

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 киберфотоники

Отчет по лабораторной работе №1
Дисциплина: «Инженерия данных»

Выполнила: Цибикина А.И.
Группа: 6233-010402D

Самара 2025

Архитектура:

Схема пайплайна:

Prefect (оркестратор) → Open-Meteo API (источник) → MinIO (сырые данные), → ClickHouse (агрегированные таблицы) → Telegram Bot (уведомления).

Какие инструменты и почему:

- Prefect 2.14.21 — для оркестрации, управления зависимостями и расписания. Выбор стабильной версии 2.x обусловлен совместимостью с зависимостями и простотой настройки.
 - MinIO — S3-совместимое объектное хранилище для сырых JSON-ответов от API.
 - ClickHouse — высокопроизводительная колоночная СУБД.
 - Telegram Bot API — канал для доставки push-уведомлений. Прост в интеграции

Источник данных:

Бесплатный API Open-Meteo.

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

Параметры запроса:

- latitude, longitude — точные координаты Москвы и Самары, жестко заданные в коде.
 - hourly=temperature_2m,precipitation,windspeed_10m,winddirection_10m — набор необходимых почасовых метрик.
 - daily=temperature_2m_max,temperature_2m_min,precipitation_sum,windspeed_10m_max — набор дневных агрегатов для расчета сводки.
 - timezone=Europe/Moscow — для получения времени в нужном часовом поясе.
 - start_date, end_date — устанавливаются на дату "завтра", чтобы запросить прогноз строго за один день.

Extract → Transform → Load:

Extract: Пайpline запускается по расписанию, после чего асинхронно отправляется HTTP-запросы к Open-Meteo API для каждого города. Сырые JSON-ответы сохраняются в MinIO в структурированные папки по городу и дате.

Transform: Почасовые данные из JSON разворачиваются в плоскую структуру для таблицы weather_hourly. Параллельно рассчитываются дневные агрегаты (min/max/avg температура, сумма осадков) для таблицы weather_daily.

Load: Преобразованные данные пакетом загружаются в таблицы ClickHouse. В параллельной задаче формируется и отправляется текстовое уведомление в Telegram с прогнозом и алертами на основе дневных данных.

Качество данных:

Проверки:

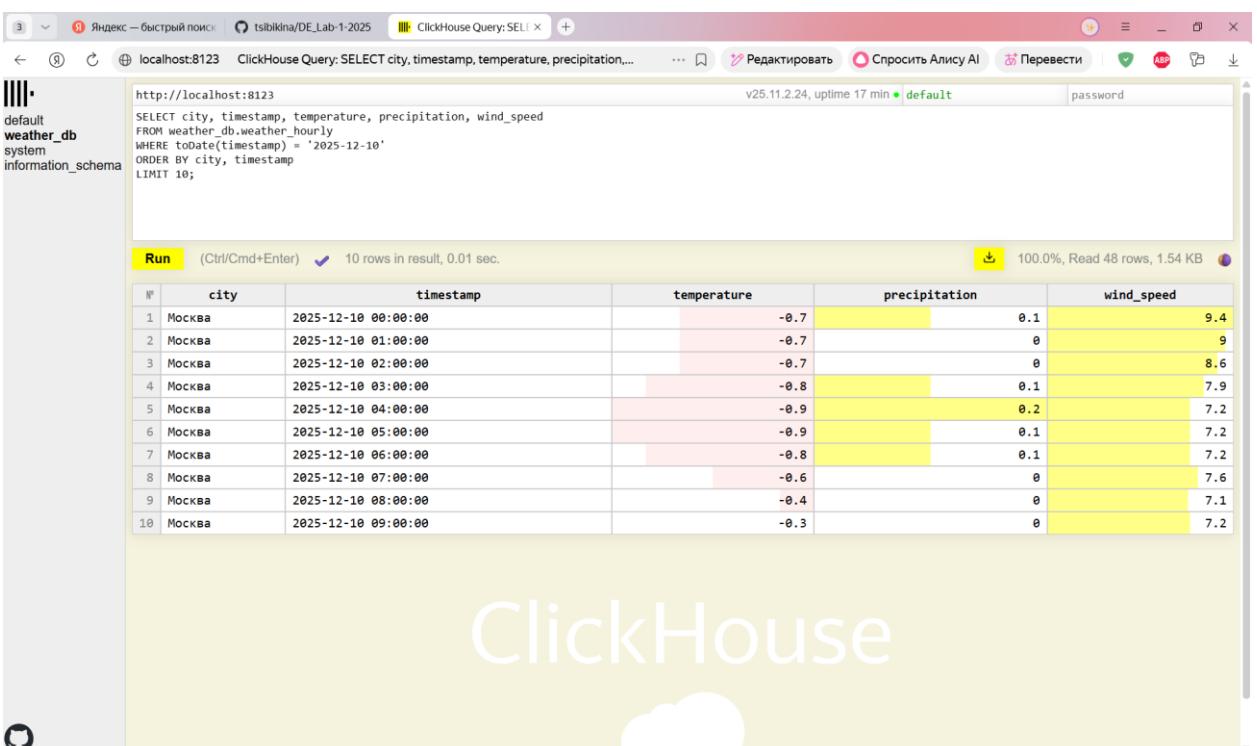
- Валидация HTTP-статуса ответа API через `response.raise_for_status()`.
- Механизм повторных попыток (`retries=3`) при извлечении данных.
- Неявная проверка структуры данных при трансформации.

Точки сбоя:

- Недоступность или изменение формата ответа у Open-Meteo API.
- Недоступность сервисов MinIO или ClickHouse из-за проблем с сетью или конфигурацией Docker.
- Отсутствие или некорректность секретов для Telegram (токен, `chat_id`).

Результаты работы пайплайна:

1. Содержимое в ClickHouse



The screenshot shows a ClickHouse query results window. The query is:

```
http://localhost:8123
SELECT city, timestamp, temperature, precipitation, wind_speed
FROM weather_db.weather_hourly
WHERE toDate(timestamp) = '2025-12-10'
ORDER BY city, timestamp
LIMIT 10;
```

The results table has columns: №, city, timestamp, temperature, precipitation, and wind_speed. The data is as follows:

№	city	timestamp	temperature	precipitation	wind_speed
1	Москва	2025-12-10 00:00:00	-0.7	0.1	9.4
2	Москва	2025-12-10 01:00:00	-0.7	0	9
3	Москва	2025-12-10 02:00:00	-0.7	0	8.6
4	Москва	2025-12-10 03:00:00	-0.8	0.1	7.9
5	Москва	2025-12-10 04:00:00	-0.9	0.2	7.2
6	Москва	2025-12-10 05:00:00	-0.9	0.1	7.2
7	Москва	2025-12-10 06:00:00	-0.8	0.1	7.2
8	Москва	2025-12-10 07:00:00	-0.6	0	7.6
9	Москва	2025-12-10 08:00:00	-0.4	0	7.1
10	Москва	2025-12-10 09:00:00	-0.3	0	7.2

Рисунок 1 - Содержимое запроса для таблицы `weather_hourly`

The screenshot shows the ClickHouse Query interface. On the left, there's a sidebar with database and schema names: default, weather_db, system, and information_schema. The main area displays a query window with the following content:

```
http://localhost:8123
SELECT *
FROM weather_db.weather_daily
WHERE date = '2025-12-10'
ORDER BY city;
```

Below the query window, a results table is shown:

#	city	date	min_temp	max_temp	avg_temp	total_precipitation	max_wind_speed
1	Москва	2025-12-10	-0.9	0.4	-0.25	2.2	10.2
2	Самара	2025-12-10	-7.2	-4.9	-6.05000000000001	0	16.3

At the bottom right of the results table, it says "100.0%, Read 2 rows, 124.00 B".

Рисунок 2 - Содержимое для запроса таблицы weather_daily

2. Содержимое бакета в MinIO

The screenshot shows the MinIO Object Browser interface. On the left, there's a sidebar with a 'Create Bucket' button and a 'Buckets' section containing 'weather-raw'. The main area shows the 'weather-raw' bucket details:

- Created on: Tue, Dec 09 2025 14:55:23 (GMT+4)
- Access: PRIVATE
- 5.2 KIB - 2 Objects

The object list shows two objects: 'Москва' and 'Самара'.

Name	Last Modified	Size
Москва		-
Самара		-

Рисунок 3 - Работа в MinIO

3. Perfect логи

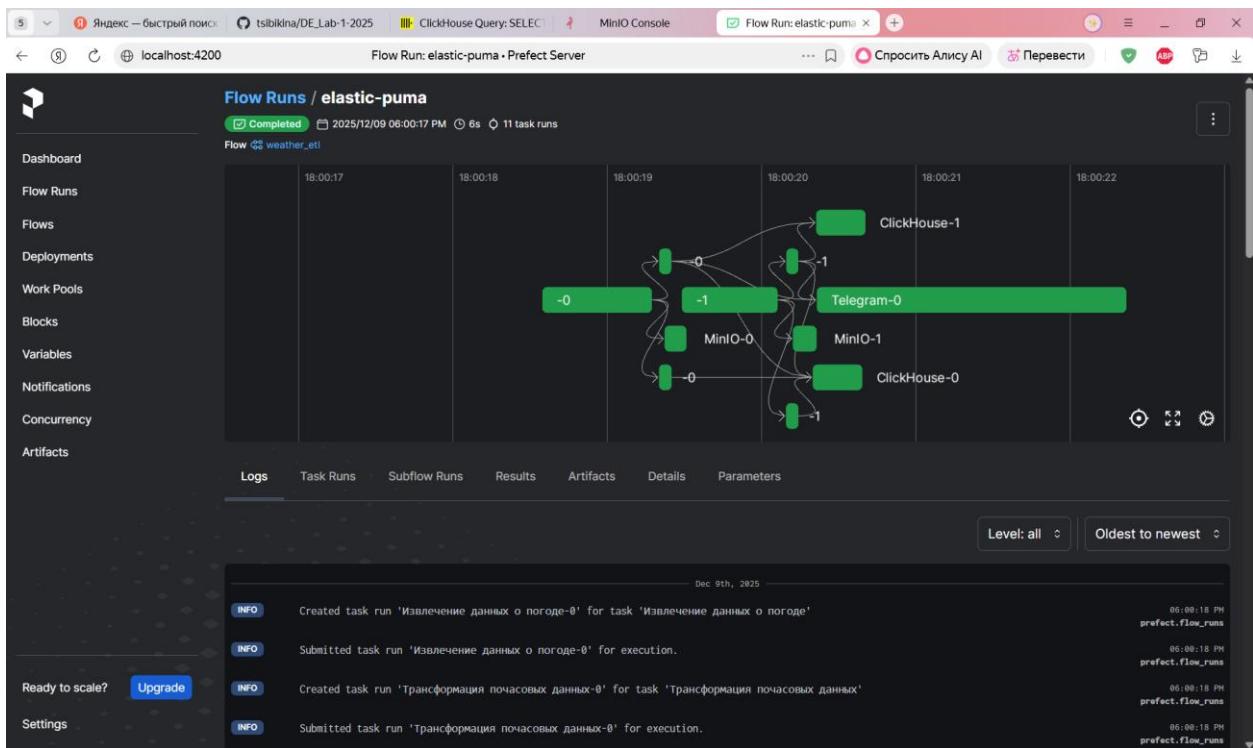


Рисунок 4 - Лог работы в Perfect

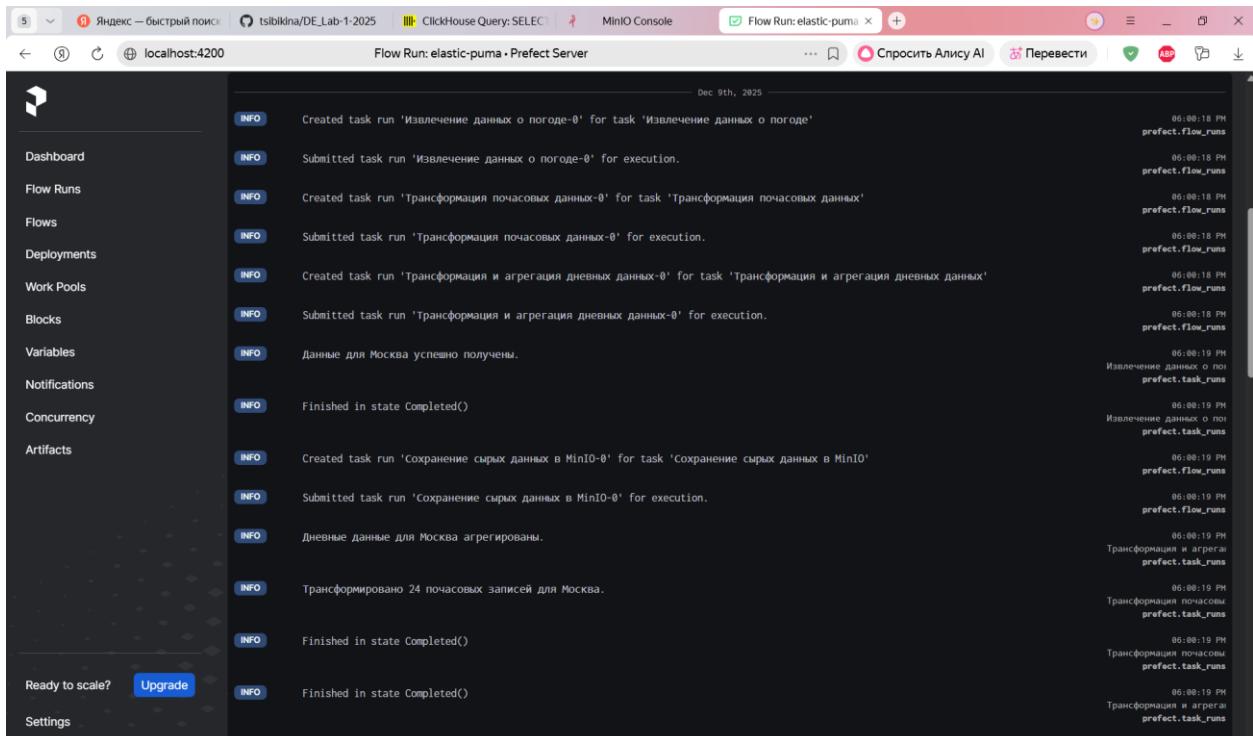


Рисунок 5 - Лог работы в Perfect

Flow Run: elastic-puma • Prefect Server

Dec 9th, 2025

Task	Log Message	Time
Submitted task run 'Загрузка данных в ClickHouse-0'	for execution.	06:00:20 PM prefect.flow_runs
Created task run 'Загрузка данных в ClickHouse-1'	for task 'Загрузка данных в Clickhouse'	06:00:20 PM prefect.flow_runs
Submitted task run 'Загрузка данных в ClickHouse-1'	for execution.	06:00:20 PM prefect.flow_runs
Created task run 'Отправка уведомления в Telegram-0'	for task 'Отправка уведомления в Telegram'	06:00:20 PM prefect.flow_runs
Submitted task run 'Отправка уведомления в Telegram-0'	for execution.	06:00:20 PM prefect.flow_runs
Сырые данные для Самара сохранены в MinIO: weather/Самара/2025-12-10.json		06:00:20 PM Сохранение сырых данных prefect.task_runs
Finished in state Completed()		06:00:20 PM Сохранение сырых данных prefect.task_runs
Загружено 48 строк в таблицу weather_hourly.		06:00:20 PM Загрузка данных в Click prefect.task_runs
Загружено 2 строки в таблицу weather_daily.		06:00:20 PM Загрузка данных в Click prefect.task_runs
Finished in state Completed()		06:00:20 PM Загрузка данных в Click prefect.task_runs
Finished in state Completed()		06:00:20 PM Загрузка данных в Click prefect.task_runs
Уведомление в Telegram успешно отправлено.		06:00:20 PM Отправка уведомления в prefect.task_runs
Finished in state Completed()		06:00:22 PM Отправка уведомления в prefect.task_runs
Finished in state Completed('All states completed.')		06:00:22 PM prefect.flow_runs

Рисунок 6- Лог работы в Perfect

4. Результат работы уведомления в Telegram

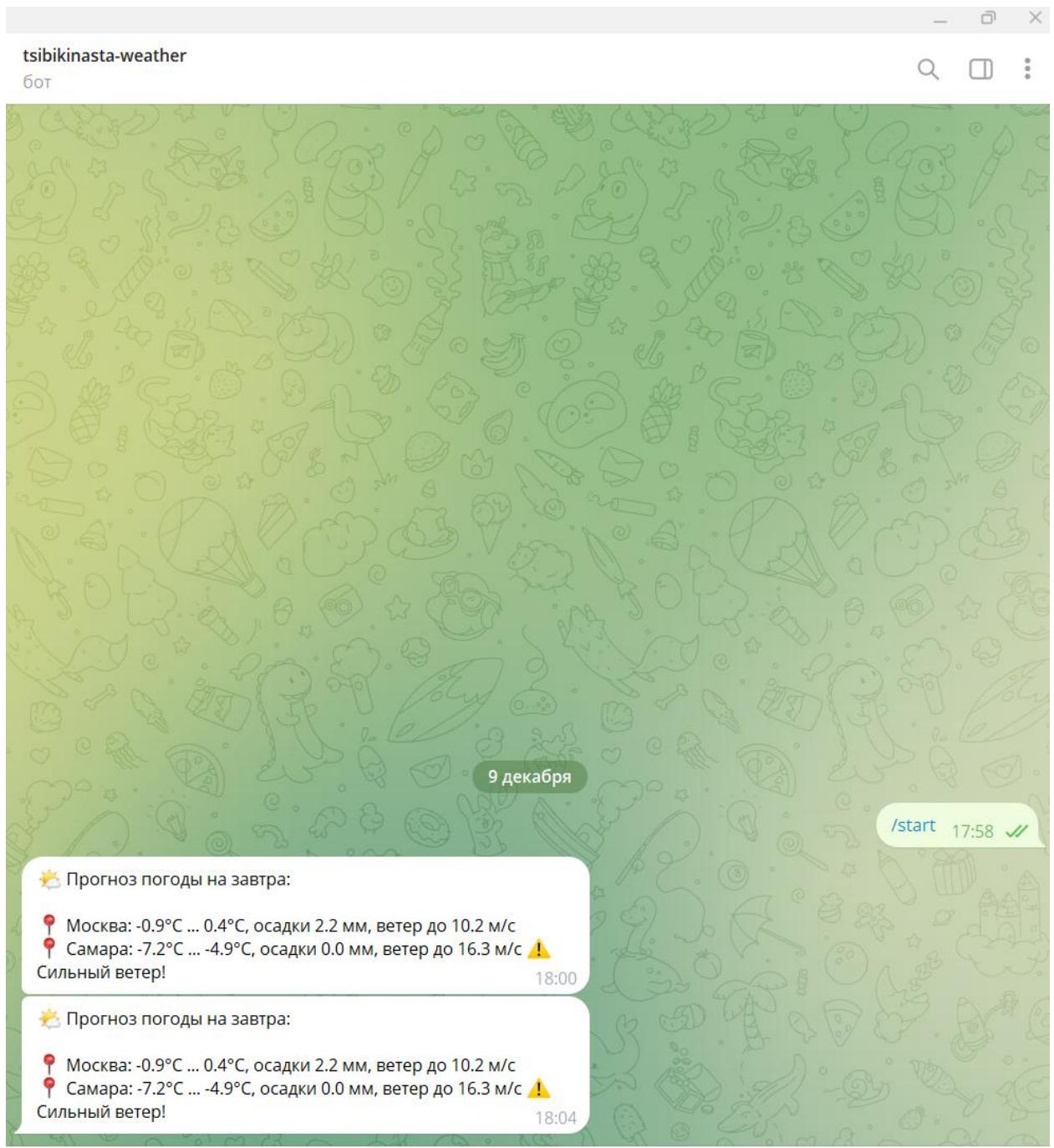


Рисунок 7 - Уведомления в Telegram

Вывод по работе и её дальнейшие улучшения.

Сложности возникли с настройкой и отладкой всего окружения, а не самой логикой ETL-процесса. Из ключевых трудностей - это конфликты версий библиотек и настройка уведомлений в telegram.

Можно улучшить программу добавив виртуальное окружения, чтобы убрать проблему с конфликтами библиотек и может сделать более детальную обработку ошибок.