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
Кафедра суперкомпьютеров и общей информатики

Отчет по лабораторной работе №1

Дисциплина: «Инженерия данных»

Выполнил: Вечканова П.А.

Группа: 6233-010402D

Самара 2025

ЗАДАНИЕ

Цели работы

- На практике освоить полный цикл ETL: извлечение из публичного API, трансформация и загрузка в ClickHouse.
- Научиться собирать и отлаживать пайплайны в **Prefect**.
- Научиться настраивать окружение для работы с данными.

Описание пайплайна

Заданием на лабораторную работу является реализация следующего пайплайна:

1. В качестве источника данных предлагается использовать Free Weather API
2. (Extract) Получить прогноз **на завтра** по переменным: *температура, осадки, скорость и направление ветра* для городов **Самара и Москва**. Сырые ответы API сохранить в объектном хранилище
3. (Transform)
    - Извлечь почасовые значения и нормализовать для таблицы weather_hourly
    - Посчитать дневную статистику (min, max, avg температуру и количество осадков) и подготовить для сохранения в таблице weather_daily
4. (Load) Загрузить преобразованные данные в соответствующие таблицы ClickHouse
5. Автоматически отправить уведомления в Telegram с кратким прогнозом на завтра и предупреждать о сильном ветре/осадках
6. **(Опционально)** Реализовать обработку различных ошибок.

ВЫПОЛНЕНИЕ РАБОТЫ

1. АРХИТЕКТУРА

Пайплайн реализован по схеме ETL с использованием**:**

- Prefect (версия Prefect 2.10): для оркестрации задач, планирования (ежедневно в 08:00 по МСК), логирования и повторных запусков;
- MinIO (RELEASE.2024-12-05T23-40-55Z): для хранения сырых JSON-ответов API (обеспечивает воспроизводимость и аудит);
- ClickHouse (версия 24.8): для эффективного хранения и анализа структурированных метеоданных;
- Telegram Bot API: для отправки краткого прогноза и предупреждений.

Выбор обусловлен открытостью, лёгкостью развёртывания в Docker и соответствием задаче обработки временных рядов.

2. ИСТОЧНИКИ ДАННЫХ

Использован публичный Open-Meteo API (https://api.open-meteo.com/v1/forecast). Параметры запроса:

- latitude, longitude – координаты Москвы (55.7522, 37.6156) и Самары (53.1959, 50.1001);
- hourly=temperature_2m,precipitation,wind_speed_10m,wind_direction_10m;
- timezone=Europe/Moscow;
- forecast_days=2 (чтобы получить данные на завтра).

3. ОПИСАНИЕ РАБОТЫ ETL-ПАЙПЛАЙНА

Извленчение данных (Extract): Для каждого города выполняется HTTP-запрос к Open-Meteo. Полученный JSON сохраняется в MinIO в формате {город}/{дата}.json – это позволяет сохранить исходные данные для отладки.

Обработка данных (Transform): Из сырых данных извлекаются почасовые значения только на следующий день. На их основе формируются записи для таблиц weather_hourly и агрегированные дневные статистики (min, max, avg температура, сумма осадков) для weather_daily.

Загрузка данных (Load): Данные загружаются в ClickHouse через clickhouse-driver. В обе таблицы добавляется поле _loaded_at с меткой времени загрузки. Одновременно формируется и отправляется Telegram-уведомление.

4. КАЧЕСТВО ДАННЫХ И ОБРАБОТКА ОШИБОК

Реализованы базовые проверки: валидация HTTP-статуса, проверка существования бакета MinIO, корректный парсинг дат.

5. ВЫПОЛНЕНИЕ ПАЙПЛАЙНА И РЕЗУЛЬТАТЫ

Первым этапом работы была настройка телеграм бота с помощью @BotFather и получение ID чата для дальнейшей настройки работы.

Далее, написав код и настроив docker-compose, запускаем контейнер с помощью команды docker-compose up -d (пример запуска и вывод представлены на рисунке 1)

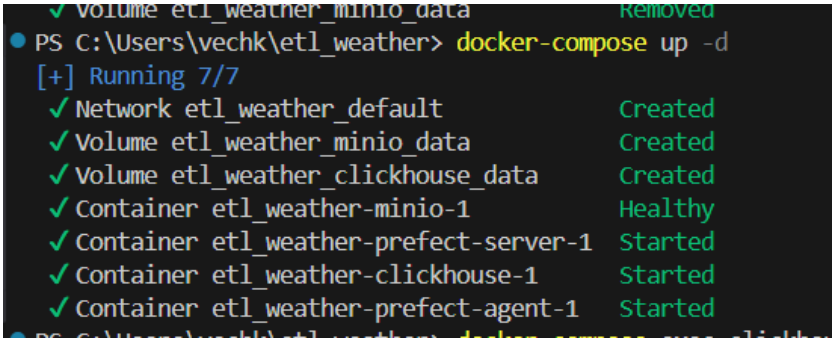


Рисунок 1 – Запуск контейнера

Делаем деплой расписания для того, чтобы отправка была автоматической. Пример команды представлен на рисунке 2.

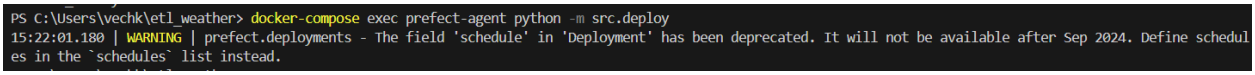


Рисунок 2 – Деплой расписания

Заходим в Prefect и проверяем работу пайплайна

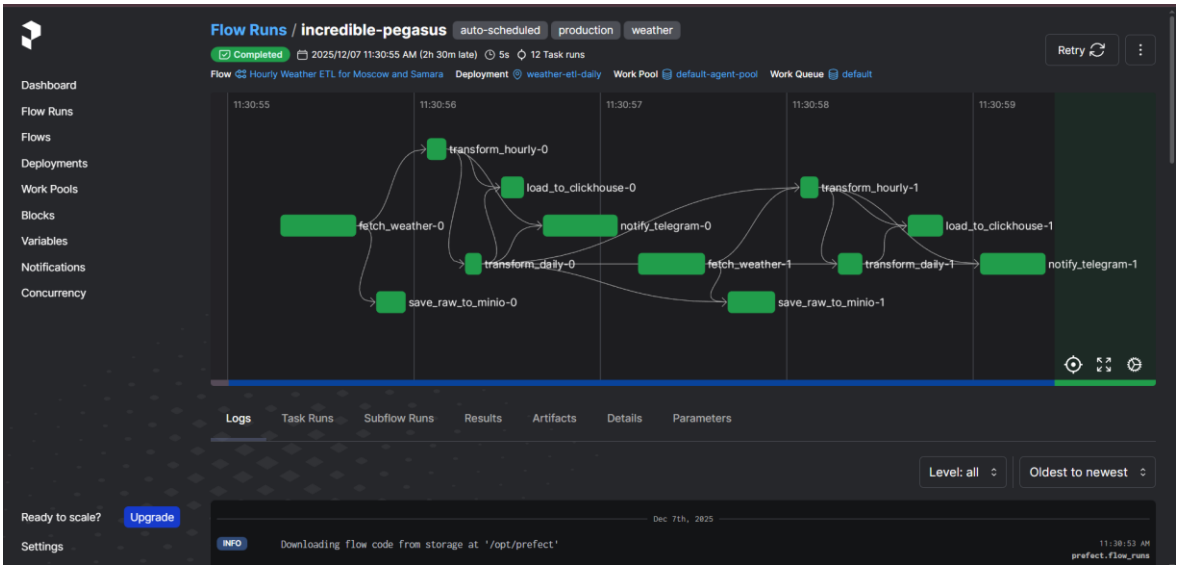


Рисунок 3 – Граф работы пайплайна

Зайдем в UI Clickhose (рисунок 4) и убедимся в том, что данные загружены с помощью команд

- SELECT * FROM weather.weather_daily (вывод представлен на рисунке 5)
- SELECT * FROM weather.weather_hourly (вывод представлен на рисунке 6)

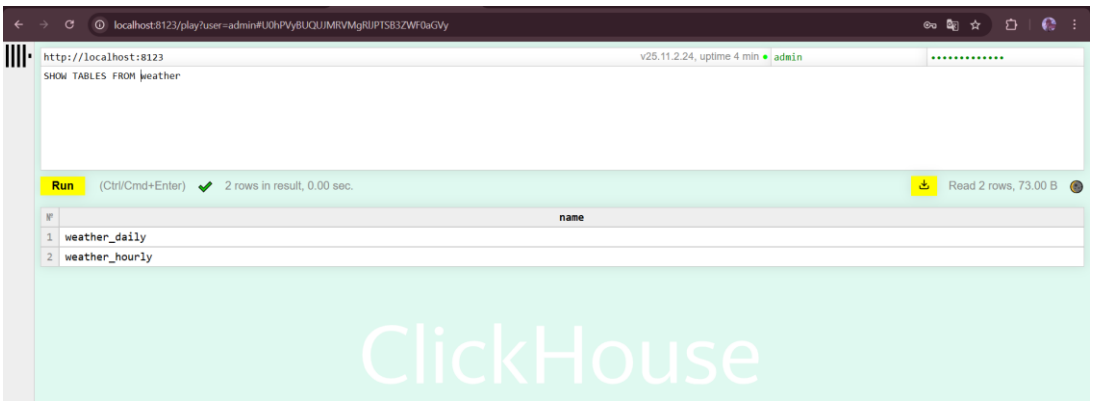
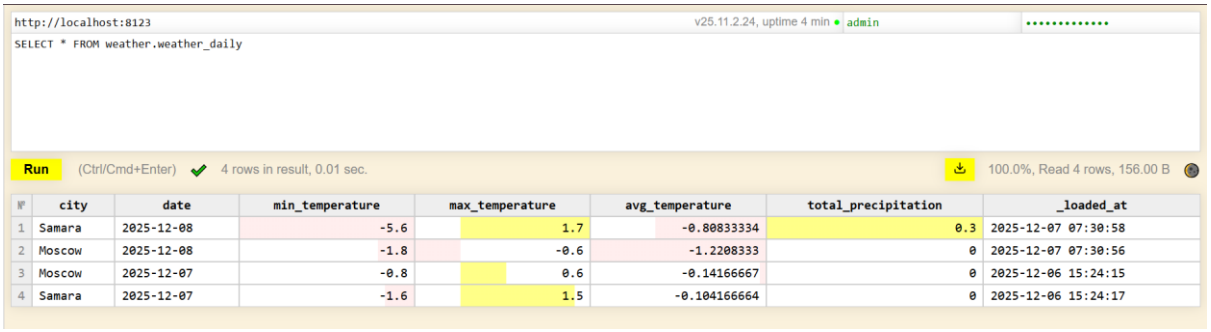
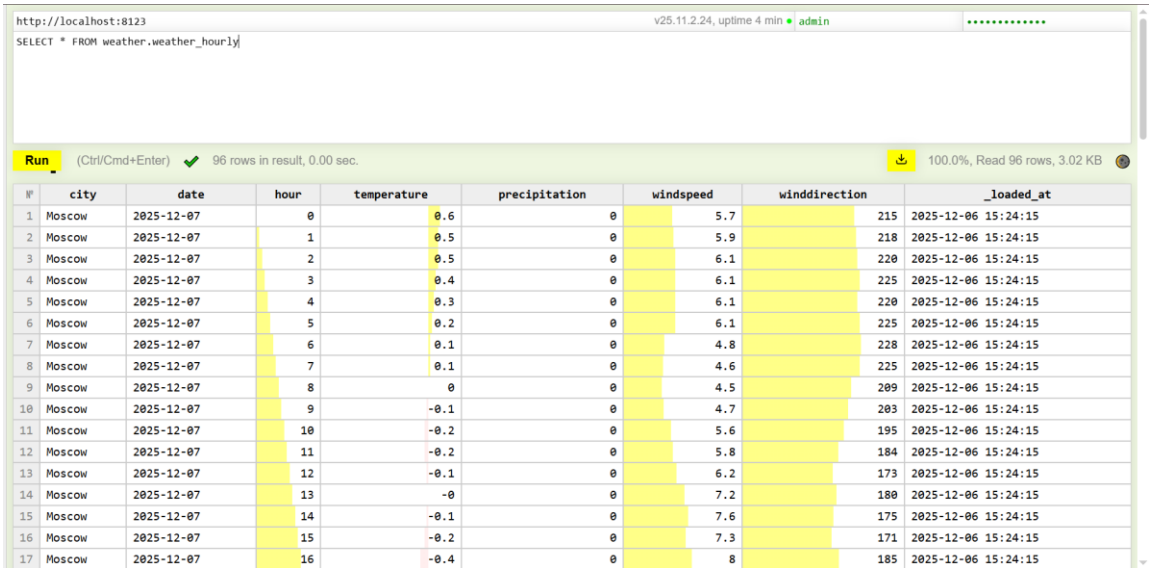


Рисунок 4 – UI Clickhose с созданными таблицами



№	city	date	min_temperature	max_temperature	avg_temperature	total_precipitation	_loaded_at
1	Samara	2025-12-08	-5.6	1.7	-0.80833334	0.3	2025-12-07 07:30:58
2	Moscow	2025-12-08	-1.8	-0.6	-1.2208333	0	2025-12-07 07:30:56
3	Moscow	2025-12-07	-0.8	0.6	-0.14166667	0	2025-12-06 15:24:15
4	Samara	2025-12-07	-1.6	1.5	-0.104166664	0	2025-12-06 15:24:17

Рисунок 5 – SELECT запрос к таблице weather.weather_daily



№	city	date	hour	temperature	precipitation	windspeed	winddirection	_loaded_at
1	Moscow	2025-12-07	0	0.6	0	5.7	215	2025-12-06 15:24:15
2	Moscow	2025-12-07	1	0.5	0	5.9	218	2025-12-06 15:24:15
3	Moscow	2025-12-07	2	0.5	0	6.1	220	2025-12-06 15:24:15
4	Moscow	2025-12-07	3	0.4	0	6.1	225	2025-12-06 15:24:15
5	Moscow	2025-12-07	4	0.3	0	6.1	220	2025-12-06 15:24:15
6	Moscow	2025-12-07	5	0.2	0	6.1	225	2025-12-06 15:24:15
7	Moscow	2025-12-07	6	0.1	0	4.8	228	2025-12-06 15:24:15
8	Moscow	2025-12-07	7	0.1	0	4.6	225	2025-12-06 15:24:15
9	Moscow	2025-12-07	8	0	0	4.5	209	2025-12-06 15:24:15
10	Moscow	2025-12-07	9	-0.1	0	4.7	203	2025-12-06 15:24:15
11	Moscow	2025-12-07	10	-0.2	0	5.6	195	2025-12-06 15:24:15
12	Moscow	2025-12-07	11	-0.2	0	5.8	184	2025-12-06 15:24:15
13	Moscow	2025-12-07	12	-0.1	0	6.2	173	2025-12-06 15:24:15
14	Moscow	2025-12-07	13	-0	0	7.2	180	2025-12-06 15:24:15
15	Moscow	2025-12-07	14	-0.1	0	7.6	175	2025-12-06 15:24:15
16	Moscow	2025-12-07	15	-0.2	0	7.3	171	2025-12-06 15:24:15
17	Moscow	2025-12-07	16	-0.4	0	8	185	2025-12-06 15:24:15

Рисунок 6 – SELECT запрос к таблице weather.weather_hourly

Проверим, что json-файлы сохранены в MinIO (рисунки 7 и 8)

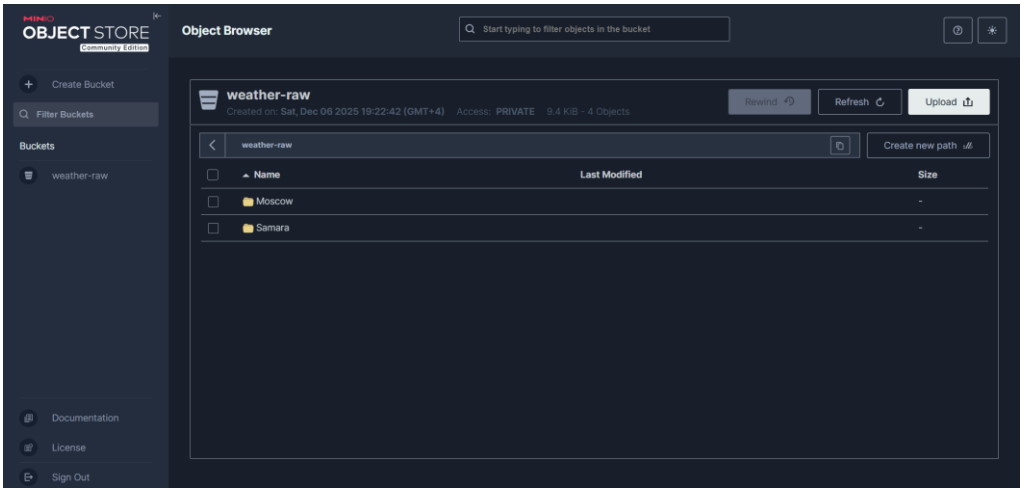


Рисунок 7 – Интерфейс MinIO с папками по городам

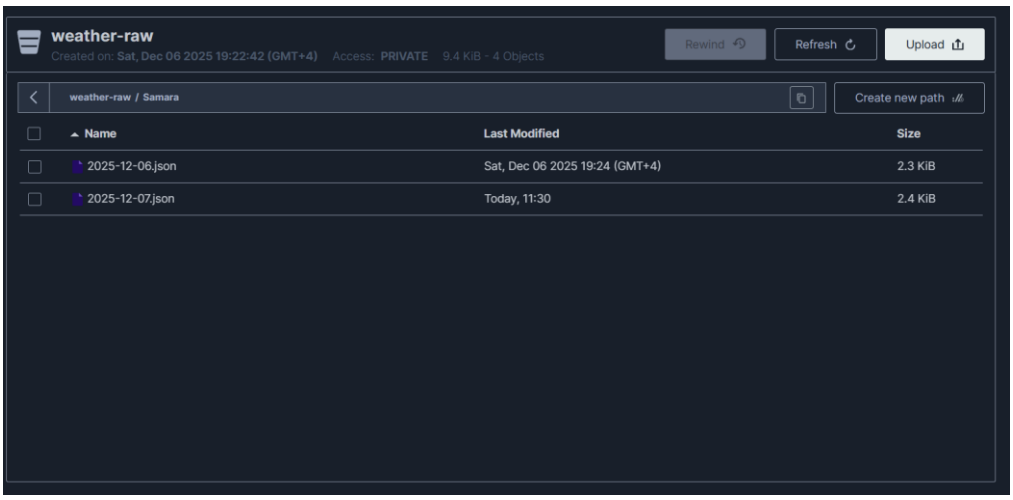


Рисунок 8 – Интерфейс MinIO с json-файлами для Самары

В телеграм также видим сообщения о прогнозе погоды, подтверждающие успех работы пайплайна (рисунок 8).

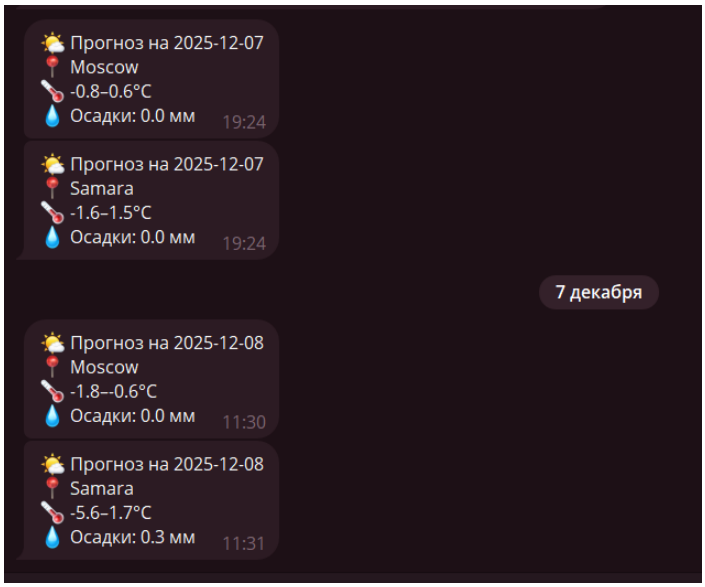


Рисунок 9 – Сообщения от бота в телеграм

ВЫВОДЫ

В ходе выполнения лабораторной работы выполнены все поставленные цели: настроен полный цикл ETL (извлечение из внешнего ресурса, обработка и заргузка в Кликхаус), собран и отлажен пайплайн в Prefect, настроено окружение для работы с данными. Также настроена отправка сообщений через телеграмм бота.

Возникали трудности с настройкой окружения, так как некоторые версии инструментов конфликтовали друг с другом, но данная проблема была решена.

Из улучшений можно добавить выбор городов и определенных параметров в телеграмм боте, а также динамичное изменение времени отправки сообщений (выбирается пользователем во сколько будет приходить сообщение).