

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 уращает, 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 Самара+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 многоократной 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

```
PS D:\Магистратура\Учеба\ИНЖЕНЕРИЯ\lab1> .\venv\Scripts\Activate.ps1
(.venv) PS D:\Магистратура\Учеба\ИНЖЕНЕРИЯ\lab1> .\venv\Scripts\Activate.ps1
(.venv) PS D:\Магистратура\Учеба\ИНЖЕНЕРИЯ\lab1> prefect version
Version:          2.20.0
API version:      0.8.4
Python version:   3.11.9
Git commit:       15274df8
Built:            Thu, Aug 1, 2024 3:14 PM
OS/Arch:          win32/AMD64
Profile:          default
Server type:     ephemeral
Server:
  Database:      sqlite
  SQLite version: 3.45.1
(.venv) PS D:\Магистратура\Учеба\ИНЖЕНЕРИЯ\lab1> prefect server start

[----] [----] [----] [----] [----]
[----] [----] [----] [----] [----]
[----] [----] [----] [----] [----]

Configure Prefect to communicate with the server with:

prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api

View the API reference documentation at http://127.0.0.1:4200/docs

Check out the dashboard at http://127.0.0.1:4200
```

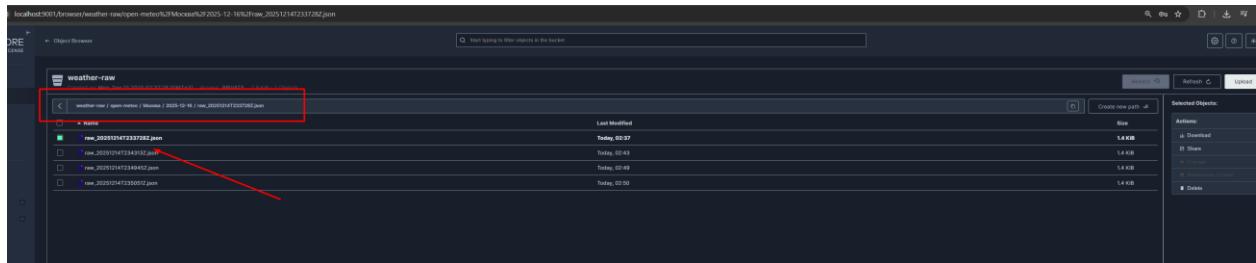
*Рисунок 1 - запуск + все работает*

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

```
(.venv) PS D:\Магистратура\Учеба\ИНЖЕНЕРИЯ\lab1> python src/flow_weather_etl.py
02:50:51.038 | INFO | prefect.engine - Created flow run 'fast-rhino' for flow 'weather_etl'
02:50:51.041 | INFO | Flow run 'fast-rhino' - Forecast date: 2025-12-16
02:50:51.062 | INFO | Flow run 'fast-rhino' - Created task run 'extract_city-0' for task 'extract_city'
02:50:51.062 | INFO | Flow run 'fast-rhino' - Executing 'extract_city-0' immediately...
02:50:51.326 | INFO | Task run 'extract_city-0' - Finished in state Completed()
02:50:51.335 | INFO | Flow run 'fast-rhino' - Created task run 'save_raw-0' for task 'save_raw'
02:50:51.335 | INFO | Flow run 'fast-rhino' - Executing 'save_raw-0' immediately...
02:50:51.421 | INFO | Task run 'save_raw-0' - Finished in state Completed()
02:50:51.421 | INFO | Flow run 'fast-rhino' - Saved raw JSON to MinIO: open-meteo/Moskva/2025-12-16/raw_20251214T235051Z.json
02:50:51.433 | INFO | Flow run 'fast-rhino' - Created task run 'transform_hourly-0' for task 'transform_hourly'
02:50:51.433 | INFO | Flow run 'fast-rhino' - Executing 'transform_hourly-0' immediately...
02:50:51.453 | INFO | Task run 'transform_hourly-0' - Finished in state Completed()
02:50:51.458 | INFO | Flow run 'fast-rhino' - Created task run 'transform_daily-0' for task 'transform_daily'
02:50:51.458 | INFO | Flow run 'fast-rhino' - Executing 'transform_daily-0' immediately...
02:50:51.484 | INFO | Task run 'transform_daily-0' - Finished in state Completed()
02:50:51.490 | INFO | Flow run 'fast-rhino' - Created task run 'load_hourly-0' for task 'load_hourly'
02:50:51.490 | INFO | Flow run 'fast-rhino' - Executing 'load_hourly-0' immediately...
02:50:51.568 | INFO | Task run 'load_hourly-0' - Finished in state Completed()
02:50:51.578 | INFO | Flow run 'fast-rhino' - Created task run 'load_daily-0' for task 'load_daily'
02:50:51.578 | INFO | Flow run 'fast-rhino' - Executing 'load_daily-0' immediately...
02:50:51.653 | INFO | Task run 'load_daily-0' - Finished in state Completed()
02:50:51.660 | INFO | Flow run 'fast-rhino' - Created task run 'notify-0' for task 'notify'
02:50:51.660 | INFO | Flow run 'fast-rhino' - Executing 'notify-0' immediately...
02:50:51.861 | INFO | Task run 'notify-0' - Finished in state Completed()
02:50:51.868 | INFO | Flow run 'fast-rhino' - Created task run 'extract_city-1' for task 'extract_city'
02:50:51.869 | INFO | Flow run 'fast-rhino' - Executing 'extract_city-1' immediately...
02:50:52.111 | INFO | Task run 'extract_city-1' - Finished in state Completed()
02:50:52.115 | INFO | Flow run 'fast-rhino' - Created task run 'save_raw-1' for task 'save_raw'
02:50:52.115 | INFO | Flow run 'fast-rhino' - Executing 'save_raw-1' immediately...
02:50:52.205 | INFO | Task run 'save_raw-1' - Finished in state Completed()
02:50:52.208 | INFO | Flow run 'fast-rhino' - Saved raw JSON to MinIO: open-meteo/Camapa/2025-12-16/raw_20251214T235052Z.json
02:50:52.213 | INFO | Flow run 'fast-rhino' - Created task run 'transform_hourly-1' for task 'transform_hourly'
02:50:52.214 | INFO | Flow run 'fast-rhino' - Executing 'transform_hourly-1' immediately...
02:50:52.239 | INFO | Task run 'transform_hourly-1' - Finished in state Completed()
02:50:52.244 | INFO | Flow run 'fast-rhino' - Created task run 'transform_daily-1' for task 'transform_daily'
02:50:52.244 | INFO | Flow run 'fast-rhino' - Executing 'transform_daily-1' immediately...
02:50:52.264 | INFO | Task run 'transform_daily-1' - Finished in state Completed()
02:50:52.272 | INFO | Flow run 'fast-rhino' - Created task run 'load_hourly-1' for task 'load_hourly'
02:50:52.272 | INFO | Flow run 'fast-rhino' - Executing 'load_hourly-1' immediately...
02:50:52.355 | INFO | Task run 'load_hourly-1' - Finished in state Completed()
02:50:52.363 | INFO | Flow run 'fast-rhino' - Created task run 'load_daily-1' for task 'load_daily'
02:50:52.364 | INFO | Flow run 'fast-rhino' - Executing 'load_daily-1' immediately...
02:50:52.449 | INFO | Task run 'load_daily-1' - Finished in state Completed()
02:50:52.449 | INFO | Flow run 'fast-rhino' - Created task run 'notify-1' for task 'notify'
02:50:52.449 | INFO | Flow run 'fast-rhino' - Executing 'notify-1' immediately...
02:50:52.654 | INFO | Task run 'notify-1' - Finished in state Completed()
02:50:52.664 | INFO | Flow run 'fast-rhino' - Finished in state Completed('All states completed.')
(.venv) PS D:\Магистратура\Учеба\ИНЖЕНЕРИЯ\lab1>
```

*Рисунок 2 - загружено -> отправлено*

## Рисунок с хранилищем мино

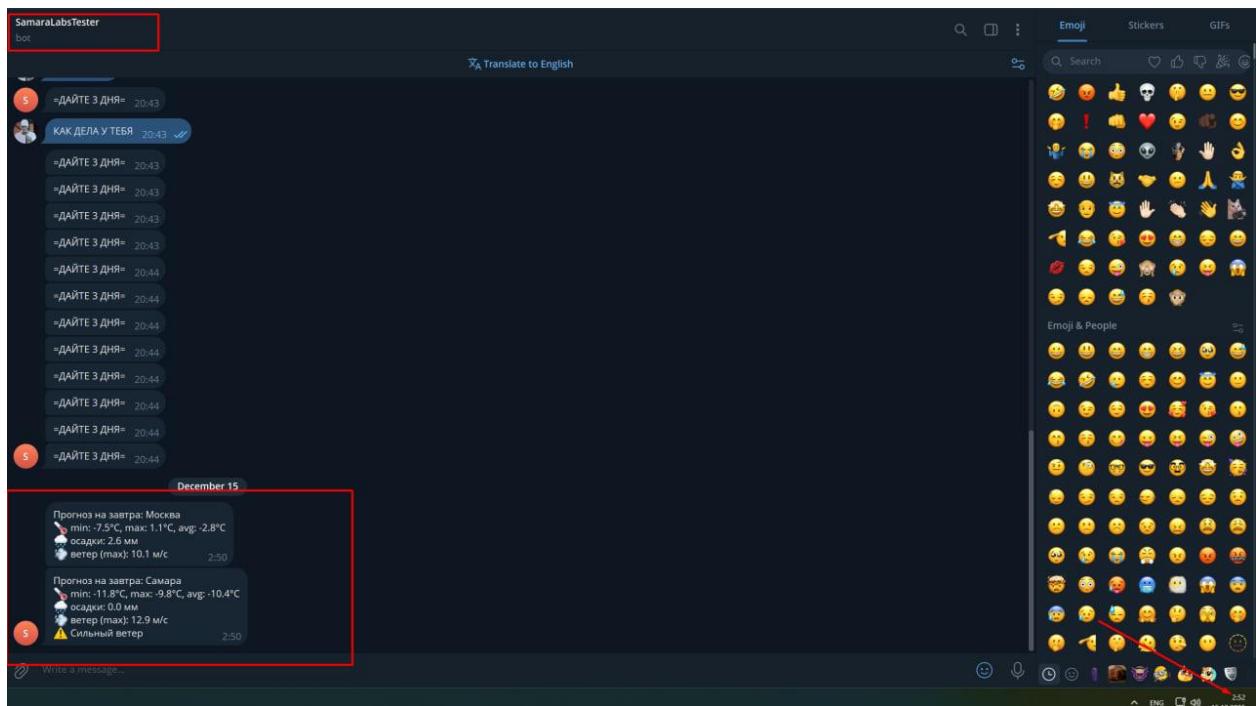


А так же уже сохраненные после в кликхаус данные в двух таблицах

```
[.venv] PS D:\Магистратура\Учеба\ИИМЕНЕРИ\lab1> docker exec -it clickhouse clickhouse-client --query "SELECT * FROM weather.weather_hourly ORDER BY ts DESC LIMIT 10"
Самара 2025-12-16 23:00:00 -11.8 0 9.1 214 2025-12-14 23:50:52
Москва 2025-12-16 23:00:00 1.1 0 18.1 274 2025-12-14 23:50:51
Самара 2025-12-16 22:00:00 -11.7 0 6.9 219 2025-12-14 23:50:52
Москва 2025-12-16 22:00:00 0.9 0 8.6 268 2025-12-14 23:50:51
Самара 2025-12-16 21:00:00 -11.4 0 6.2 234 2025-12-14 23:50:52
Москва 2025-12-16 21:00:00 0.8 0 9.7 266 2025-12-14 23:50:51
Самара 2025-12-16 20:00:00 -11 0 6.5 264 2025-12-14 23:50:52
Москва 2025-12-16 20:00:00 0.7 0 9.2 259 2025-12-14 23:50:51
Самара 2025-12-16 19:00:00 -10.8 0 6.6 279 2025-12-14 23:50:52
Москва 2025-12-16 19:00:00 0.5 0 8.4 250 2025-12-14 23:50:51

[.venv] PS D:\Магистратура\Учеба\ИИМЕНЕРИ\lab1> docker exec -it clickhouse clickhouse-client --query "SELECT * FROM weather.weather_daily ORDER BY date DESC LIMIT 10"
Самара 2025-12-16 -11.8 -9.8 -10.429167 0 12.9 2025-12-14 23:56:52
Москва 2025-12-16 -7.5 1.1 -2.8208334 2.6 10.1 2025-12-14 23:56:51
(.venv) PS D:\Магистратура\Учеба\ИИМЕНЕРИ\lab1>
```

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

```
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

```
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