmitrij-kosarev
Release "stg-service" does not exist. Installing it now.
NAME: stg-service
LAST DEPLOYED: Wed Dec  6 18:48:07 2023
NAMESPACE: c14-dmitrij-kosarev
STATUS: deployed
REVISION: 1
TEST SUITE: None

c:\GitHub\sprint-9-sample-service\service_stg>cd c:\GitHub\sprint-9-sample-service\service_dds

c:\GitHub\sprint-9-sample-service\service_dds>helm upgrade --install --atomic dds-service app -n c14-dmitrij-kosarev
Release "dds-service" does not exist. Installing it now.
NAME: dds-service
LAST DEPLOYED: Wed Dec  6 18:49:22 2023
NAMESPACE: c14-dmitrij-kosarev
STATUS: deployed
REVISION: 1
TEST SUITE: None

c:\GitHub\sprint-9-sample-service\service_dds>cd c:\GitHub\sprint-9-sample-service\service_cdm

c:\GitHub\sprint-9-sample-service\service_cdm>helm upgrade --install --atomic cdm-service app -n c14-dmitrij-kosarev
Release "cdm-service" does not exist. Installing it now.
NAME: cdm-service
LAST DEPLOYED: Wed Dec  6 18:49:46 2023
NAMESPACE: c14-dmitrij-kosarev
STATUS: deployed
REVISION: 1
TEST SUITE: None

c:\GitHub\sprint-9-sample-service\service_cdm>
```

Проверяем

```
kubectl config get-contexts
kubectl get deployment
kubectl get pods
helm list
helm delete stg-service
helm delete dds-service
helm delete cdm-service
```

```
Командная строка | Windows PowerShell | Command Prompt

c:\GitHub\sprint-9-sample-service\service_cdm>kubectl config get-contexts
CURRENT   NAME                          CLUSTER                       AUTHINFO             NAMESPACE
*         practicum-data-engineer   practicum-data-engineer   c14-dmitrij-kosarev   c14-dmitrij-kosarev

c:\GitHub\sprint-9-sample-service\service_cdm>kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
cdm-service   1/1     1            1           114s
dds-service   1/1     1            1           2m19s
stg-service   1/1     1            1           3m34s

c:\GitHub\sprint-9-sample-service\service_cdm>kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
cdm-service-bd9d7bb94-bwmzv    1/1     Running   0          2m
dds-service-6d6c88f95-cf57d    1/1     Running   0          2m25s
stg-service-ff4d88fc6-zr9fx    1/1     Running   0          3m40s

c:\GitHub\sprint-9-sample-service\service_cdm>helm list
NAME            NAMESPACE               REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
cdm-service     c14-dmitrij-kosarev     1               2023-12-06 18:49:46.7452949 +0300 MSK   deployed        cdm-service-0.1.0       1.16.0
dds-service     c14-dmitrij-kosarev     1               2023-12-06 18:49:22.2849303 +0300 MSK   deployed        dds-service-0.1.0       1.16.0
stg-service     c14-dmitrij-kosarev     1               2023-12-06 18:48:07.4589253 +0300 MSK   deployed        stg-service-0.1.0       1.16.0

c:\GitHub\sprint-9-sample-service\service_cdm>
```

9. Визуализируем данные из `DWH` в `Datalens` ▲

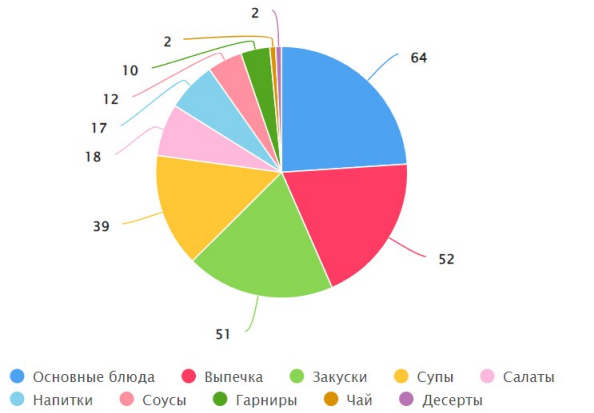
Требуются диаграммы:

- **круговая диаграмма**, показывающая популярность категорий блюд на основе доли пользователей, которые заказывали блюда этой категории;
- **круговая диаграмма**, показывающая популярность категорий блюд на основе доли заказов, в которых были блюда этой категории;
- **столбчатая диаграмма**, показывающая популярность блюд по количеству уникальных пользователей, заказавших блюдо;
- **столбчатая диаграмма**, показывающая популярность блюд по количеству заказов, в которых было блюдо.

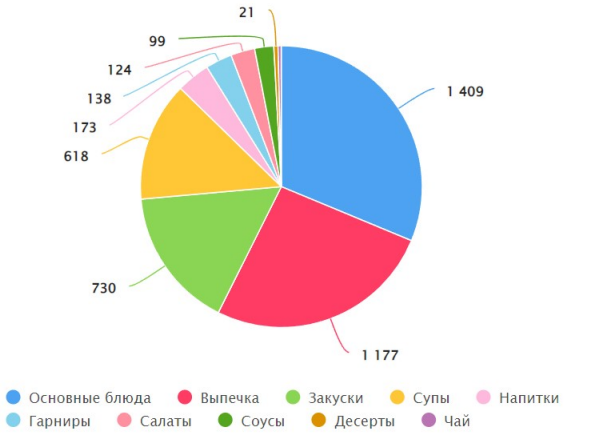
Sprint - 9 - Yandex Cloud

Пропорции

Популярность категорий блюд на основе доли уникальных пользователей

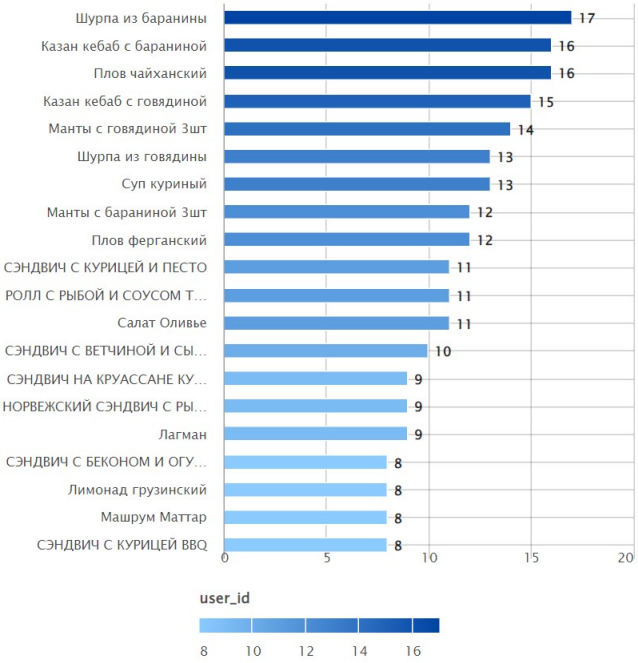


Популярность категорий блюд на основе доли заказов



Количество

Популярность блюд по количеству уникальных пользователей



Популярность блюд по количеству заказов