

Департамент образования города Москвы

Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

**Практическая работа на вебинаре 21.03.2025
по дисциплине «Проектный практикум по разработке ETL-решений»**

Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
St_88

Москва
2025

ВАРИАНТ 1

Вариант	Задание 1	Задание 2	Задание 3
1	Получить прогноз погоды в Москве на 3 дня	Сохранить поля: date, avgtemp_c	Сохранить как CSV

ХОД РАБОТЫ

Регистрация на сайте Free Weather Api (<https://www.weatherapi.com/>) для получения api key:



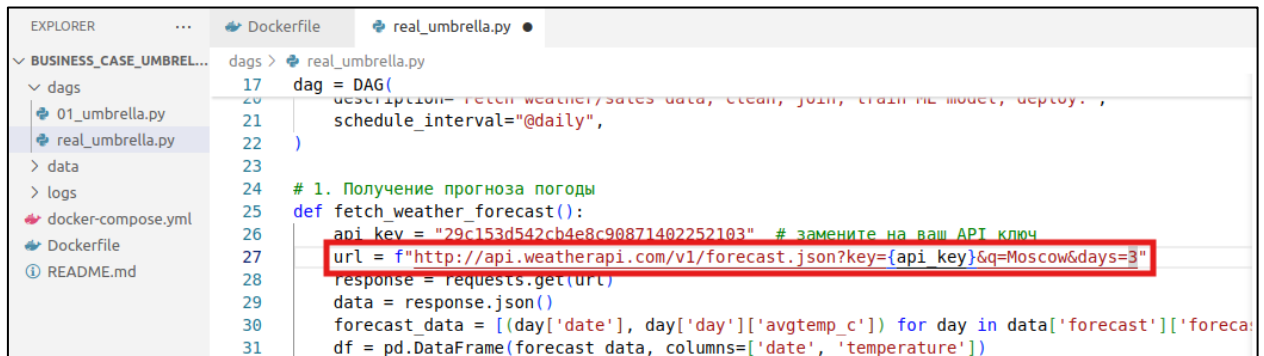
В подготовленной заранее виртуальной машине необходимо загрузить сопутствующие файлы бизнес-кейса «Umbrella».

В файле дага real_umbrella.py требуется подставить собственный api key:

```
EXPLORER    ...    Dockerfile    real_umbrella.py
BUSINESS_CASE_UMBREL...
  dags
    01_umbrella.py
    real_umbrella.py
  data
  logs
  docker-compose.yml
  Dockerfile
  README.md

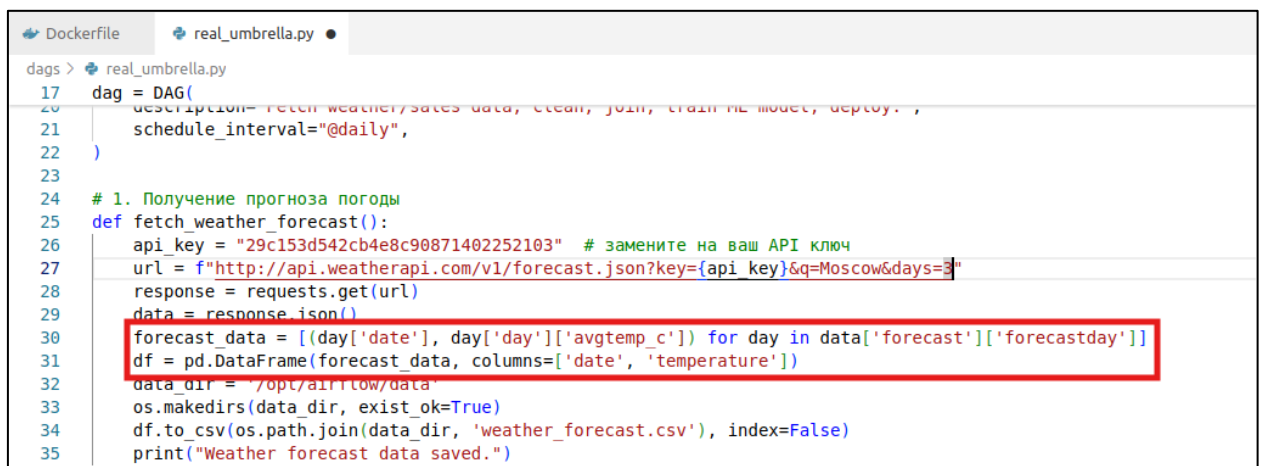
dags > real_umbrella.py
17 dag = DAG(
20     description='Fetch weather/sales data, clean, join, train ML model, deploy.',
21     schedule_interval="@daily",
22 )
23
24 # 1. Получение прогноза погоды
25 def fetch_weather_forecast():
26     api_key = "29c153d542cb4e8c90871402252103" # замените на ваш API ключ
27     url = f"http://api.weatherapi.com/v1/forecast.json?key={api_key}&q=Moscow&days=3"
28     response = requests.get(url)
29     data = response.json()
30     forecast_data = [(day['date'], day['day']['avgtemp_c']) for day in data['forecast']['forecastday']]
31     df = pd.DataFrame(forecast_data, columns=['date', 'temperature'])
32     data_dir = '/opt/airflow/data'
33     os.makedirs(data_dir, exist_ok=True)
34     df.to_csv(os.path.join(data_dir, 'weather_forecast.csv'), index=False)
35     print("Weather forecast data saved.")
```

Далее по заданию необходимо получить прогноз погоды для Москвы на 3 дня. Для этого меняем url (ключи: q = локация, days = количество дней, для которых нужен прогноз):



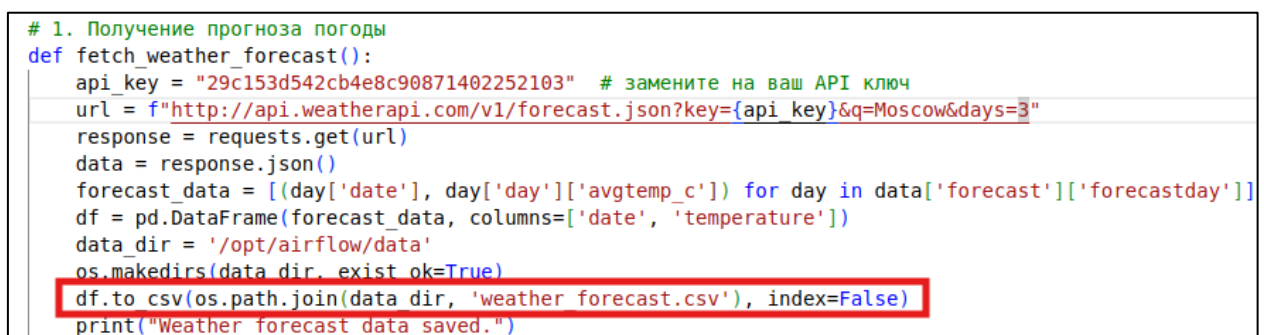
```
EXPLORER  ...  Dockerfile  real_umbrella.py
BUSINESS_CASE_UMBREL...
  dags
    01_umbrella.py
    real_umbrella.py
  data
  logs
  docker-compose.yml
  Dockerfile
  README.md
dags > real_umbrella.py
17 dag = DAG(
20     description='Fetch weather/sales data, clean, join, train ML model, deploy.',
21     schedule_interval="@daily",
22 )
23
24 # 1. Получение прогноза погоды
25 def fetch_weather_forecast():
26     api_key = "29c153d542cb4e8c90871402252103" # замените на ваш API ключ
27     url = f"http://api.weatherapi.com/v1/forecast.json?key={api_key}&q=Moscow&days=3"
28     response = requests.get(url)
29     data = response.json()
30     forecast_data = [(day['date'], day['day']['avgtemp_c']) for day in data['forecast']['forecastday']]
31     df = pd.DataFrame(forecast_data, columns=['date', 'temperature'])
```

Также по варианту необходимо сохранить поля: date, avgtemp_c, которые будут соответственно записываться в колонки «date» и «temperature»:



```
Dockerfile  real_umbrella.py
dags > real_umbrella.py
17 dag = DAG(
20     description='Fetch weather/sales data, clean, join, train ML model, deploy.',
21     schedule_interval="@daily",
22 )
23
24 # 1. Получение прогноза погоды
25 def fetch_weather_forecast():
26     api_key = "29c153d542cb4e8c90871402252103" # замените на ваш API ключ
27     url = f"http://api.weatherapi.com/v1/forecast.json?key={api_key}&q=Moscow&days=3"
28     response = requests.get(url)
29     data = response.json()
30     forecast_data = [(day['date'], day['day']['avgtemp_c']) for day in data['forecast']['forecastday']]
31     df = pd.DataFrame(forecast_data, columns=['date', 'temperature'])
32     data_dir = '/opt/airflow/data'
33     os.makedirs(data_dir, exist_ok=True)
34     df.to_csv(os.path.join(data_dir, 'weather_forecast.csv'), index=False)
35     print("Weather forecast data saved.")
```

Помимо этого, в соответствии с заданием требуется, чтобы прогноз сохранился в CSV-файле:



```
# 1. Получение прогноза погоды
def fetch_weather_forecast():
    api_key = "29c153d542cb4e8c90871402252103" # замените на ваш API ключ
    url = f"http://api.weatherapi.com/v1/forecast.json?key={api_key}&q=Moscow&days=3"
    response = requests.get(url)
    data = response.json()
    forecast_data = [(day['date'], day['day']['avgtemp_c']) for day in data['forecast']['forecastday']]
    df = pd.DataFrame(forecast_data, columns=['date', 'temperature'])
    data_dir = '/opt/airflow/data'
    os.makedirs(data_dir, exist_ok=True)
    df.to_csv(os.path.join(data_dir, 'weather_forecast.csv'), index=False)
    print("Weather forecast data saved.")
```

Далее также лучше сразу настроить данные о продажах. В данном случае будут сведения о прогнозе на ближайшие 3 дня (21 – 23 марта 2025 года) + настроим сведения о продажах примерные (исходя из реальных облачности и температур), будем учитывать, что это средняя торговая сеть, а не маленький ларек:



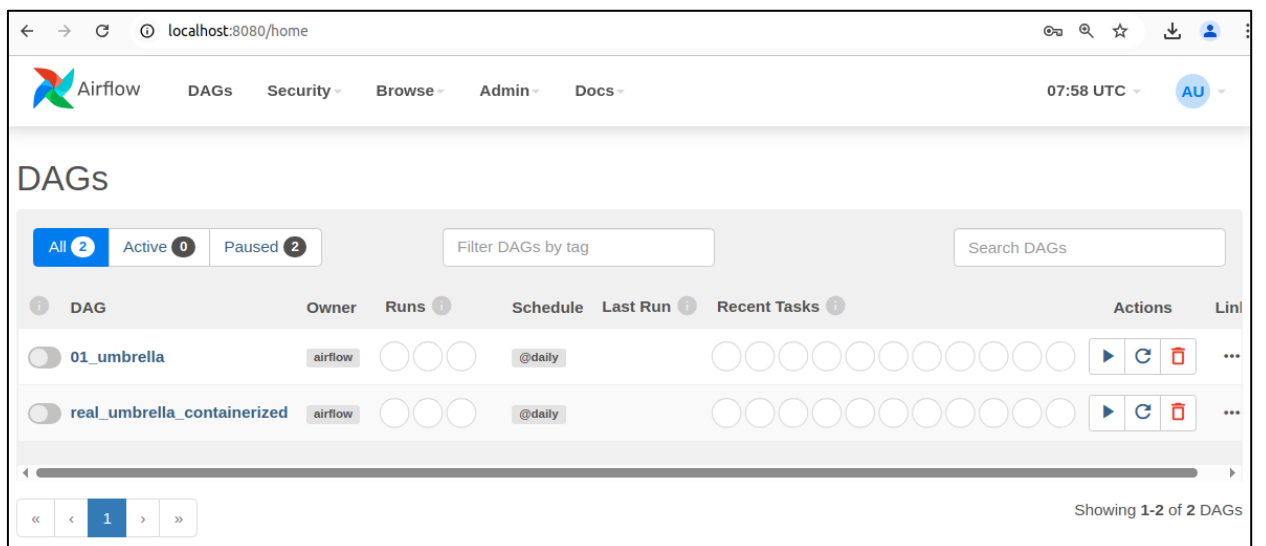
```
Dockerfile  real_umbrella.py
dags > real_umbrella.py
38 def clean_weather_data():
44
45 # 3. Получение данных продаж
46 def fetch_sales_data():
47     sales_data = {
48         'date': ['2025-03-21', '2025-03-22', '2025-03-23'],
49         'sales': [117, 32, 46]
50     }
```

После всех внесенных изменений в файл дага `real_umbrella.py` необходимо их сохранить, после чего можно билдить и запускать контейнеры:

```
dev@dev-vm:~/workshop-on-ETL-main/business_case_umbrella_25$ sudo docker build -t custom-airflow:2.0.0-python3.8 .
2025/03/21 10:48:10 in: [{}string{}]
2025/03/21 10:48:10 Parsed entitlements: []
[+] Building 147.8s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                docker:default 0.2s
=> => transferring dockerfile: 491B                                              0.1s
=> [internal] load metadata for docker.io/apache/airflow:2.0.0-python3.8        4.5s
=> [internal] load .dockerignore                                                 0.1s
=> => transferring context: 2B                                                  0.0s
```

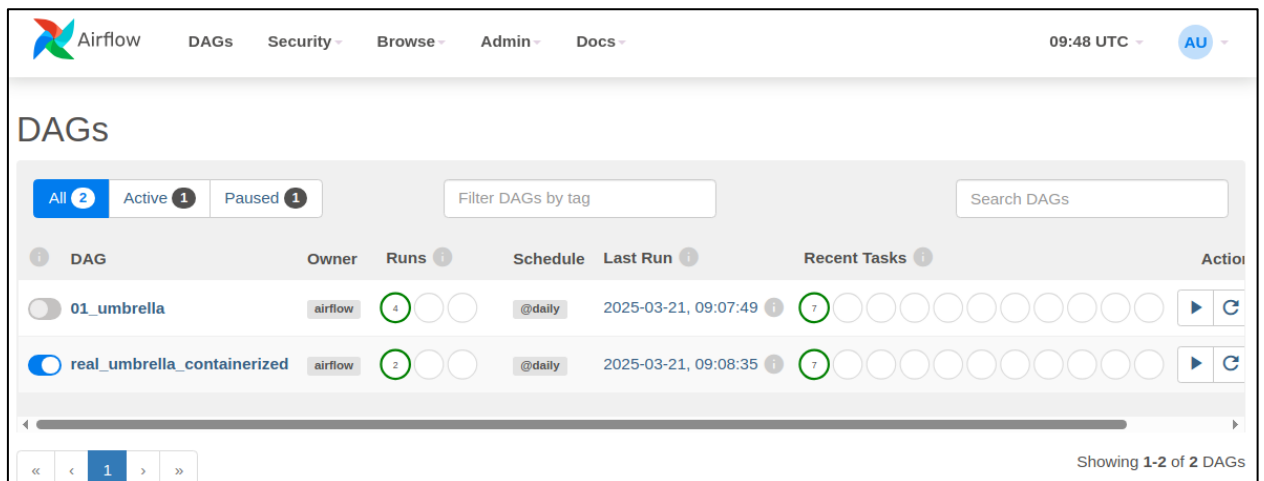
```
dev@dev-vm:~/workshop-on-ETL-main/business_case_umbrella_25$ sudo docker compose up --build
[+] Running 12/12
  ✓ postgres Pulled                                20.7s
  ✓ 1f3e46996e29 Already exists                    0.0s
  ✓ 47e20ba03731 Pull complete                     2.1s
  ✓ 101b82465a4f Pull complete                     2.3s
  ✓ 319529a7ccb0 Pull complete                     2.4s
  ✓ c2f9392cfd4c Pull complete                     2.5s
  ✓ 4e04446ce95d Pull complete                    15.0s
  ✓ 47bfe778b869 Pull complete                     15.1s
  ✓ b1d66b287aa8 Pull complete                     15.2s
  ✓ 7865e52a4759 Pull complete                     15.2s
  ✓ 7d75f14147c2 Pull complete                     15.3s
  ✓ 11052a5424e7 Pull complete                     15.4s
[+] Running 7/7
  ✓ Network business_case_umbrella_25_default      Created          0.3s
  ✓ Volume "business_case_umbrella_25_postgres_data" Create... 0.0s
  ✓ Volume "business_case_umbrella_25_logs"        Created          0.0s
  ✓ Container business_case_umbrella_25-postgres-1 Created         1.0s
  ✓ Container business_case_umbrella_25-init-1     Created          0.2s
  ✓ Container business_case_umbrella_25-webserver-1 Created         0.2s
  ✓ Container business_case_umbrella_25-scheduler-1 Created         0.2s
Attaching to init-1, postgres-1, scheduler-1, webserver-1
```

Проверка доступности Airflow (на порту 8080) – успешно:



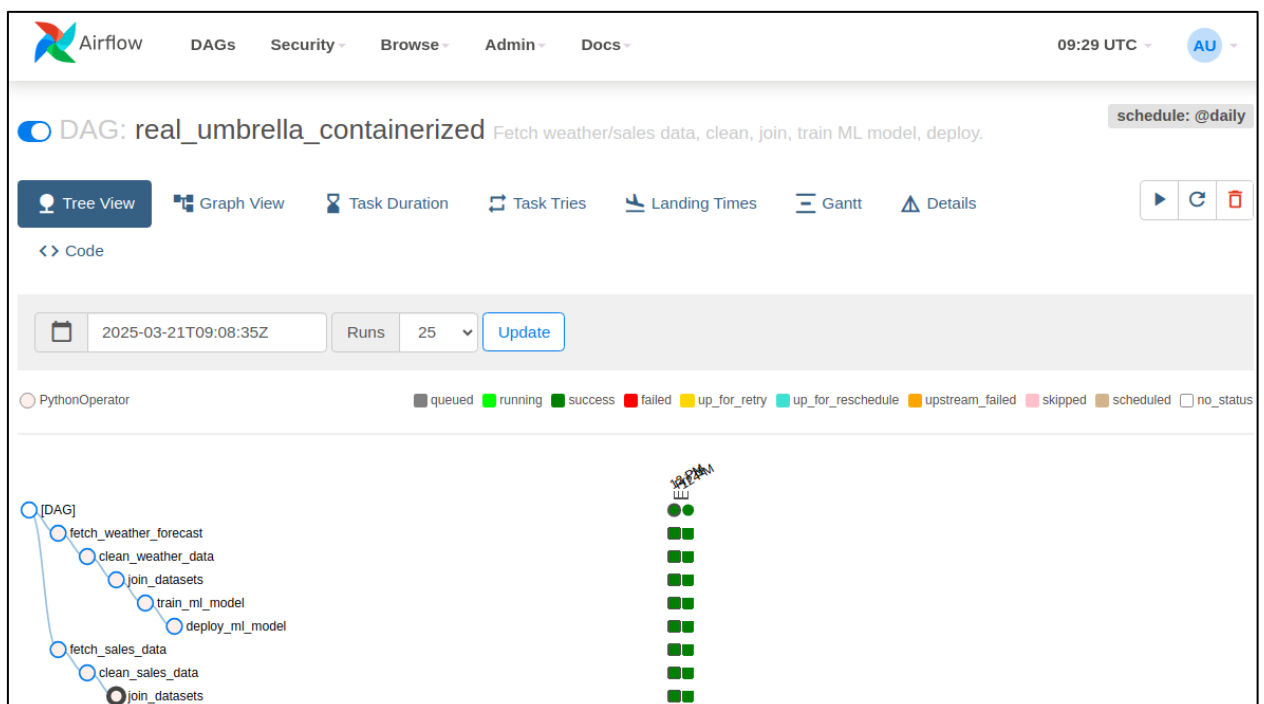
ОСНОВНЫЕ ЭЛЕМЕНТЫ ИНТЕРФЕЙСА AIRFLOW

Главная страница при входе представляет из себя список всех доступных дагов:

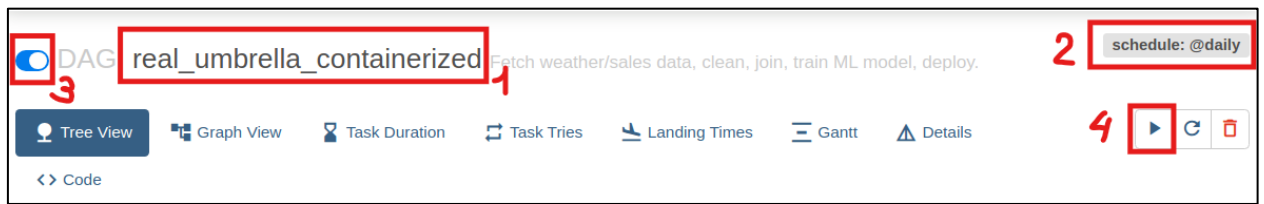


По каждому дагу отображается основная информация по нему: запущен ли, наименование, владелец, количество выполненных раз (при том, зеленым отображаются успешно выполненные даги, красным – в случае ошибки), также частота выполнения, время последнего запуска и количество задач в даге.

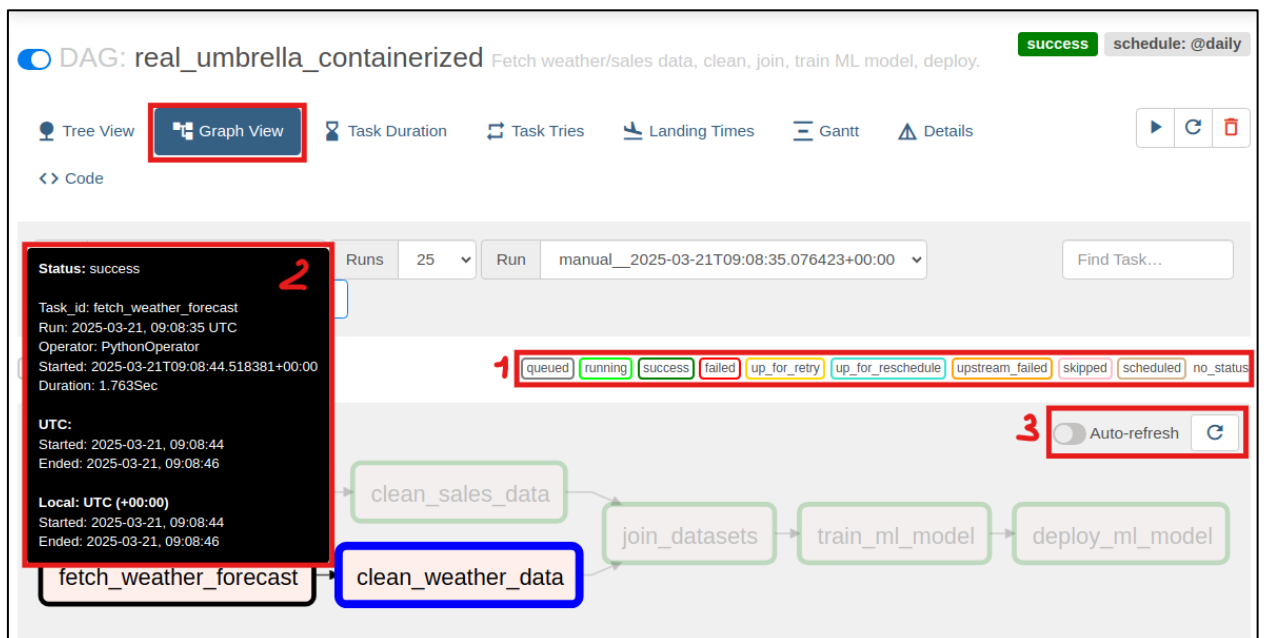
Информационная карта дага выглядит следующим образом:



Соответственно, на верхней панели отображается наименование графа (1) и интервал его запуска (2). Также, доступен свитч для включения/выключения дага (3) и кнопка для триггера дага (4):



Далее ниже доступна вариация представления графа: в виде дерева задач, в виде графа, просмотр информации о длительности выполнения и другое. Наиболее удобным вариантом для меня является отображение в виде графа:

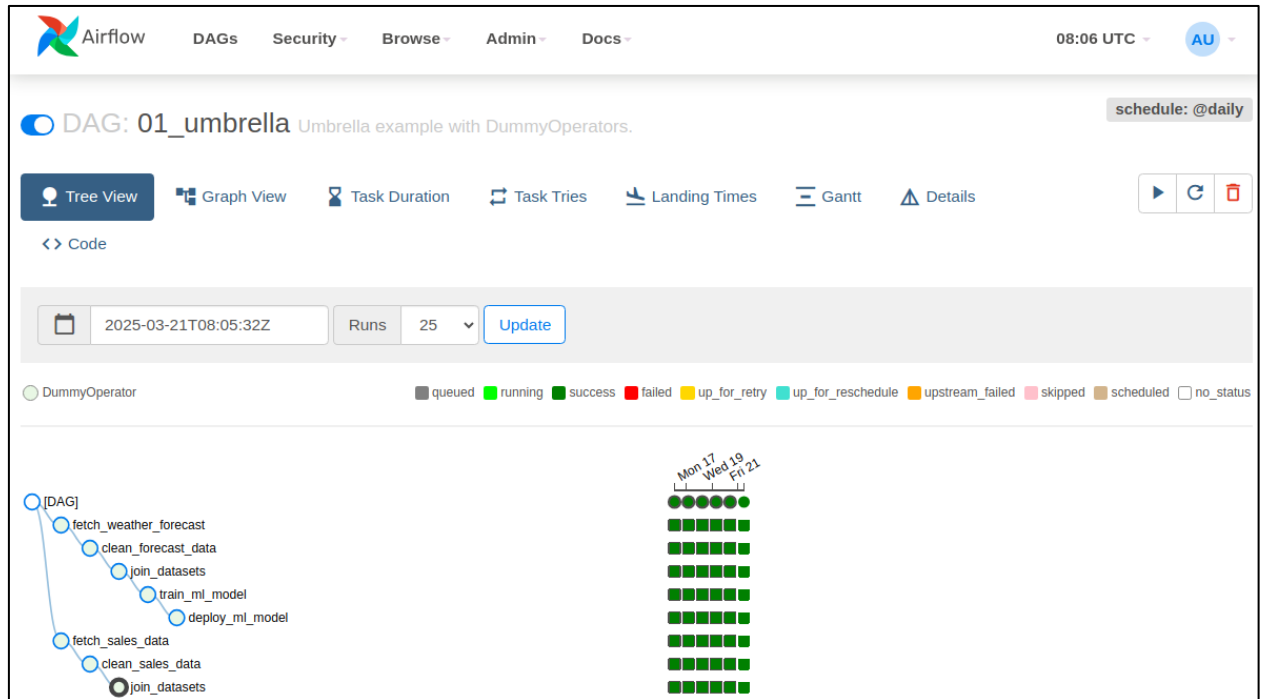


В данном случае отображаются задачи в виде графа. Каждая задача подсвечивается цветом, соответствующим текущему статусу выполнения (1). Помимо этого, при наведении на task в открывшемся модальном окошке (2) можно изучить всю информацию о нем: id, оператор, время запуска и длительность выполнения.

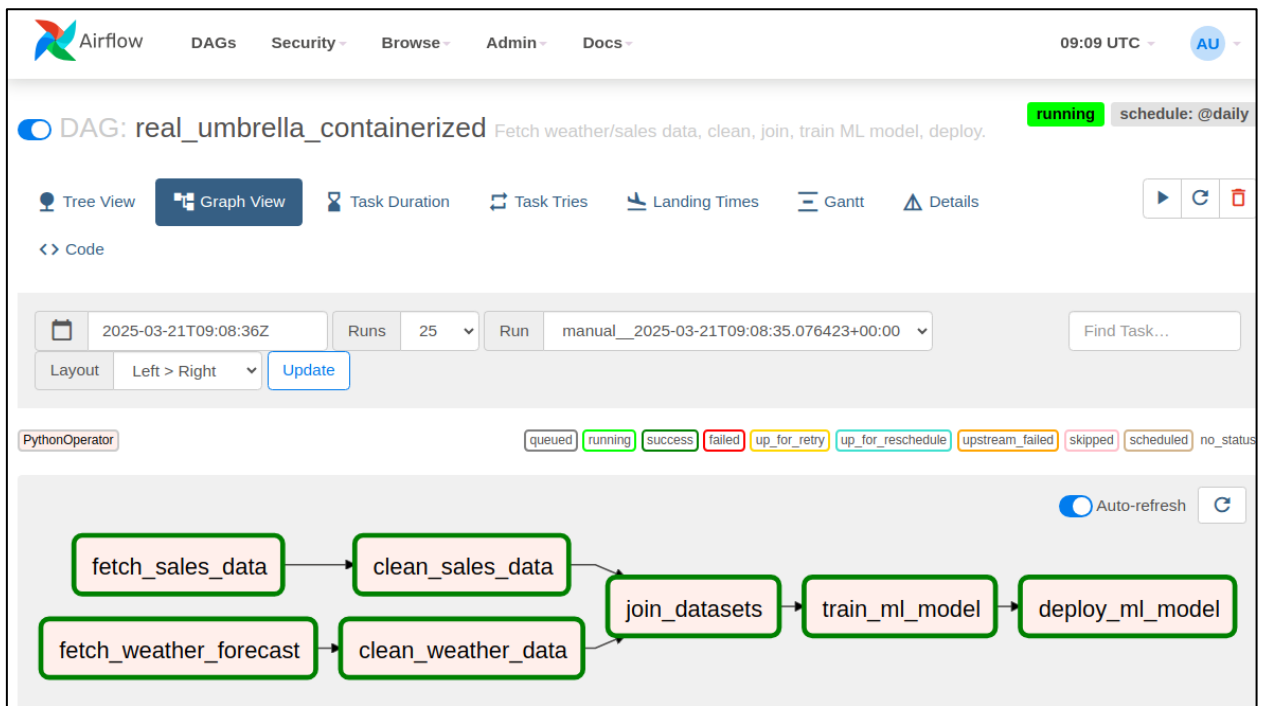
Для отображения актуальных статусов выполнения задач можно настроить автообновление, либо же запустить принудительное обновление (3).

Вернемся к выполнению практического задания.

Запускаем модельный даг, иллюстрирующий вариант использования бизнес-кейса «Umbrella» в модельном виде, проверяем, что все отработывает и ошибок не возникает:



Запуск дага `real_umbrella.py`, иллюстрирующего вариант использования бизнес-кейса «Umbrella», который автоматизирует весь pipeline обработки погодных и продажных данных, обучения модели машинного обучения и её «развёртывания»:

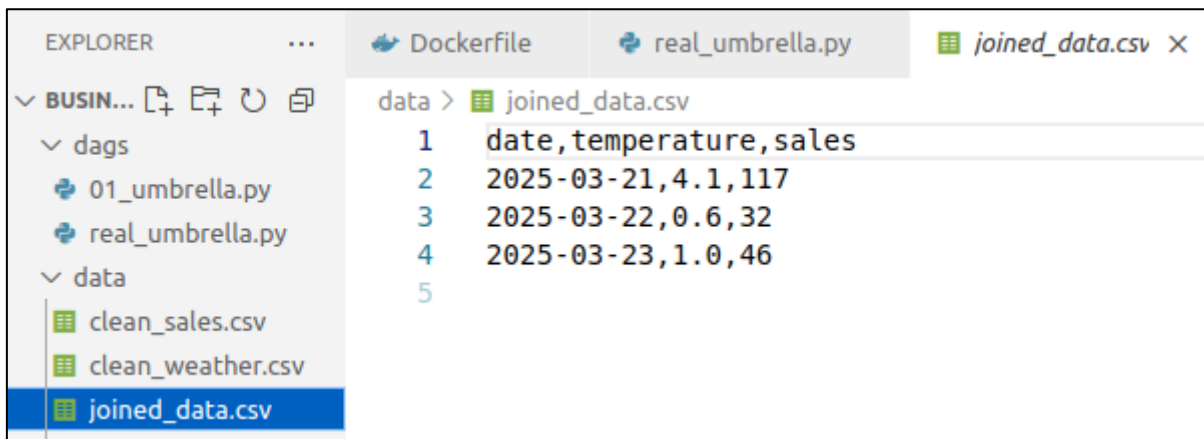


По итогам успешного выполнения графа получили данные о прогнозе погоды в Москве на 3 дня (файл в формате CSV, что требовалось по индивидуальному заданию):

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a directory structure with folders 'BUSIN...', ' dags', and ' data'. Under ' dags', there are files '01_umbrella.py' and 'real_umbrella.py'. Under ' data', there are files 'clean_sales.csv', 'clean_weather.csv', 'joined_data.csv', 'ml_model.pkl', 'sales_data.csv', and 'weather_forecast.csv'. The code editor shows the contents of 'weather_forecast.csv', which is a CSV file with 5 lines of data. The first line is the header 'date,temperature'. The subsequent lines are '2025-03-21,4.1', '2025-03-22,0.6', and '2025-03-23,1.0'. The file is named 'weather_forecast.csv'.

```
data > weather_forecast.csv
1 date,temperature
2 2025-03-21,4.1
3 2025-03-22,0.6
4 2025-03-23,1.0
5
```

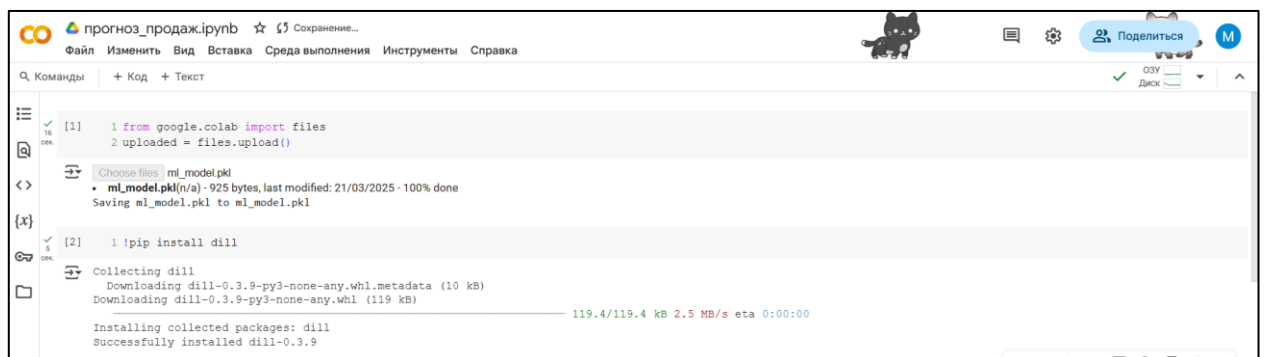
Они успешно сопоставились с данными о продажах зонтов:



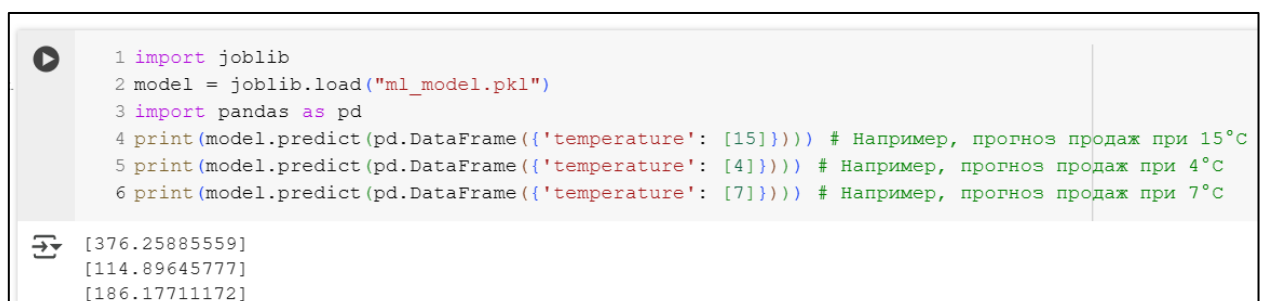
Выгрузка файла обученной на предоставленных данных модели:

```
dev@dev-vm:~/workshop-on-ETL-main/business_case_umbrella_25$ sudo docker cp business_case_umbrella_25-we
bserver-1:/opt/airflow/data/ml_model.pkl ./ml_model.pkl
Successfully copied 2.56kB to /home/dev/workshop-on-ETL-main/business_case_umbrella_25/ml_model.pkl
dev@dev-vm:~/workshop-on-ETL-main/business_case_umbrella_25$
```

Используя Google Colab и исходный файл кода, на основе обученной модели спрогнозируем количество продаж зонтов в зависимости от температуры:



Результат:



Как видно, модель предсказывает, что от повышения температуры количество продаж возрастает. В моем понимании, это может быть вполне

верно, т.к. при заполнении данных о продажах я отталкивалась от текущей погоды: то есть, сейчас только наступает весна, и при низкой температуре возможен снег, а как раз таки в зависимости от потепления вероятность, что пойдет дождь увеличивается.

Да и в принципе, зимой продажи зонтов резко ниже, чем в теплое время года, потому что дожди идут весной, летом и осенью)

АРХИТЕКТУРА РЕШЕНИЯ

