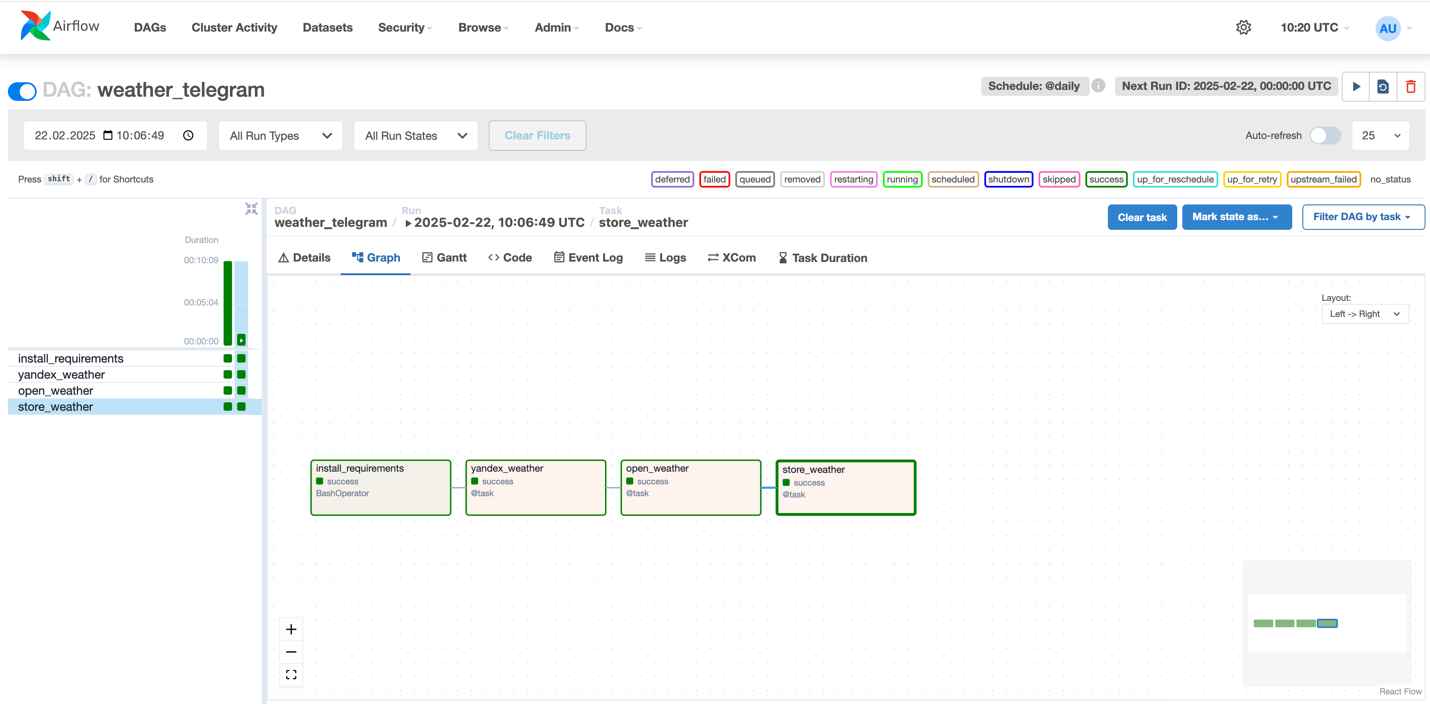
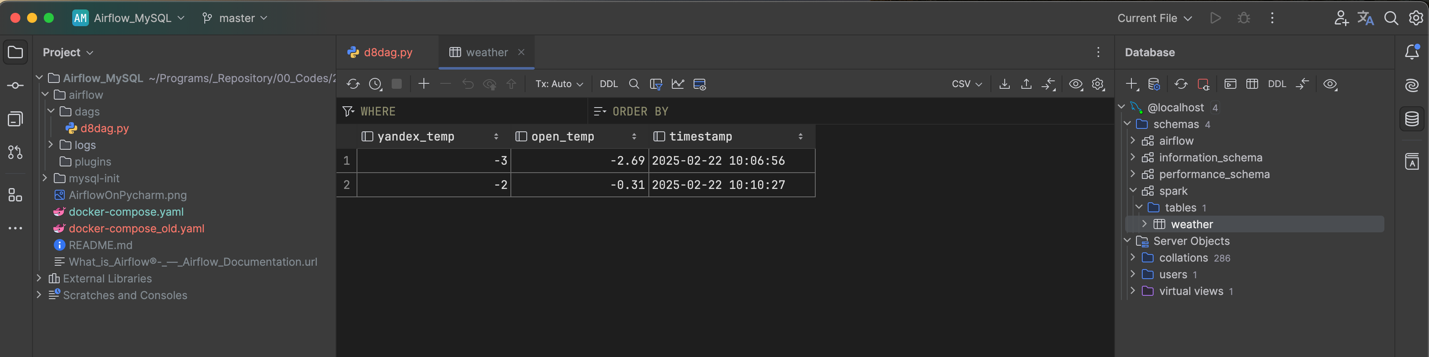
Этот DAG предназначен для взаимодействия с погодными API и отправки полученных данных в чат Telegram. Он извлекает данные о погоде из сервисов Yandex и OpenWeather, обрабатывает их, записывает результат в БД MySQL и может использоваться для отправки этих данных в Telegram (отправку в telegram запустить не удалось).

Вот подробное описание DAG:

# docker exec -it -u root airflow-webserver bash && apt-get update && apt-get upgrade  
# pip install apache-airflow-providers-telegram python-telegram-bot  
  
import datetime  
import os  
import requests  
import pendulum  
import pandas as pd  
from airflow.decorators import dag, task  
from airflow.hooks.base import BaseHook  
from airflow.models import Variable  
from sqlalchemy import create\_engine  
from airflow.operators.bash import BashOperator  
  
# Отключаем прокси, если они мешают  
os.environ["no\_proxy"] = "\*"  
  
YANDEX\_API\_KEY = '33f45b91-bcd4-46e4-adc2-33cfdbbdd88e'  
OPENWEATHER\_API\_KEY = '2cd78e55c423fc81cebc1487134a6300'  
TELEGRAM\_TOKEN = '7248934509:AAE4KtbM5wDjmrU-xFH8bMAwBwDZw-C\_cMA'  
CHAT\_ID = '783160683'  
  
# Дефолтные аргументы DAG  
default\_args = {  
 "owner": "airflow",  
 "retries": 3,  
}  
  
  
@dag(  
 dag\_id="weather\_telegram",  
 schedule="@daily",  
 start\_date=pendulum.datetime(2023, 1, 1, tz="UTC"),  
 catchup=False,  
 dagrun\_timeout=datetime.timedelta(minutes=60),  
 default\_args=default\_args,  
 tags=["weather", "telegram"],  
)  
def weather\_etl():  
 # Шаг 1: Установка необходимых зависимостей  
 install\_requirements = BashOperator(  
 task\_id="install\_requirements",  
 bash\_command=(  
 "pip install apache-airflow-providers-telegram python-telegram-bot "  
 "requests pandas sqlalchemy "  
 "apache-airflow-providers-mysql"  
 ),  
 )  
  
 @task(task\_id="yandex\_weather")  
 def get\_yandex\_weather(\*\*kwargs):  
 ti = kwargs["ti"]  
 url = "https://api.weather.yandex.ru/v2/informers/?lat=55.75396&lon=37.620393"  
 headers = {"X-Yandex-API-Key": YANDEX\_API\_KEY}  
  
 response = requests.get(url, headers=headers)  
 if response.status\_code != 200:  
 raise ValueError(f"Ошибка API Яндекса: {response.status\_code}, {response.text}")  
  
 temp = response.json().get("fact", {}).get("temp")  
 if temp is None:  
 raise ValueError("Не удалось получить температуру из ответа API Яндекса")  
  
 ti.xcom\_push(key="weather", value=temp)  
 return temp  
  
 @task(task\_id="open\_weather")  
 def get\_open\_weather(\*\*kwargs):  
 ti = kwargs["ti"]  
 url = f"https://api.openweathermap.org/data/2.5/weather?lat=55.75&lon=37.61&appid={OPENWEATHER\_API\_KEY}"  
  
 response = requests.get(url)  
 if response.status\_code != 200:  
 raise ValueError(f"Ошибка API OpenWeather: {response.status\_code}, {response.text}")  
  
 temp\_k = response.json().get("main", {}).get("temp")  
 if temp\_k is None:  
 raise ValueError("Не удалось получить температуру из ответа OpenWeather")  
  
 temp\_c = round(temp\_k - 273.15, 2)  
 ti.xcom\_push(key="open\_weather", value=temp\_c)  
 return temp\_c  
  
 @task(task\_id="store\_weather")  
 def store\_weather(\*\*kwargs):  
 ti = kwargs["ti"]  
  
 yandex\_temp = ti.xcom\_pull(task\_ids="yandex\_weather", key="weather")  
 open\_temp = ti.xcom\_pull(task\_ids="open\_weather", key="open\_weather")  
 timestamp = datetime.datetime.now()  
  
 # Настройки подключения к MySQL  
 MYSQL\_USER = "airflow"  
 MYSQL\_PASSWORD = "airflow"  
 MYSQL\_HOST = "mysql-db"  
 MYSQL\_DB = "spark"  
  
 # Подключение к MySQL через SQLAlchemy с таймаутом  
 con = create\_engine(f"mysql+pymysql://{MYSQL\_USER}:{MYSQL\_PASSWORD}@{MYSQL\_HOST}:3306/{MYSQL\_DB}",  
 connect\_args={'connect\_timeout': 10})  
  
 # db\_conn = BaseHook.get\_connection("mysql\_conn") # Используем Airflow Connection  
 # engine = create\_engine(  
 # f"mysql://{db\_conn.login}:{db\_conn.password}@{db\_conn.host}:{db\_conn.port}/{db\_conn.schema}")  
  
 data = [[yandex\_temp, open\_temp, timestamp]]  
 df = pd.DataFrame(data, columns=["yandex\_temp", "open\_temp", "timestamp"])  
  
 df.to\_sql("weather", con=con, if\_exists="append", index=False)  
  
 # from airflow.providers.telegram.operators.telegram import TelegramOperator  
 #  
 # send\_message\_telegram\_task = TelegramOperator(  
 # task\_id="send\_message\_telegram",  
 # telegram\_conn\_id="telegram\_conn",  
 # token=TELEGRAM\_TOKEN,  
 # chat\_id=CHAT\_ID,  
 # text="""  
 # Weather in Moscow:  
 # Yandex: {{ ti.xcom\_pull(task\_ids='yandex\_weather', key='weather') }} degrees  
 # OpenWeather: {{ ti.xcom\_pull(task\_ids='open\_weather', key='open\_weather') }} degrees  
 # """,  
 # )  
  
 # Определяем последовательность задач  
 install\_requirements >> get\_yandex\_weather() >> get\_open\_weather() >> store\_weather()  
 # >> send\_message\_telegram\_task()  
  
  
dag = weather\_etl()

Результат выполнения:





**Общее описание:**

Этот DAG включает следующие задачи:

1. **Получение погоды из Yandex**: Извлекает данные о температуре из API Yandex Weather, используя ключ API Yandex и координаты (широту и долготу) конкретного местоположения.

2. **Получение погоды из OpenWeather**: Извлекает данные о температуре из API OpenWeather, переводит температуру из Кельвина в Цельсия и сохраняет результат в XCom.

3. **Отправка погоды в Telegram**: Отправляет собранную информацию о погоде (из Yandex и OpenWeather) в указанный чат Telegram через бота.

**Ключевые компоненты:**

• **API ключ Yandex**: Используется для получения данных о погоде из Yandex.

• **API ключ OpenWeather**: Используется для получения данных о погоде из OpenWeather.

• **Токен Telegram и Chat ID**: Используются для отправки данных о погоде в чат Telegram.

**Основные шаги:**

1. **Задача погоды Yandex (get\_yandex\_wether)**:

• Эта функция вызывает API Yandex Weather с заранее заданными координатами (широта и долгота).

• Функция извлекает текущую температуру из данных Yandex и сохраняет её в XCom.

2. **Задача погоды OpenWeather (get\_open\_wether)**:

• Эта функция вызывает API OpenWeather с фиксированными координатами, затем обрабатывает температуру, преобразуя её из Кельвинов в Цельсии.

• Результат сохраняется в XCom для дальнейшего использования.

3. **Задача отчетности о погоде (get\_wether)**:

• Эта функция извлекает данные о погоде из XCom для обоих источников (Yandex и OpenWeather).

• Форматирует данные и может использоваться для вывода информации или отправки данных в другие системы, например, в Telegram.