

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 технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № 1

по курсу
Инженерия данных

Группа 6232-010402D

Студент _____ Л.А. Абакумов
(подпись)

Преподаватель,

к.т.н. _____ Р.А. Парингер
(подпись)

Самара 2025

1. ЗАДАНИЕ

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)
4. Извлечь почасовые значения и нормализовать для таблицы `weather_hourly`
5. Посчитать дневную статистику (`min`, `max`, `avg` температуру и количество осадков) и подготовить для сохранения в таблице `weather_daily`
6. (Load) Загрузить преобразованные данные в соответствующие таблицы ClickHouse
7. Автоматически отправить уведомления в Telegram с кратким прогнозом на завтра и предупреждать о сильном ветре/осадках
8. (Опционально) Реализовать обработку различных ошибок.

2. ОПИСАНИЕ РАБОТЫ

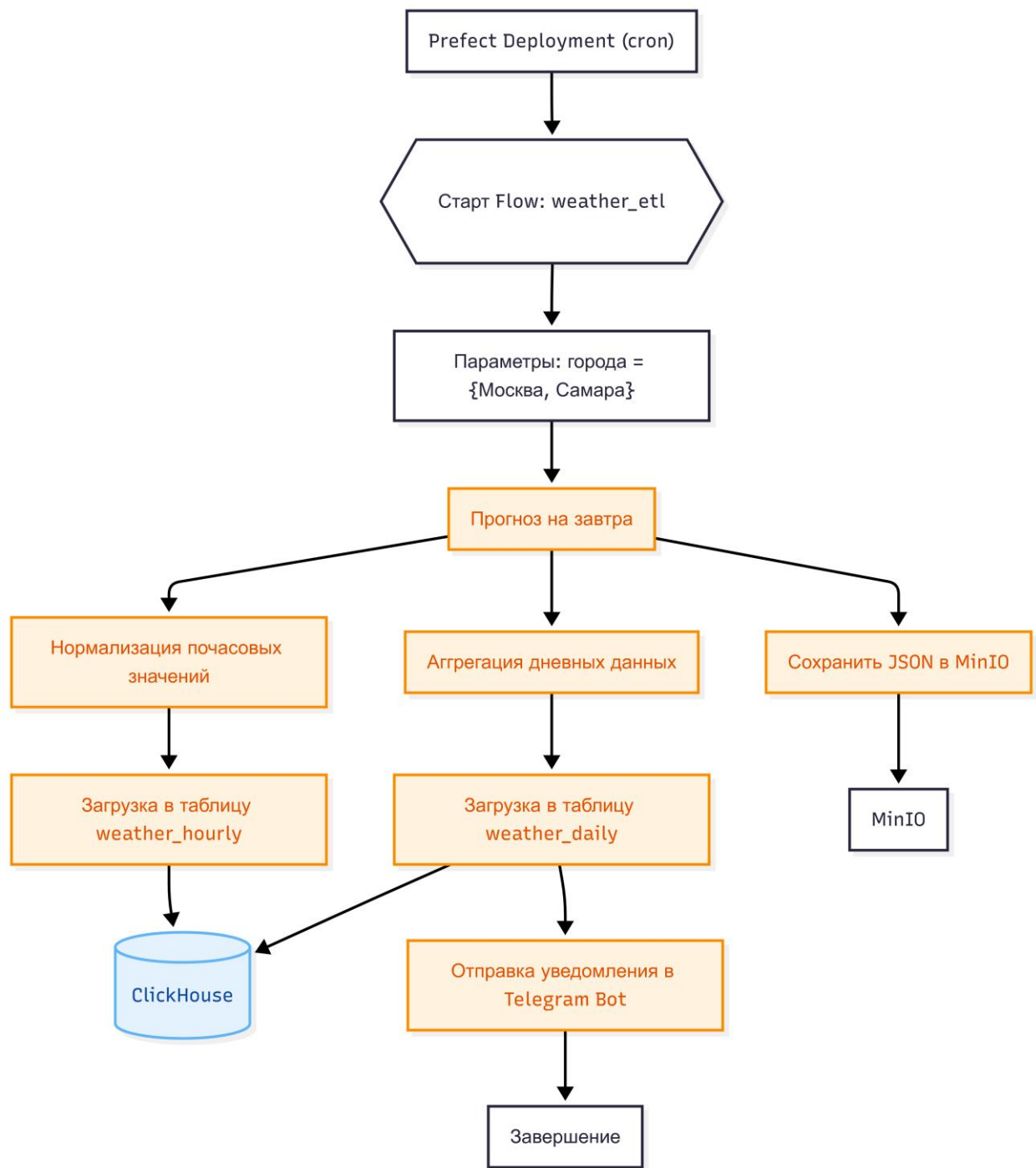


Рисунок 1 – Схема пайплайна

Разработанная система реализует ETL-пайплайн с использованием Prefect для оркестрации задач и планирования запусков по cron. Поток weather_etl загружает данные о погоде на следующий день для городов Москва и Самара из API Open-Meteo, сохраняет их в необработанном виде в S3-совместимом хранилище MinIO, преобразует в почасовой формат, агрегирует

в суточную статистику и загружает результат в аналитическую СУБД ClickHouse в таблицы `weather_hourly` и `weather_daily`. Краткий прогноз с предупреждениями о неблагоприятных погодных условиях автоматически отправляется пользователю через Telegram Bot API. Все инфраструктурные компоненты (MinIO и ClickHouse) развёрнуты в контейнерах с помощью Docker Compose, что обеспечивает переносимость и воспроизводимость окружения.

В качестве источника исходных данных используется погодное API Open-Meteo. Для каждого города выполняется HTTP-запрос методом GET к эндпоинту `https://api.open-meteo.com/v1/forecast` с набором параметров: координаты Москвы и Самары (широта и долгота), список почасовых переменных (`temperature_2m`, `precipitation`, `windspeed_10m`, `winddirection_10m`), временная зона (`timezone=Europe/Moscow`) и даты `start_date/end_date`, соответствующие завтрашнему дню. API возвращает ответ в формате JSON, содержащий массивы почасовых значений, которые далее используются как для хранения «сырых» данных, так и для построения производных агрегатов.

На этапе извлечения данных поток `weather_etl` последовательно вызывает задачу `fetch_weather` для Москвы и Самары. Для каждого города выполняется запрос к Open-Meteo на один день вперёд (прогноз на завтра) с необходимым набором почасовых параметров, а полученный JSON-ответ логируется и передаётся на дальнейшую обработку.

На этапе преобразования исходный JSON-ответ Open-Meteo сначала приводится к почасовой табличной форме: для каждого города формируется набор записей с меткой времени и основными погодными параметрами, после чего данные фильтруются по целевой дате, чтобы оставить только значения прогноза на завтра. На основе этого набора вычисляются суточные агрегаты

по каждому городу: минимальная, максимальная и средняя температура, суммарное количество осадков за сутки и максимальная скорость ветра. Почасовой набор используется для загрузки в таблицу `weather_hourly`, а суточные агрегаты — для таблицы `weather_daily` и формирования предупреждающих сообщений в Telegram.

Этап загрузки реализован в двух независимых ветках. Во-первых, исходный JSON-ответ Open-Meteo сериализуется и записывается в MinIO в бакет `weather-raw` в виде файлов `moscow-YYYY-MM-DD.json` и `samara-YYYY-MM-DD.json`, что обеспечивает долгосрочное хранение «сырых» данных. Во-вторых, нормализованные почасовые и суточные таблицы конвертируются в списки строк и с помощью драйвера `clickhouse-connect` вставляются в ClickHouse в таблицы `weather_hourly` и `weather_daily`, что позволяет выполнять по ним аналитические запросы.

На завершающем этапе объединяются суточные агрегаты и почасовые данные для каждого города, после чего формируется текстовое уведомление с указанием даты прогноза, минимальной, максимальной и средней температуры, суммарных осадков и максимальной скорости ветра. Если суточный объём осадков или максимальная скорость ветра превышают заданные пороги, к сообщению добавляются дополнительные строчки-предупреждения о сильном дожде и/или ветре. Готовый текст отправляется пользователю через Telegram Bot API с использованием параметров бота, заданных в конфигурации.

Для обеспечения качества данных на этапе извлечения проверяется HTTP-статус ответа Open-Meteo; при ошибках сетевого уровня или некорректном статусе запрос автоматически повторяется несколько раз за счёт механизма `retries` в задачах Prefect. В задачах преобразования предусмотрена проверка наличия блока `hourly` и защита от пустых `DataFrame`: при отсутствии

данных на нужную дату модуль логирует проблему и пропускает соответствующие шаги загрузки, не прерывая работу всего пайплайна. Основными точками отказа являются недоступность API, изменение формата JSON, ошибки подключения к MinIO или ClickHouse и некорректная конфигурация Telegram-бота; для этих случаев предусмотрено логирование и частичная обработка, при которой сырые ответы всё равно сохраняются в MinIO.

В веб-интерфейсе Prefect отображаются регулярные запуски потока `weather_etl` по расписанию, для каждого запуска доступен детальный лог выполнения всех задач. На временной диаграмме видно, что пайплайн обрабатывает последовательно для двух городов и завершается в состоянии Completed без ошибок.

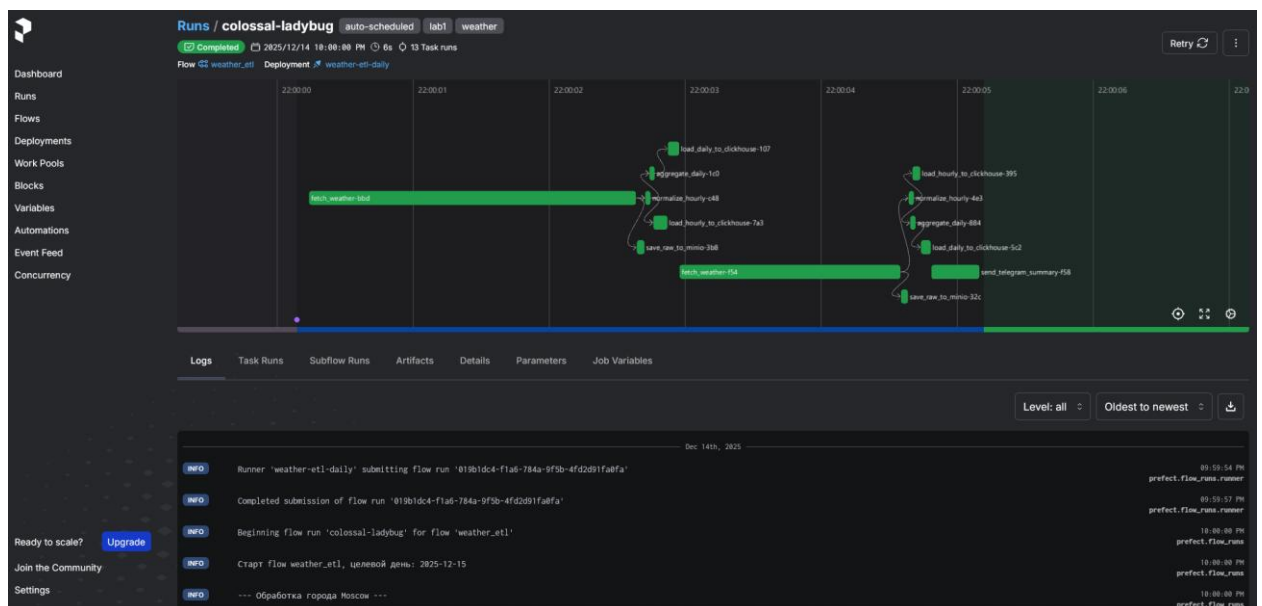


Рисунок 2 – Prefect UI

Для проверки корректности загрузки выполнены выборки из таблиц `weather_hourly` и `weather_daily` через веб-интерфейс ClickHouse. В `weather_hourly` отображаются почасовые значения температуры, осадков и ветра по городам, а в `weather_daily` – агрегированные суточные показатели для каждого города на выбранную дату.

http://localhost:8123 v25.11.2.24, uptime 2 hr default

```
SELECT *
FROM weather_hourly
ORDER BY ts, city
LIMIT 20;
```

Run (Ctrl/Cmd+Enter) 20 rows in result, 0.05 sec. 100.0%, Read 672 rows, 17.47 KB

#	city	ts	temperature	precipitation	wind_speed	wind_direction
1	Moscow	2025-12-11 00:00:00	-0.1	0	6.3	193
2	Moscow	2025-12-11 00:00:00	-0.1	0	6.3	193
3	Moscow	2025-12-11 00:00:00	-0.1	0	6.3	193
4	Samara	2025-12-11 00:00:00	-7.8	0	11.1	155
5	Samara	2025-12-11 00:00:00	-7.8	0	11.1	155
6	Samara	2025-12-11 00:00:00	-7.8	0	11.1	155
7	Moscow	2025-12-11 01:00:00	-0	0	7	192
8	Moscow	2025-12-11 01:00:00	-0	0	7	192
9	Moscow	2025-12-11 01:00:00	-0	0	7	192
10	Samara	2025-12-11 01:00:00	-7.9	0	11.5	160
11	Samara	2025-12-11 01:00:00	-7.9	0	11.5	160
12	Samara	2025-12-11 01:00:00	-7.9	0	11.5	160
13	Moscow	2025-12-11 02:00:00	0.1	0	7.2	198
14	Moscow	2025-12-11 02:00:00	0.1	0	7.2	198
15	Moscow	2025-12-11 02:00:00	0.1	0	7.2	198
16	Samara	2025-12-11 02:00:00	-8.2	0	12	159
17	Samara	2025-12-11 02:00:00	-8.2	0	12	159
18	Samara	2025-12-11 02:00:00	-8.2	0	12	159
19	Moscow	2025-12-11 03:00:00	0.2	0	7.4	194
20	Moscow	2025-12-11 03:00:00	0.2	0	7.4	194

Рисунок 3 – ClickHouse UI таблица weather_hourly

http://localhost:8123 v25.11.2.24, uptime 2 hr default

```
SELECT *
FROM weather_daily
ORDER BY date, city;
```

Run (Ctrl/Cmd+Enter) 28 rows in result, 0.01 sec. 100.0%, Read 28 rows, 896.00 B

#	city	date	temp_min	temp_max	temp_avg	precipitation_sum
1	Moscow	2025-12-11	-0.1	3.1	1.6916667	4.6
2	Moscow	2025-12-11	-0.1	3.1	1.6916667	4.6
3	Moscow	2025-12-11	-0.1	3.1	1.6916667	4.6
4	Samara	2025-12-11	-8.4	-4.4	-6.4791665	0.2
5	Samara	2025-12-11	-8.4	-4.4	-6.4791665	0.2
6	Samara	2025-12-11	-8.4	-4.4	-6.4791665	0.2
7	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
8	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
9	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
10	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
11	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
12	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
13	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
14	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
15	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
16	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
17	Moscow	2025-12-15	-8.9	-6.3	-7.608333	0
18	Samara	2025-12-15	-10.4	-9	-9.725	0.5
19	Samara	2025-12-15	-10.4	-9	-9.725	0.5
20	Samara	2025-12-15	-10.4	-9	-9.725	0.5
21	Samara	2025-12-15	-10.4	-9	-9.725	0.5

Рисунок 4 – ClickHouse UI таблица weather_daily

В хранилище MinIO в бакете weather-raw для каждого запуска создаются файлы с исходными ответами Open-Meteo по городам. Имена файлов включают название города и дату прогноза, что облегчает поиск и повторную обработку данных.

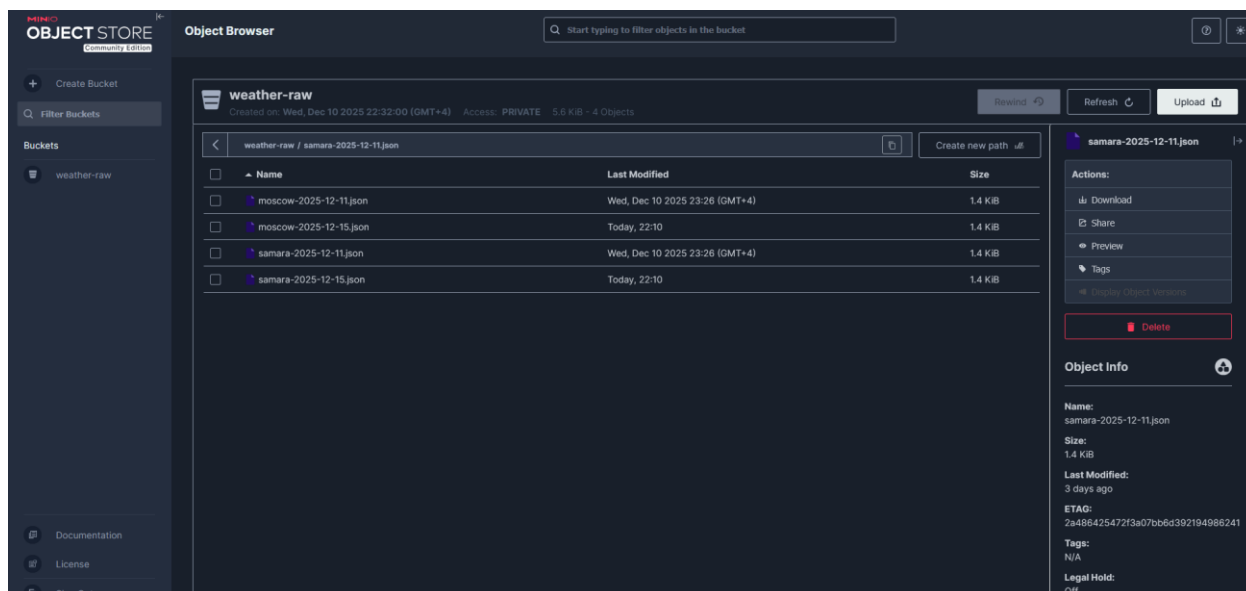


Рисунок 5 – MinIO UI

Telegram-бот автоматически отправляет краткий прогноз на завтра по каждому городу с указанием минимальной, максимальной и средней температуры, суммарных осадков и максимальной скорости ветра. При превышении пороговых значений осадков или ветра в сообщении появляются дополнительные строки-предупреждения о неблагоприятных погодных условиях.



Рисунок 6 – Чат Telegram-бота

ЗАКЛЮЧЕНИЕ

В ходе работы был реализован и развёрнут полный ETL-пайплайн для сбора, обработки и хранения прогнозов погоды с последующей отправкой уведомлений пользователю. Система автоматически извлекает данные из Open-Метео для городов Москва и Самара, сохраняет сырые JSON в MinIO, преобразует их в почасовой и суточный формат, загружает в ClickHouse и с заданной периодичностью отправляет сообщения через Telegram-бота. Наиболее сложными оказались настройка окружения (Docker, ClickHouse, MinIO), согласование версий Python-библиотек.