МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Отчет по лабораторной работе №1

Дисциплина: «Инженерия данных»

Выполнил: Морозов Д. А

Группа: 6233-010402D

Самара 2025

1. Кратко об архитектуре: схема пайплайна, какие инструменты и почему.

Пайплайн реализован в виде ETL-процесса с использованием оркестратора Prefect, который управляет выполнением задач, логированием и обработкой ошибок. Архитектура построена по модульному принципу и включает этапы извлечения данных, трансформации и загрузки результатов в аналитическое хранилище.

В качестве источника данных используется публичный API Open-Meteo, предоставляющий погодные прогнозы. Сырые ответы API сохраняются в MinIO, который используется как объектное хранилище для хранения неизменённых данных и обеспечения воспроизводимости обработки.

Когда я создавал свои тг. Решения, то не использовал подробный инструментарий. Поработал и понял:

1. Очень удобно хранить исходники в Метео (Я возможно начну с ним работать)
2. Perfect по сути работу урощает, сокращает код, которые нужно написать + организовывает бизнес-логику (точнее следует ей)
3. Кликхаус единственной проблемой стал – я потратил 2 дня на то, чтобы отчет сформировать из-за постоянных проблем с доступом/несоответствием полей (такое вообще впервые вижу()

2. Источник данных: эндпоинт Open-Meteo, параметры запроса.

В качестве источника данных используется публичный погодный API Open-Meteo, удобно можно забирать данные необходимые (С парсингом новостей +- так же, взял по вашему совету)

Эндпоинт - https://api.open-meteo.com/v1/forecast

Параметры запроса:

1. latitude, longitude — координаты города
2. timezone — временная зона города
3. start_date, end_date — дата прогноза (завтра)
4. hourly — список почасовых показателей

3. Extract → Transform → Load: по 2–3 предложения на этап.

1) Extract

На этапе Extract = запрос к API Open-Meteo для получения прогноза погоды на следующий день по Самаре+Москва. Запрос формируется с указанием координат города, временной зоны и списка почасовых метеопараметров (вот это подсмотрел). Полученный JSON-ответ сохраняется в объектное хранилище MinIO, сырой файл.

2) Transform

На этапе Transform почасовые данные нормализуются в табличный формат: извлекаются временные метки, температура, осадки, скорость и направление ветра. Дополнительно рассчитываются агрегированные дневные показатели: минимальная, максимальная и средняя температура, сумма осадков и максимальная скорость ветра. В

процессе трансформации выполняется базовая фильтрация некорректных данных (удаление записей без временной метки).

3) Load

На этапе Load преобразованные данные загружаются в аналитическое хранилище ClickHouse. И оттуда с ними уже можно работать (Самый проблемный момент, опять же. Я, используя нейронку, три раза за 3 разных дня ПЕРЕПИСЫВАЛ все параметры хранилища)

4. Extract → Transform → Load: по 2–3 предложения на этап.

Контроль успешности HTTP-ответа источника данных (raise_for_status), фильтрация записей без временной метки при трансформации и сохранение сырых JSON-ответов в MinIO для трассировки и повторной обработки. Устойчивость пайплайна повышена за счёт повторных попыток выполнения задач Extract и Load в Prefect.

Основные точки сбоя: недоступность API Open-Meteo(Ни разу не было), ошибки подключения и прав доступа в ClickHouse (Главная проблема, ее не решил, ее по сути и не решить. Только избавиться от ошибок), а также несоответствие схемы данных таблицам хранилища. Наиболее проблемным этапом стала настройка хранилища и прав доступа, потребовавшая многократной переработки конфигурации.

5. Результаты: скриншоты/логи Prefect, пример содержимого бакета в MinIO, фрагменты из weather_hourly/weather_daily (1–2 запроса ClickHouse), пример уведомления с прогнозом в Telegram.

Скриншот из перфекта

Рисунок 1 - запуск + все работает

Логи перфекта, после успешного выполнения полного цикла: от получения сырых данных до подведения статистики в TG

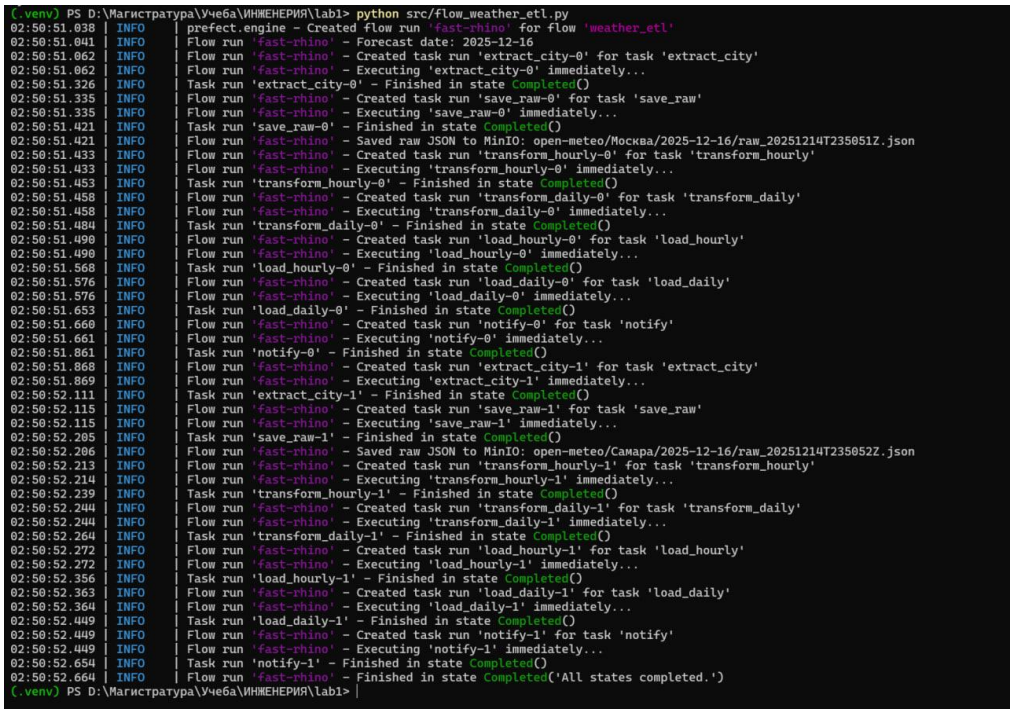
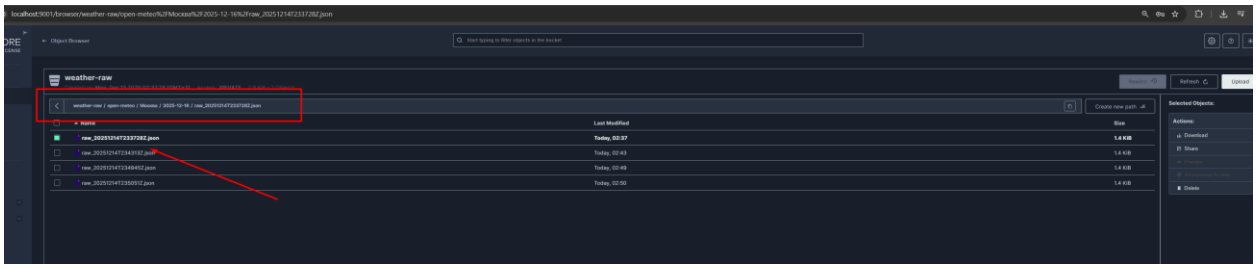
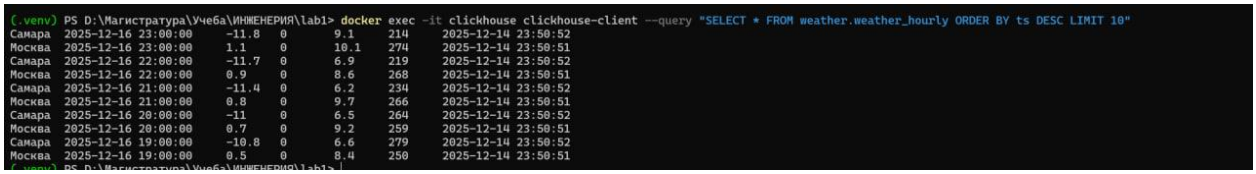


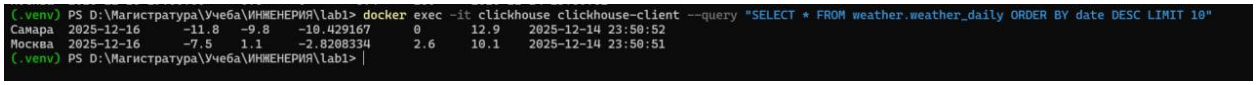
Рисунок 2 - загружено -> отправлено

Рисунок с хранилищем мино

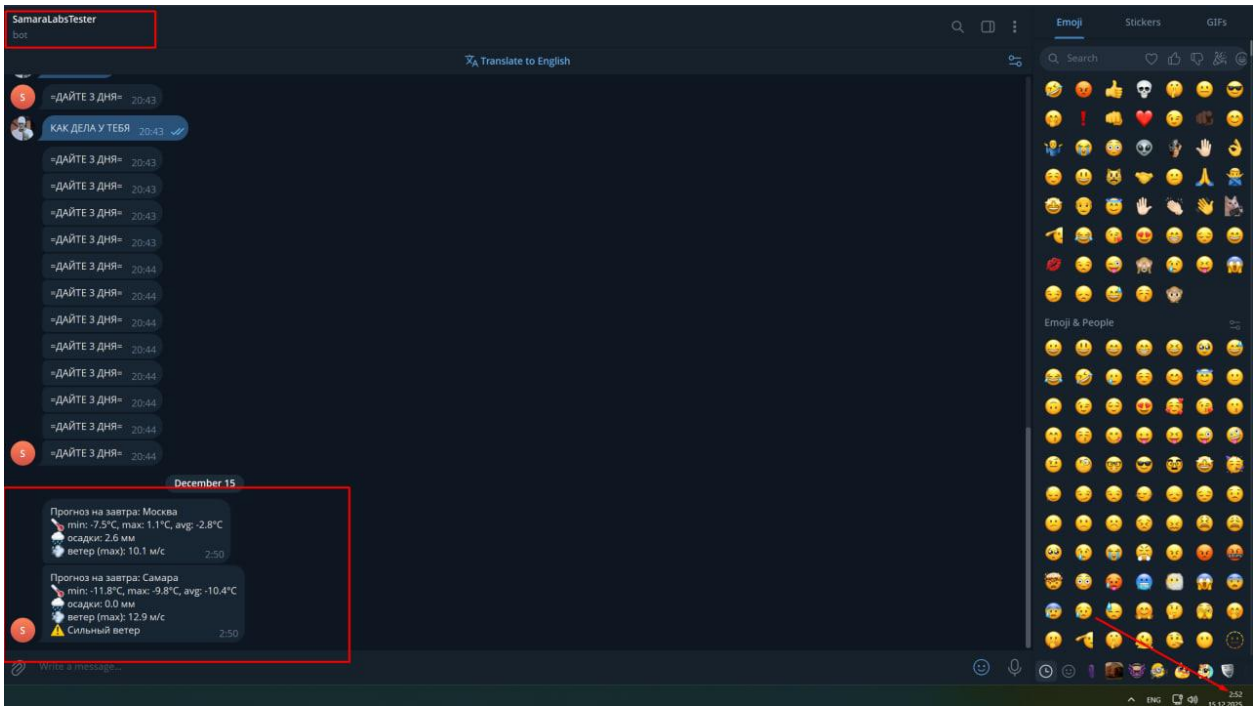


А так же уже сохраненные после в кликхаус данные в двух таблицах





Отправленная статистика в телеграмм (с датами)



6. Выводы: что было сложным, что бы улучшили.

По сути лабораторная очень прикладная и полезная, ничего сложного не было, единственное – кликхаус стал очень проблемным.

Если бы было время, я бы подумал над тем, как его ошибки обработать, т.к я, даже с ЧатГПТ и интернетом потратил часы на исправление банальных ошибок

7. Дополнительно:

Пришлось заимствовать код из интернета/нейронок, по причинам, которые я указал в начале отчета. Например:

1. Функция трансформирования была взята из интернета (изменена слегка, но сам факт), потому что мне пришлось несколько раз полностью код пересобирать для совпадения полей и прочих структур

```python
def aggregate_daily(hourly_df: pd.DataFrame) -> pd.DataFrame:
    row = {
        "city": hourly_df["city"].iloc[0],
        "date": hourly_df["ts"].dt.date.iloc[0],
        "temp_min_c": float(hourly_df["temperature"].astype(float).min()),
        "temp_max_c": float(hourly_df["temperature"].astype(float).max()),
        "temp_avg_c": float(hourly_df["temperature"].astype(float).mean()),
        "precipitation_sum_mm": float(hourly_df["precipitation"].astype(float).sum()),
        "wind_speed_max_ms": float(hourly_df["wind_speed"].astype(float).max()),
        "ingested_at": datetime.now(timezone.utc),
    }
    return pd.DataFrame([row])
```

2. Оформление уведомления в тг я забрал из своих прошлых решений, которые оформлял с помощью ЧАТ-гпт, пример

```python
warn = []
if w >= 12:
    warn.append("⚠ Сильный ветер")
if p >= 10:
    warn.append("⚠ Сильные осадки")
```

3. Код связанный с опенметио тоже подглядывал, но мне кажется, что в этом случае это уж точно обоснованно
4. Так же были пункты заимствования в файлах кликхауса и докера, но я их не вспомню (там по минимуму, просто подглядывал при написании)

5. Файл версий библиотек загрузил, но сюда продублирую:

```
prefect==2.20.0
requests==2.32.3
pandas==2.2.2
python-dotenv==1.0.1
minio==7.2.7
clickhouse-connect==0.7.19

# =П.С= Мб докинуть что-то еще, но вроде все ок пока
```

У меня возникли проблемы с версиями перфекта и кликхауса, насколько я помню, но в текстовом файле библиотек я их не заменил, забыл