Лабораторная работа 3. Программные средства консолидации данных из различных источников с использованием Python и Apache Airflow

Цель: научиться работать с Apache Airflow для автоматизации процессов ETL (Extract, Transform, Load). На практике освоить настройку и выполнение DAG в Airflow для извлечения данных из различных форматов (CSV, Excel, JSON), их обработки на Python, загрузки в базу данных SQLite и отправки уведомлений по электронной почте.

Оборудование и ПО:

- Ubuntu (с Docker)
- Apache Airflow
- SQLite
- Python
- Docker
- email сервис (например, Gmail или локальный SMTP-сервер)

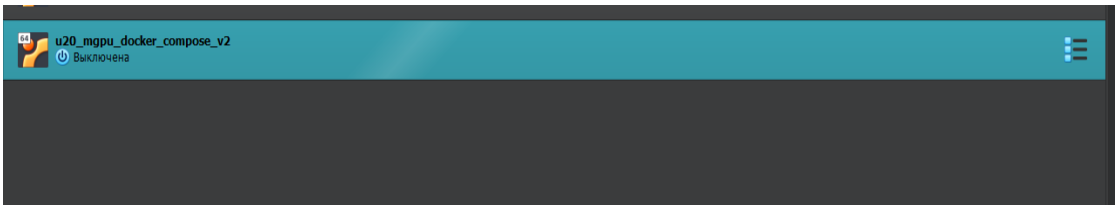
Исходные данные:

- Набор файлов CSV, Excel и JSON, содержащих данные для обработки.
- Конфигурация email для отправки уведомлений.

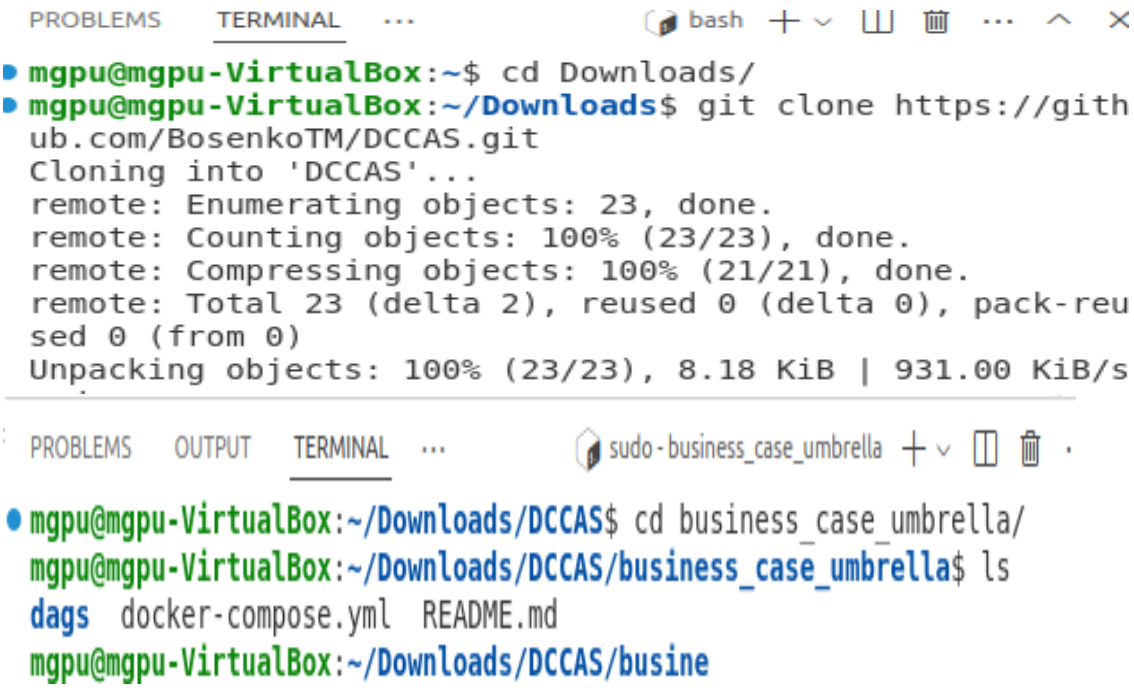
Выполнил: Петров Евгений С., БД-231м.

Ход работы

1. Развернуть ВМ ubuntu_mgpu.ova в VirtualBox.

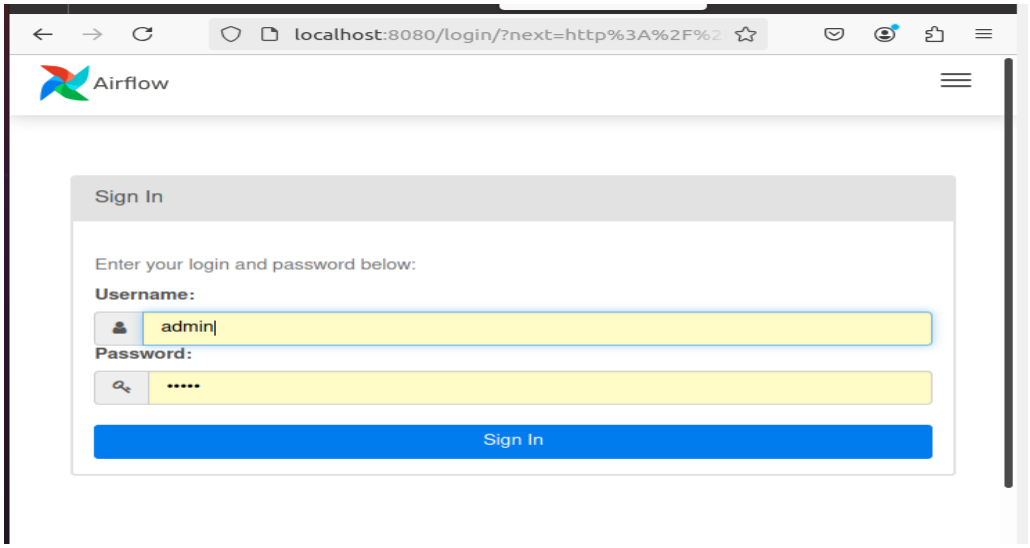


2. Клонировать на ПК задание Бизнес кейс Umbrella в домашний каталог ВМ.



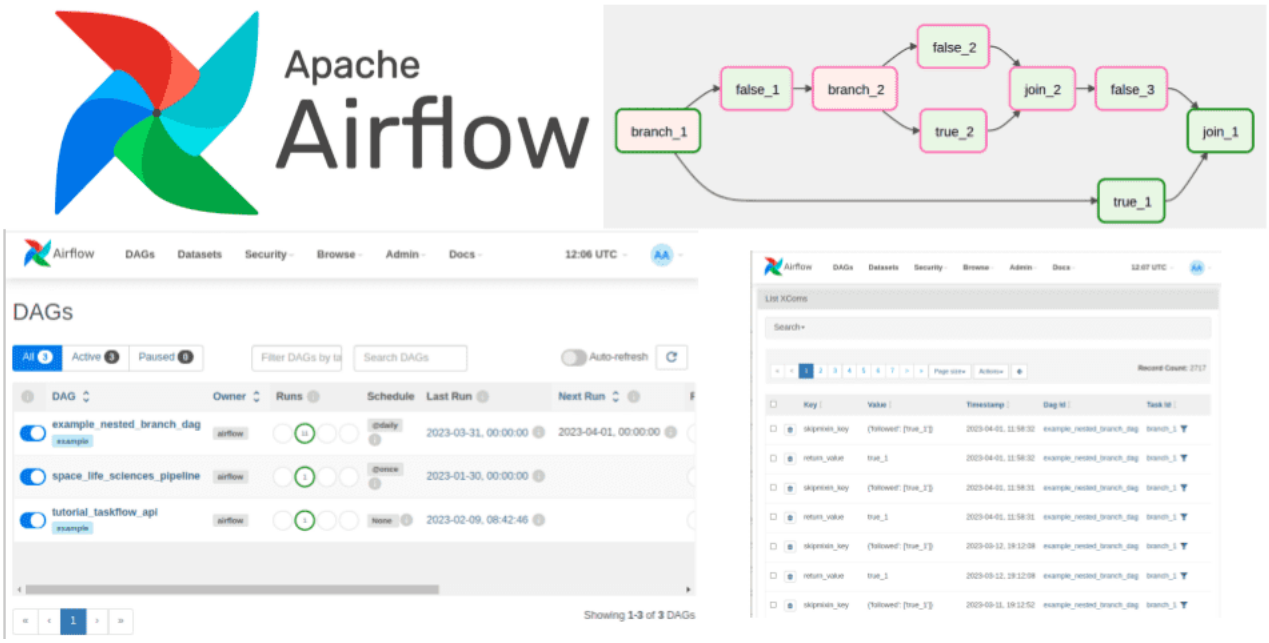
3. Запустить контейнер с кейсом.

```
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$ sudo docker c
ompose up -d
[+] Running 4/5
 ⠿ Network business_case_umbrella_default         Created                  3.4s
 ✔ Container business_case_umbrella-postgres-1    Started                  1.3s
 ✔ Container business_case_umbrella-init-1        Started                  2.5s
 ✔ Container business_case_umbrella-scheduler-1   Started                  3.0s
 ✔ Container business_case_umbrella-webserver-1   Started                  3.0s
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$
```



4. Изучить и описать основные элементы интерфейса Apache Airflow.

Основные элементы интерфейса **Apache Airflow**, представленные на скриншоте:



1. DAGS

DAGS (Directed Acyclic Graphs) — это основной элемент в Airflow, представляющий собой структуру, описывающую последовательность выполнения задач. В интерфейсе вы можете:

- Видеть список доступных DAGS.

- Проверять статус каждого DAG (активен, приостановлен и т.д.).

- Открывать визуализацию графа DAG для просмотра зависимостей между задачами.

- Запускать задачи вручную и просматривать их логи.

2. Datasets

Datasets представляют собой данные, которые потребляются или создаются в процессе выполнения задач. В интерфейсе Airflow можно видеть, какие датасеты используются в DAG, их состояния и метаданные. Это помогает в управлении зависимостями между задачами и в гарантии, что данные используются корректно.

3. Security

Элемент безопасности позволяет управлять доступом к интерфейсу Airflow. Здесь можно:

- Настраивать роли и разрешения для пользователей.

- Ограничивать доступ к определенным DAGS или функционалу на основе ролей.

4. Browse

Этот элемент интерфейса предоставляет доступ к различным данным и информации:

- Вы можете просматривать выполненные экземпляры DAG и их задачи (task instances).

- Доступ к информации о привязках к метаданным (например, логам и результатам выполнения).

- Возможность просматривать состояния выполнения задач.

5. Admin

Раздел администрирования предоставляет инструменты для управления настройками Airflow:

- Управление пользователями и ролями.

- Настройки подключения к базам данных и другим системам.

- Настройка параметров выполнения и конфигурации.

6. Docs

Этот раздел содержит документацию по Airflow, включая:

- Описание доступных операторов и как их использовать.

- Лучшие практики для создания и управления DAG.

- Примеры и справочные материалы.

7. Status: All, Active, Paused

В интерфейсе Airflow вы можете фильтровать DAGs по их статусу:

- **All**: Показывает все DAGs.

- **Active**: Показывает только активные (включенные) DAGs.

- **Paused**: Показывает приостановленные DAGs.

8. Autorefresh

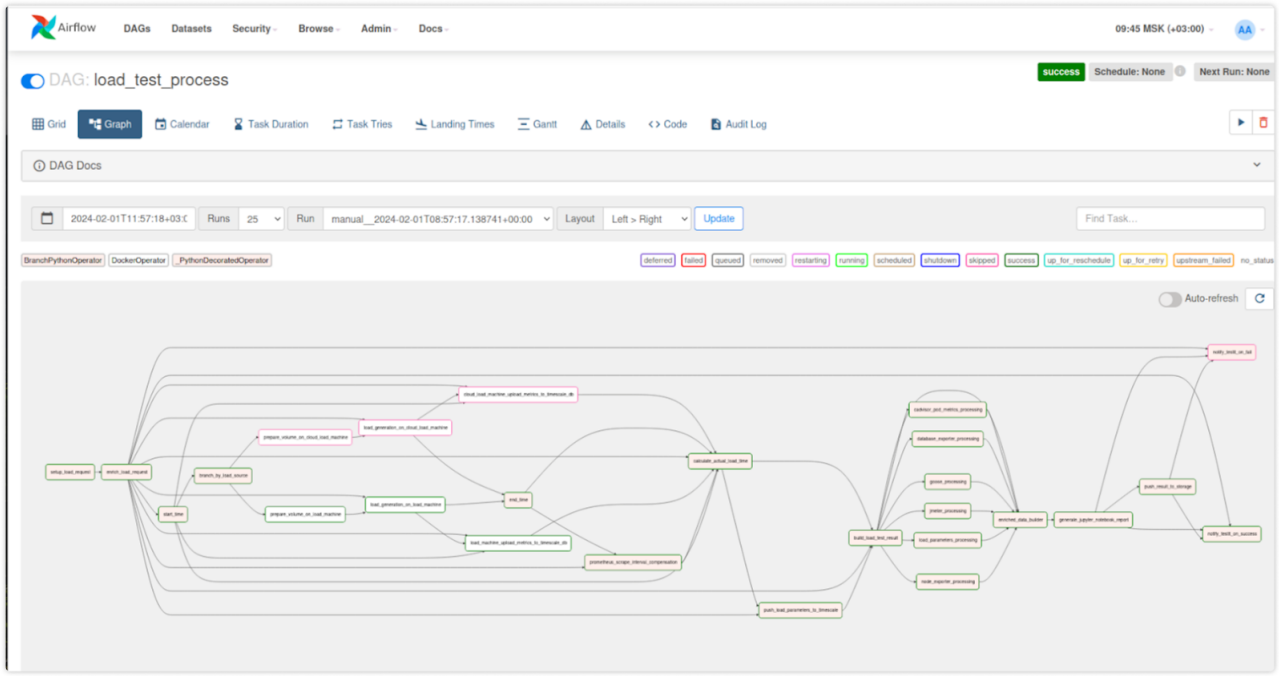
Функция автоматического обновления значения в реальном времени — это удобный инструмент, который позволяет пользователям следить за состоянием задачи и графов без необходимости вручную обновлять страницу.

9. Filter

Элемент фильтрации позволяет пользователям быстро находить конкретные DAGs или задачи по имени, статусу или другим критериям. Это особенно полезно в больших проектах с множеством DAGs.

Эти элементы интерфейса в Apache Airflow помогают пользователям эффективно управлять и мониторить их рабочие процессы, обеспечивая простоту и удобство использования.

На скриншоте представлен интерфейс графического представления **DAG** в Apache Airflow:



Apache Airflow предоставляет мощный интерфейс для графического представления Directed Acyclic Graphs (DAG), что позволяет пользователям эффективно управлять и мониторить задачи. Давайте рассмотрим основные элементы интерфейса:

1. Grid (Сетка):
Это одно из представлений DAG, где отображаются все задачи графа. Каждая ячейка в сетке соответствует конкретной задаче, и пользователи могут видеть состояние задач, такие как «завершено», «в процессе» или «неудачно».

2. Graph (Граф):

Это визуальное представление зависимостей между задачами. Задачи отображаются в виде узлов, а зависимости между ними - в виде рёбер. Это позволяет пользователям легко понять, как задачи связаны друг с другом и в каком порядке они должны выполняться.

3. Calendar (Календарь):

Этот элемент интерфейса позволяет пользователю просматривать, когда были выполнены конкретные задачи и каков их граф выполнения. Календарь может включать отметки для успешных и неудачных запусков задач.

4. Task Duration (Длительность задач):

Этот параметр показывает, сколько времени понадобилось для выполнения каждой задачи. Это может помочь в определении узких мест в процессе и оптимизации производительности.

5. Task Tries (Попытки задач):

В этом разделе отображается количество попыток выполнения каждой задачи. Airflow автоматически повторяет выполнение задач в случае неудачи, и этот элемент позволяет пользователям увидеть, сколько раз каждая задача пыталась выполниться.

6. Landing Time (Время завершения):

Этот элемент показывает, когда была завершена ссылка (проверка) задач. Это может быть важно для анализа временных задержек и повышения эффективности обработки данных.

7. Gantt (График Ганта):

Это элемент, который показывает выполнение задач по временной шкале. Пользователи могут видеть, какие задачи выполняются параллельно, а также их взаимные зависимости, что позволяет оценить общую эффективность DAG.

8. Details (Детали):

Этот раздел предоставляет пользователям детальную информацию о конкретной задаче, включая ее состояние, параметры, лог выполнения и любые другие метаданные, которые могут быть важны для диагностики или мониторинга.

9. Code (Код):

В этом разделе можно просмотреть исходный код DAG. Это полезно для разработчиков, чтобы быстро просмотреть настройки и логику выполнения.

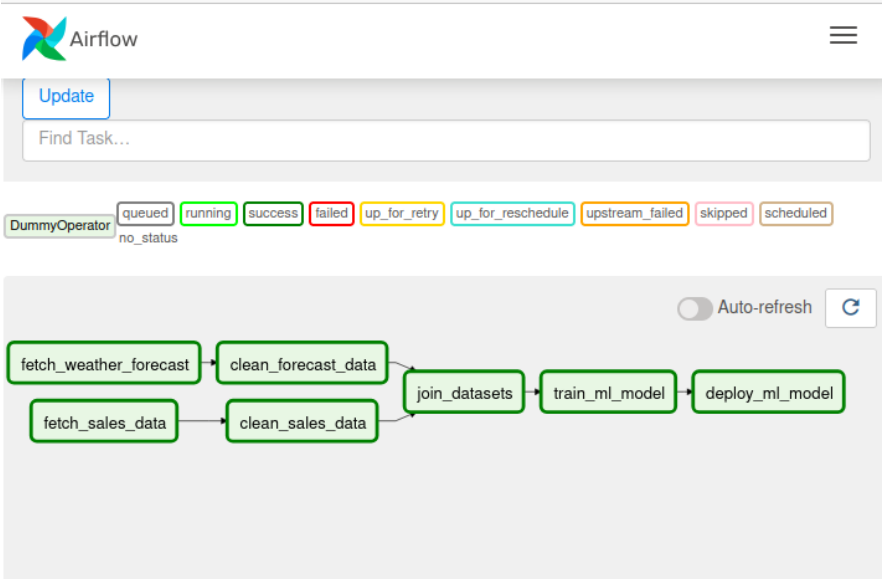
10. Audit Log (Журнал аудита):

Этот элемент сохраняет записи обо всех изменениях и событиях, происходящих в процессе выполнения DAG. Это важно для мониторинга, анализа и обеспечения безопасности.

11. Auto-refresh (Автообновление):

Эта функция автоматически обновляет интерфейс через определенные промежутки времени, что позволяет пользователям оставаться в курсе текущего состояния задач и DAG без необходимости ручного обновления страницы

На скриншоте представлен интерфейс графического представления **Graph View** в Apache Airflow:



Графический интерфейс представляет собой один из основных способов визуализации рабочих процессов (DAG - Directed Acyclic Graph) в Apache Airflow. Он предоставляет пользователям интуитивно понятный способ отслеживания состояния задач и управления ими. Вот основные элементы интерфейса Graph View:

1. Узлы (Nodes)

Каждый узел в графе представляет собой конкретную задачу (task) в рамках рабочего процесса. Узлы могут отображать различную информацию:

- **Имя задачи**: Отображается в узле.

- **Состояние задачи**: Цвет узла указывает на текущее состояние задачи.

2. Цвета узлов

Цвета узлов помогают быстро оценивать состояние задач:

- **Синий**: Задача ожидает выполнения.

- **Зеленый**: Задача успешно выполнена и завершена.

- **Красный**: Задача завершилась с ошибкой (неудача).

- **Серый**: Задача не была запущена, например, если она отключена.

- **Оранжевый**: Задача находится в состоянии "выполняется" или "обрабатывается".

- **Желтый**: Задача в состоянии "пауза" или "ожидание".

3. Связи (Dependencies)

Стрелки между узлами показывают зависимости между задачами. Это помогает понять, в каком порядке задачи должны быть выполнены. Зависимости могут быть:

- **Последовательные**: Одна задача должна завершиться, чтобы началась следующая.

- **Параллельные**: Несколько задач могут выполняться одновременно.

4. Параметры фильтрации

В графическом интерфейсе Airflow можно переключаться между различными видами представления (например, "Graph View", "Tree View" и "Gantt View"). Также есть возможность фильтровать задачи по статусу, дате выполнения и другим параметрам.

5. Всплывающие подсказки

Наведение курсора на узел задачи для вызова всплывающей подсказки, которая может содержать дополнительную информацию, такую как время выполнения, продолжительность, и другие метрики.

6. Панель инструментов

В верхней части интерфейса находится панель инструментов, позволяющая:

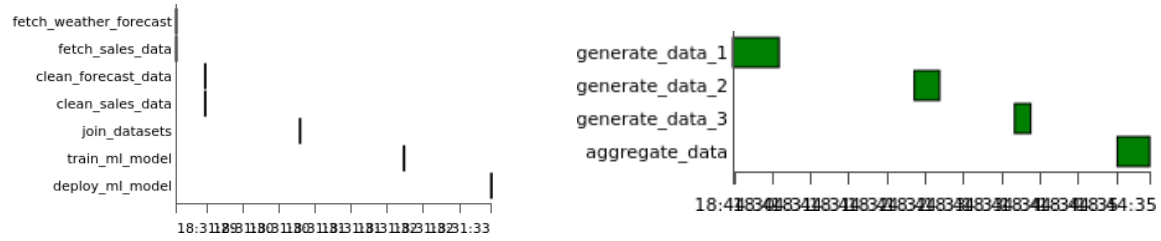
- Перезапустить DAG.

- Задавать параметры выполнения (например, выбрать дату).

- Получить доступ к журналам выполнения задач.

7. Состояние DAG

В верхней части страницы тоже может отображаться общее состояние DAG, включая общее количество задач, количество успешных и завершившихся с ошибками задач. На скриншоте представлен интерфейс графического представления **Gantt** в Apache Airflow, которая позволяет отслеживать временные интервалы выполнения каждой задачи:



Основные особенности этой вкладки:

1. **Диаграмма Ганта**:

- Оси времени: горизонтальная ось представляет собой временную шкалу, которая показывает время выполнения задач. Она позволяет визуально оценить, как долго выполнялась каждая задача и в какие моменты времени они начинались изаканчивались.

- Задачи: Каждая задача представлена горизонтальной полосой. Начало полосы соответствует времени старта задачи, а её длина указывает напродолжительность выполнения.

2. **Цветовые индикаторы**: подобно другим видам в Airflow, цвет полос задач на диаграмме Ганта показывает их статус.

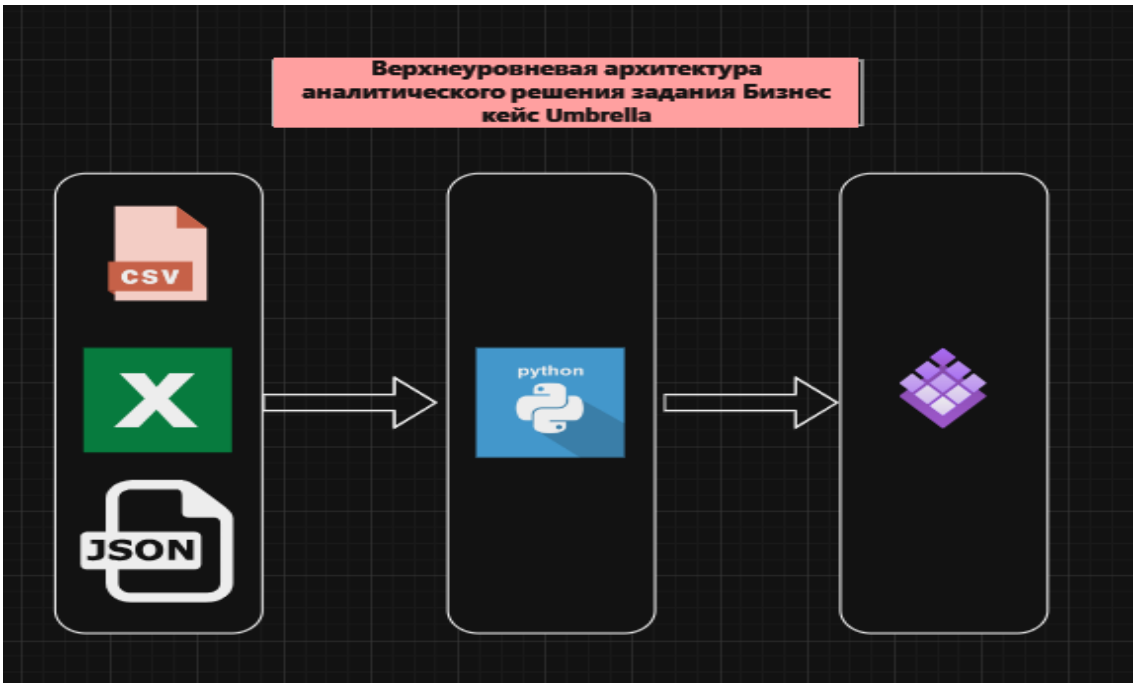
3. **Интерактивность**: нажав на любую полосу на диаграмме Ганта, можно получить более детальную информацию о задаче.

Анализ зависимости времени выполнения: диаграмма Ганта помогает анализировать, как задачи

DAG выполняются во времени относительно друг друга. Можно увидеть, какие задачи выполнялись параллельно, какие задачи задержали выполнение других задач, и какоптимизировать граф для более

5. Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес кейс Umbrella в draw.io. Необходимо использовать:

- Source Layer - слой источников данных.
- Storage Layer - слой хранения данных.
- Business Layer - слой для доступа к данным бизнес-пользователей.



6. Выполнить индивидуальное задание.

Задание: создать DAG, который извлекает данные из CSV, Excel и JSON файлов, выполняет базовые операции по очистке (удаление дубликатов, обработка пропущенных значений) и сохраняет результат в Google Drive в формате Excel.

7. Выполнить индивидуальное задание.

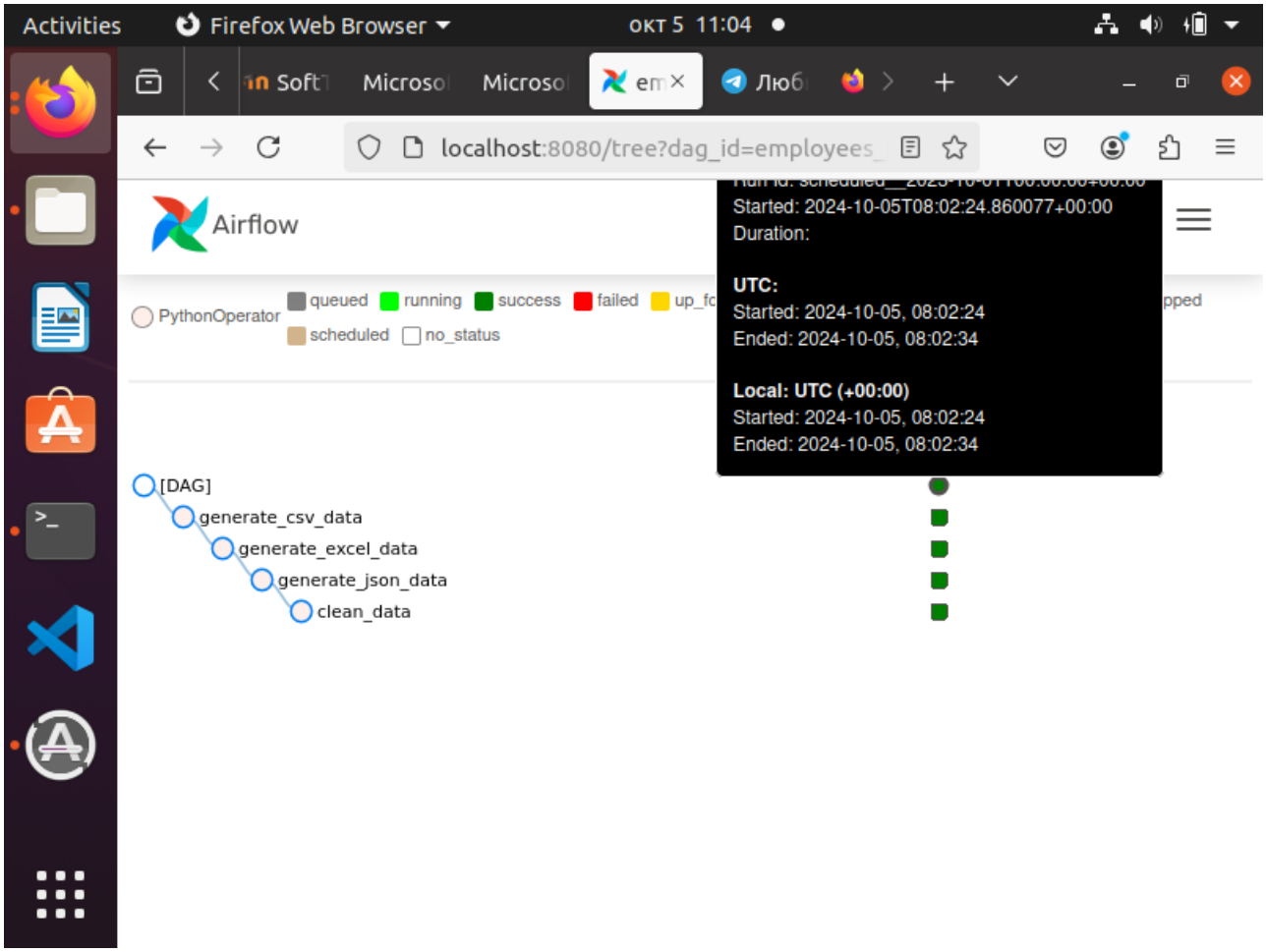
Задание: создать DAG, который извлекает данные из CSV, Excel и JSON файлов, выполняет базовые операции по очистке (удаление дубликатов, обработка пропущенных значений) и сохраняет результат в Google Drive в формате Excel.

1. **Создание файла DAG partners_data_processing_dag**:
   1. Файл CSV: данные о сотрудниках и их квалификациях (имя, квалификация, стаж).
   2. Файл Excel: данные о проектах (проект, сотрудник, часы работы).
   3. Файл JSON: данные о стоимости часа работы в зависимости от квалификации сотрудника.

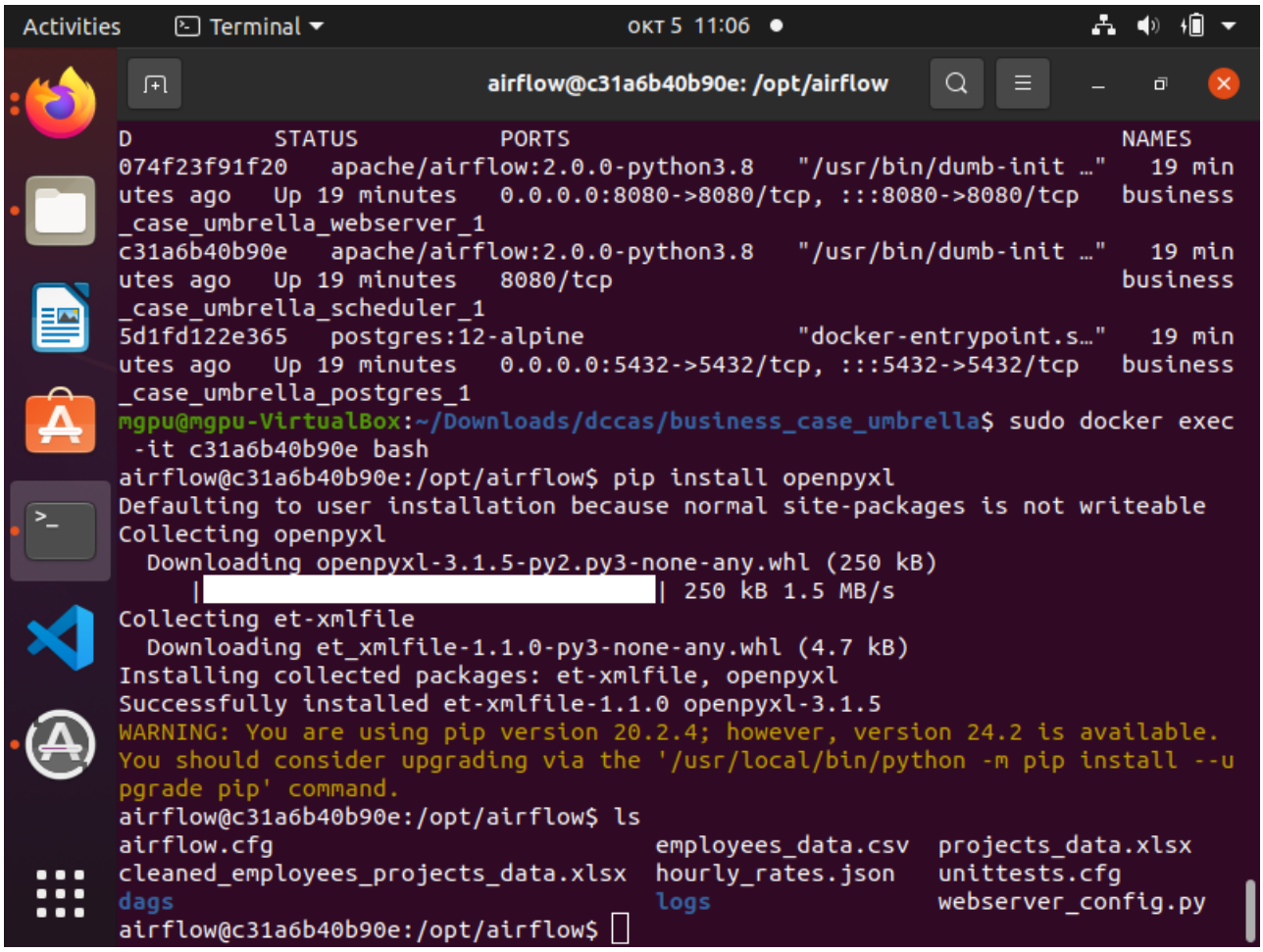
```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator  # type: ignore
from datetime import datetime
import pandas as pd  # type: ignore
import random
import json

# Список имен для генерации данных о сотрудниках
employee_names = ['Алексей', 'Мария', 'Дмитрий', 'Анна', 'Сергей', 'Елена', 'Иван', 'Ольга',
    'Николай', 'Татьяна']

# Функция для генерации CSV файла с данными о сотрудниках
def generate_csv_data():
    data_csv = {
        'Имя': [random.choice(employee_names) for _ in range(100)],
        'Квалификация': [random.choice(['Junior', 'Middle', 'Senior']) for _ in range(100)],
        'Стаж (лет)': [random.randint(0, 15) for _ in range(100)],
    }
    df_csv = pd.DataFrame(data_csv)
    df_csv.to_csv('employees_data.csv', index=False, encoding='utf-8')

# Функция для генерации Excel файла с данными о проектах
def generate_excel_data():
    data_excel = {
        'Проект': [f'Проект {i}' for i in range(1, 101)],
        'Сотрудник': [random.choice(employee_names) for _ in range(100)],
        'Часы работы': [random.randint(1, 40) for _ in range(100)],
    }
    df_excel = pd.DataFrame(data_excel)
    df_excel.to_excel('projects_data.xlsx', index=False)

# Функция для генерации JSON файла с данными о стоимости часа работы в зависимости от
    квалификации
def generate_json_data():
    data_json = {
        'Junior': round(random.uniform(200.0, 500.0), 2),
        'Middle': round(random.uniform(500.0, 1000.0), 2),
        'Senior': round(random.uniform(1000.0, 2000.0), 2)
    }
    with open('hourly_rates.json', 'w', encoding='utf-8') as f:
        json.dump(data_json, f, ensure_ascii=False)

# Функция для обработки данных (очистка, удаление дубликатов и заполнение пропусков)
def clean_data():
    # Загрузка данных
    df_csv = pd.read_csv('employees_data.csv', encoding='utf-8')
    df_excel = pd.read_excel('projects_data.xlsx')
    with open('hourly_rates.json', 'r', encoding='utf-8') as f:
        data_json = json.load(f)

    # Добавление столбца с ставкой за час в зависимости от квалификации
```

```
    df_csv['Ставка за час'] = df_csv['Квалификация'].map(data_json)

    # Объединение данных о проектах с данными о сотрудниках
    merged_df = pd.merge(df_excel, df_csv, left_on='Сотрудник', right_on='Имя', how='outer')

    # Очистка данных: удаление дубликатов и обработка пропусков
    merged_df.drop_duplicates(subset='Сотрудник', keep='first', inplace=True)
    merged_df.fillna({'Часы работы': 0, 'Ставка за час': 0}, inplace=True)

    # Сохранение результата в Excel файл
    merged_df.to_excel('cleaned_employees_projects_data.xlsx', index=False)

# Определение DAG
dag = DAG('employees_projects_data_processing_dag', description='Генерация и очистка
    данных о сотрудниках и проектах',
        schedule_interval='@once', start_date=datetime(2023, 10, 1), catchup=False)

# Определение задач
task_generate_csv_data = PythonOperator(task_id='generate_csv_data',
    python_callable=generate_csv_data, dag=dag)
task_generate_excel_data = PythonOperator(task_id='generate_excel_data',
    python_callable=generate_excel_data, dag=dag)
task_generate_json_data = PythonOperator(task_id='generate_json_data',
    python_callable=generate_json_data, dag=dag)
task_clean_data = PythonOperator(task_id='clean_data', python_callable=clean_data, dag=dag)

# Установка зависимостей между задачами
    task_generate_csv_data >> task_generate_excel_data >> task_generate_json_data >>
    task_clean_data
```

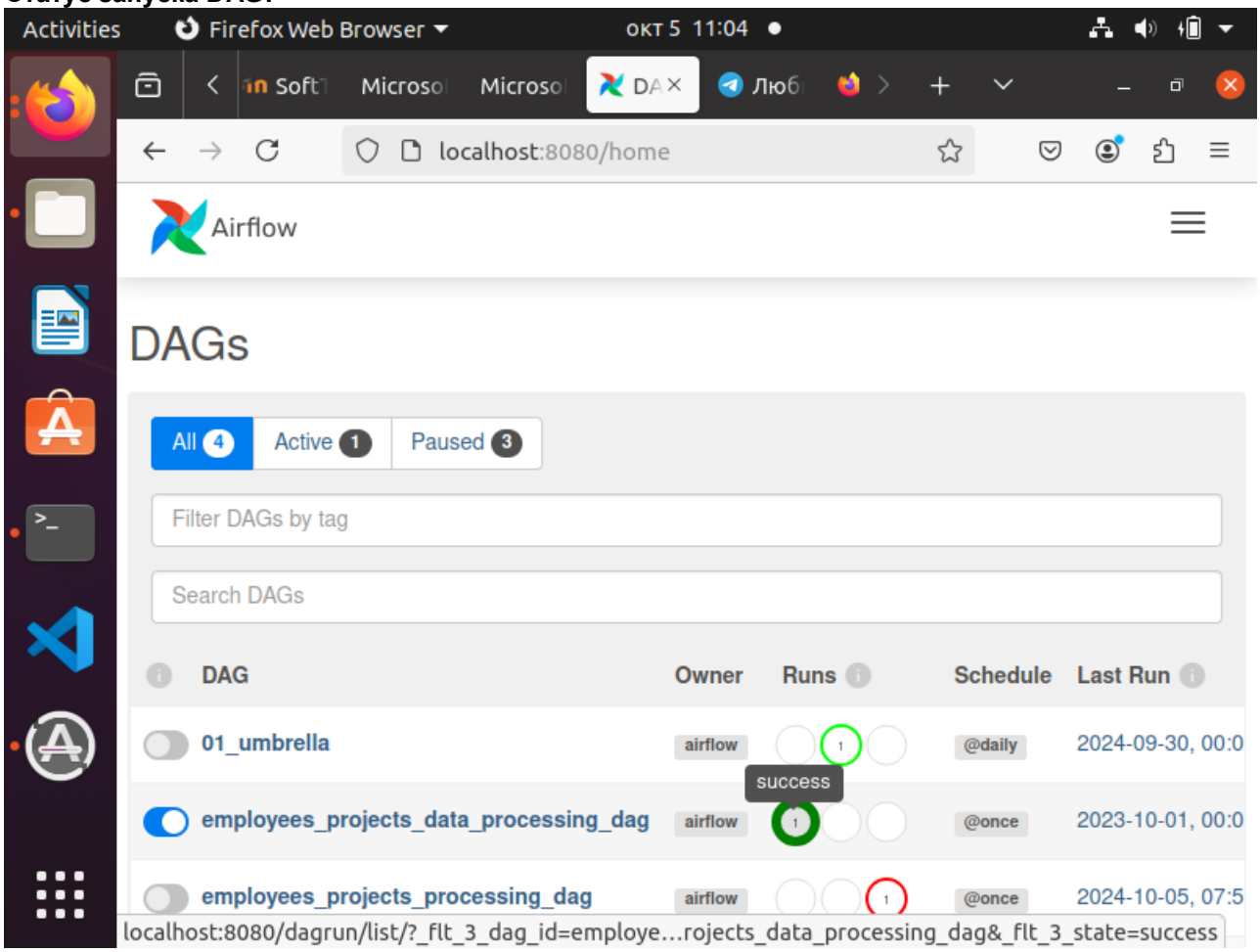
2. **Запуск DAG:**


Activities
Firefox Web Browser
OKT 5 11:04
localhost:8080/tree?dag_id=employees_
Airflow
Started: 2024-10-05T08:02:24.860077+00:00
Duration:
UTC:
Started: 2024-10-05, 08:02:24
Ended: 2024-10-05, 08:02:34
Local: UTC (+00:00)
Started: 2024-10-05, 08:02:24
Ended: 2024-10-05, 08:02:34
PythonOperator
queued
running
success
failed
scheduled
no_status
[DAG]
generate_csv_data
generate_excel_data
generate_json_data
clean_data

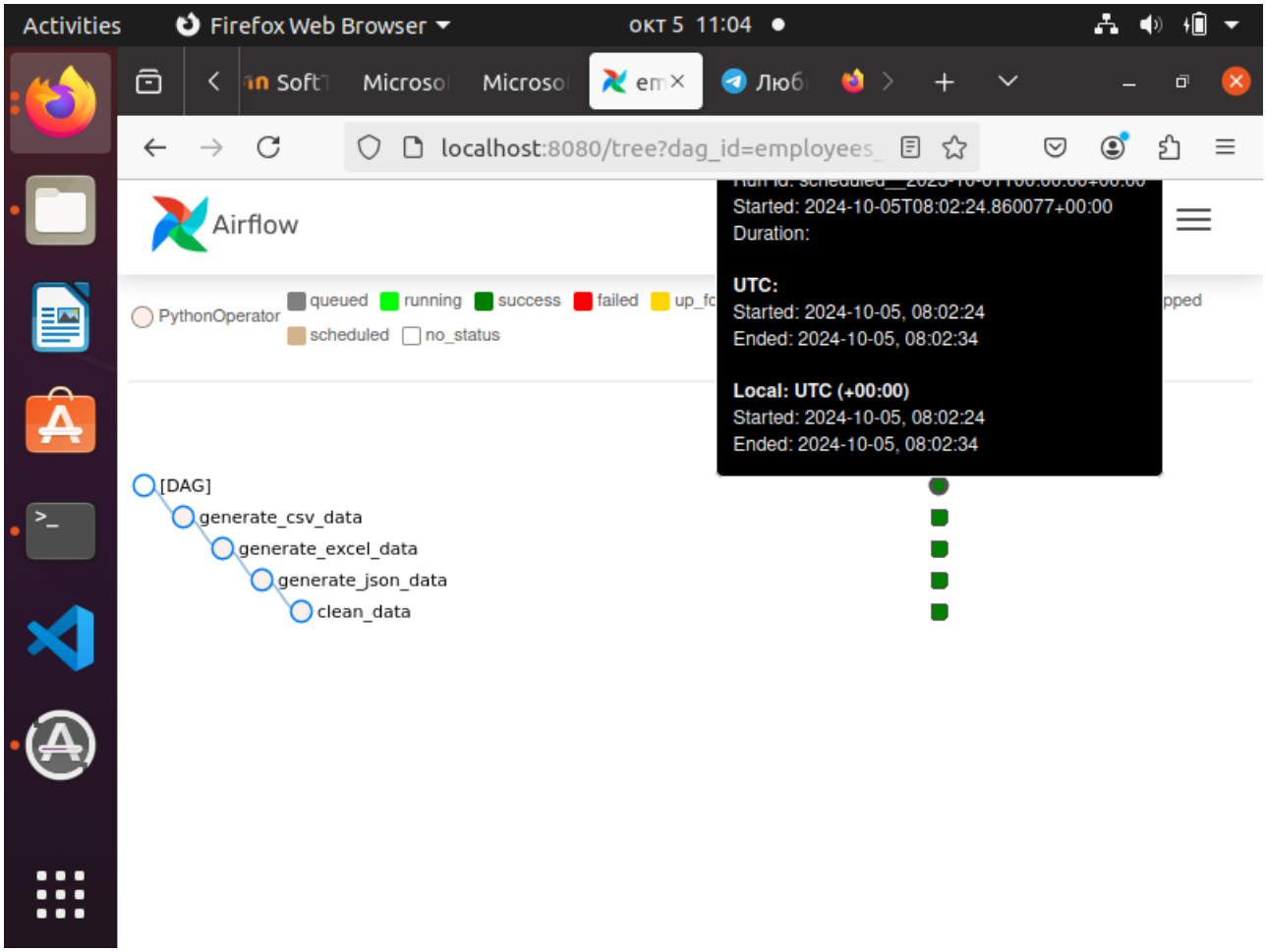
3. **Предварительно перед этим был установлен модуль openpyxl:**



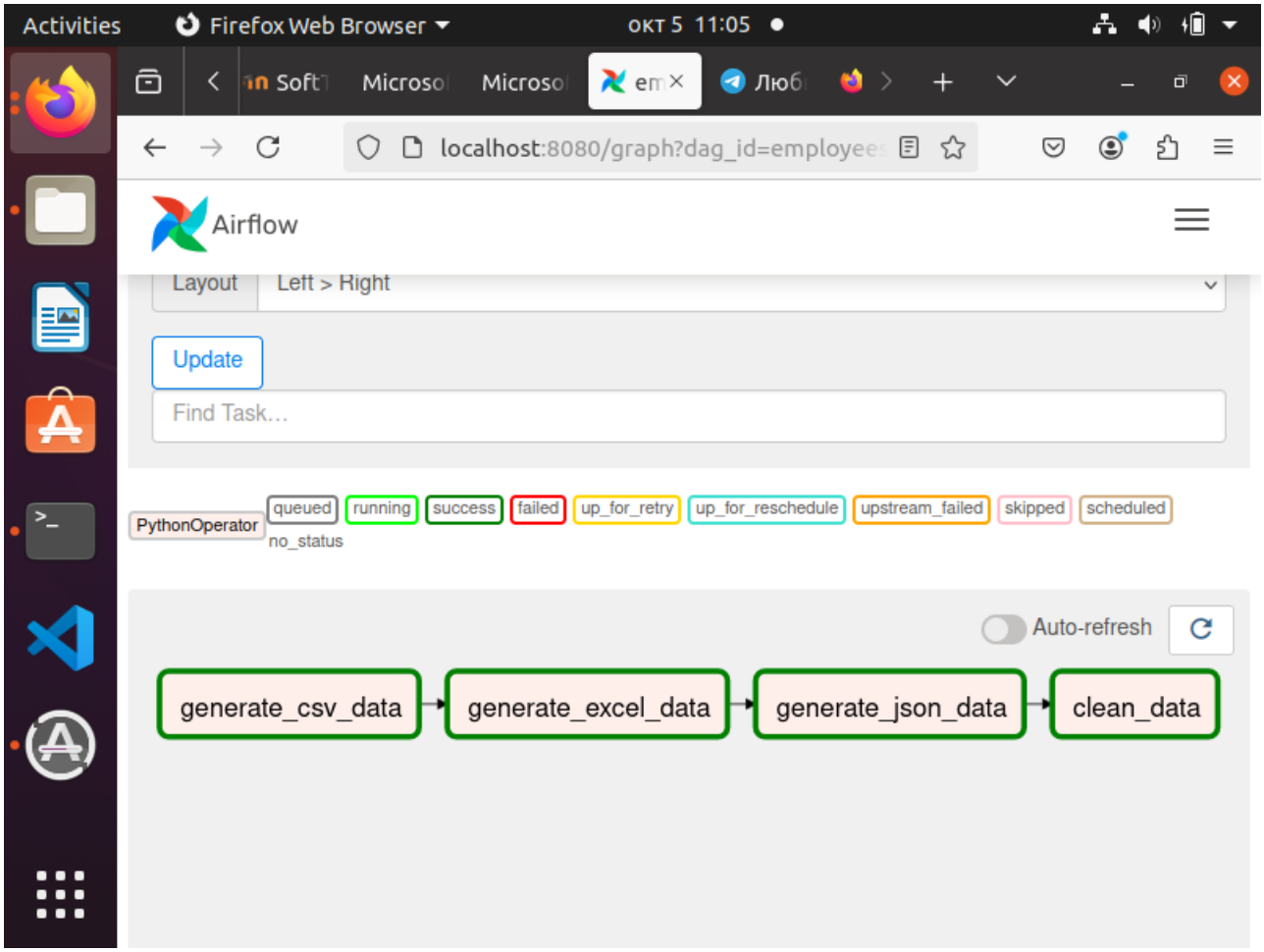
4. **Статус запуска DAG:**



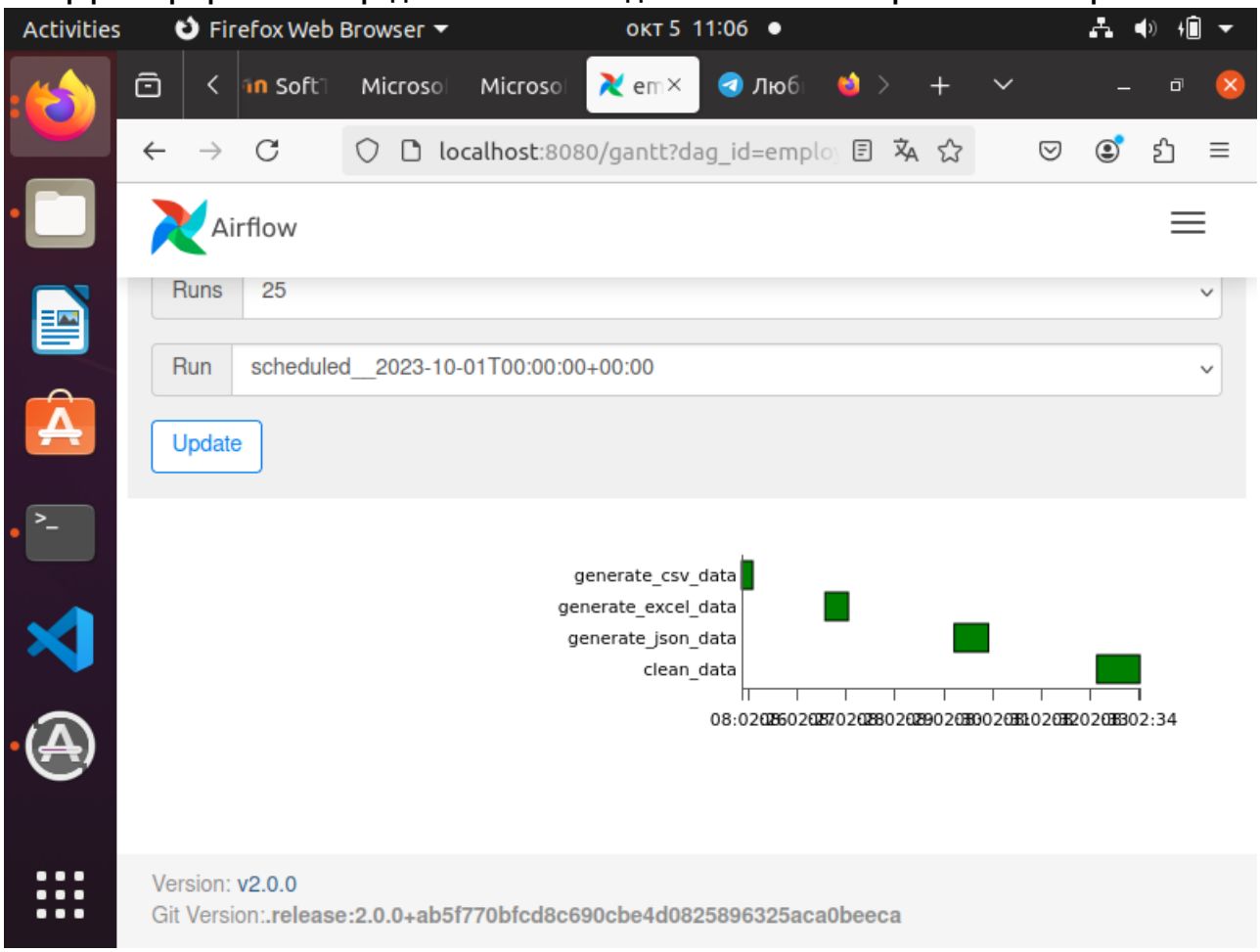
5. **Интерфейс графического представления DAG :**



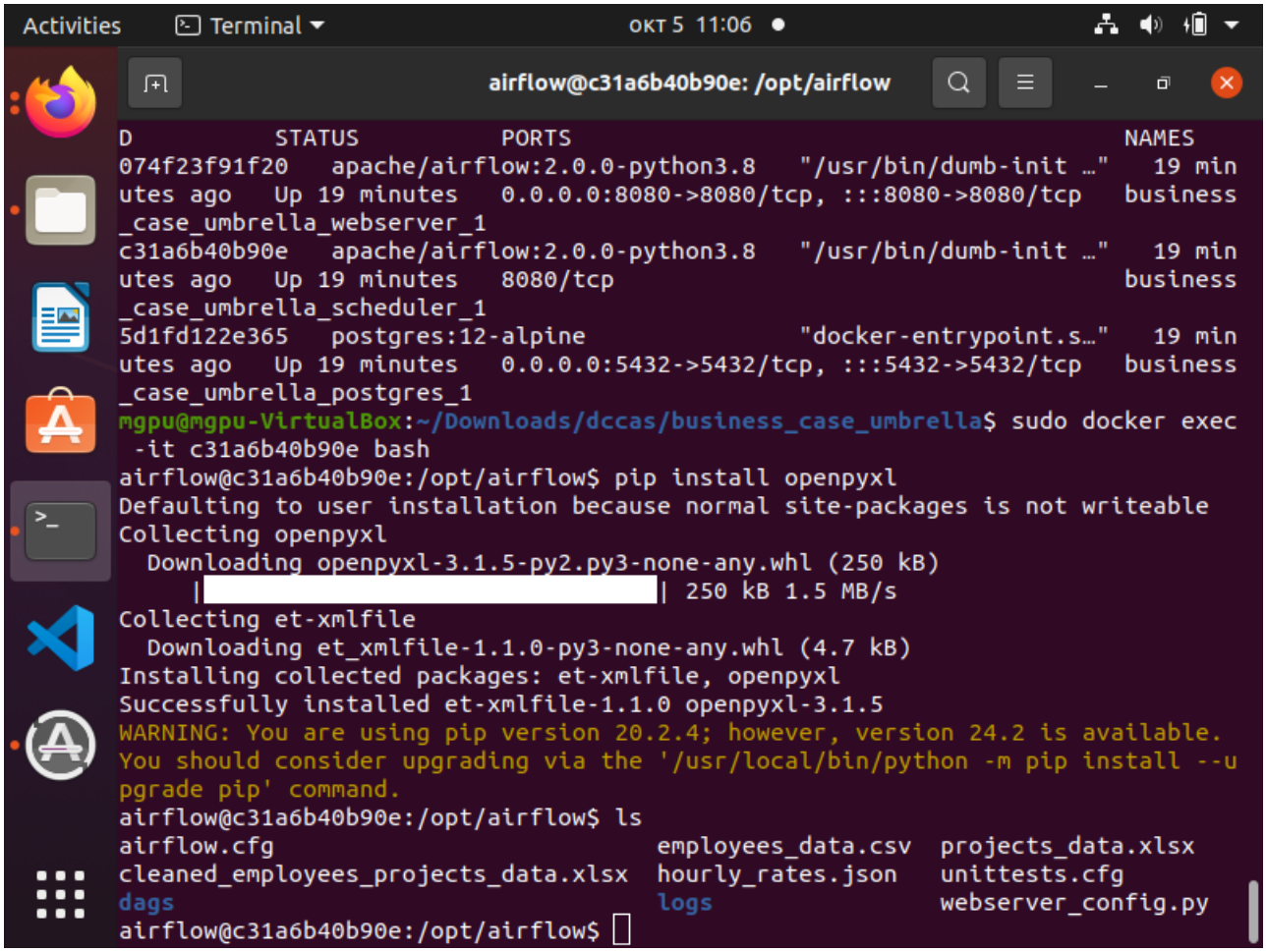
6. **Интерфейс графического представления GraphView:**


Activities
Firefox Web Browser
окт 5 11:05
localhost:8080/graph?dag_id=employee
Airflow
Layout
Left > Right
Update
Find Task...
PythonOperator
queued
running
success
failed
up_for_retry
up_for_reschedule
upstream_failed
skipped
scheduled
no_status
Auto-refresh
generate_csv_data
generate_excel_data
generate_json_data
clean_data

7. **Внтерфейс графического представления Gantt для отслеживания временных интервалов**



8. **Файлы в текущем каталоге Airflow:**



9. **Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес кейс**