

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

«Проектный практикум по разработке ETL-решений»

Практическая работа №6

Тема:

«Оркестровка конвейера данных»

Выполнила: Овсепян Милена, АДЭУ-201

Преподаватель: Босенко Т.М.

Москва

2023

СОДЕРЖАНИЕ

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ.....	3
ГЛАВА 2. РЕШЕНИЕ ЗАДАЧИ.....	5
2.1 Начало работы	5
2.2 Исходный код всех DAGs.....	9
2.3 Граф DAG в Apache Airflow.....	9
2.4 Верхнеуровневая архитектура задания Бизнес-кейса «StockSense»	10
2.5 Архитектура DAG Бизнес-кейса «StockSense».....	10
2.6 Диаграмма Ганта DAG в Apache Airflow.....	11
2.7 SQL-запросы, позволяющие проверить наличие выгруженных агрегированных данных бизнес-задачи	12
ЗАКЛЮЧЕНИЕ.....	14

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ

1. Развернуть ВМ ubuntu_mgpu.ova в VirtualBox.
2. Клонировать на ПК задание Бизнес-кейс «StockSense» в домашний каталог ВМ.

`git clone https://github.com/BosenkoTM/workshop-on-ETL.git`

3. Запустить контейнер с кейсом, изучить основные элементы DAG в Apache Airflow.
- Создать DAG согласно алгоритму, который предоставит преподаватель.
 - Изучить логи, выполненного DAG. Скачать логи из контейнера в основную ОС.

Пример docker-compose.yml создает базу данных в Postgres:

- Host: localhost
- Port: 5433
- Username: airflow
- Password: airflow
- Database: airflow

Эта база данных инициализируется таблицей pageview_counts.

4. Агрегированные данные бизнес-процесса, полученные в результате работы DAG в Apache Airflow, выгрузить в Postgr SQL.
5. Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес-кейса «StockSense» в draw.io. Необходимо использовать:

Source Layer - слой источников данных.

Storage Layer - слой хранения данных.

Business Layer - слой для доступа к данным пользователей.

6. Спроектировать архитектуру DAG Бизнес-кейса «StockSense» в draw.io. Необходимо использовать:

Source Layer - слой источников данных.

Storage Layer - слой хранения данных.

Business Layer - слой для доступа к данным пользователей.

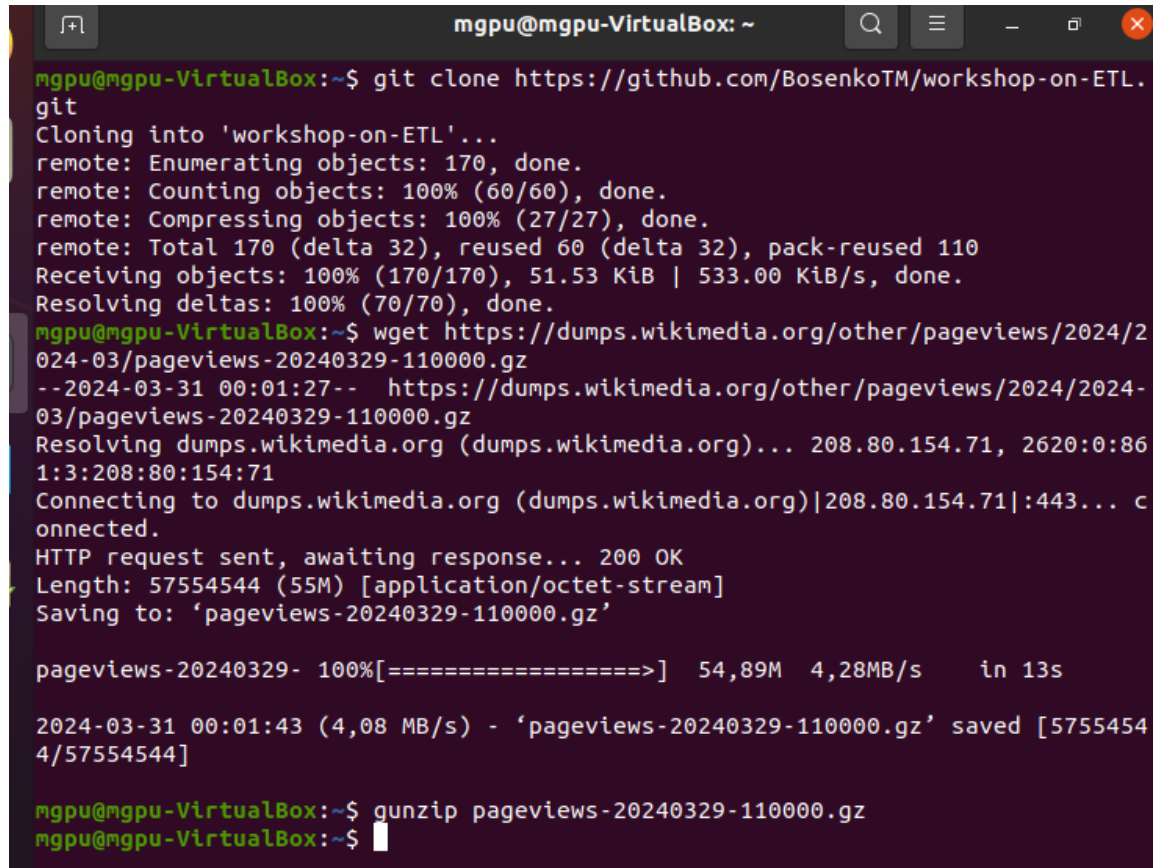
7. Построить диаграмму Ганта работы DAG в Apache Airflow.
8. Результаты исследований представить в виде файла ФИО-06.pdf, в котором отражены следующие результаты:
- постановка задачи;
 - исходный код всех DAGs, которые требовались для решения задачи, а также представить граф DAG в Apache Airflow;
 - верхнеуровневая архитектура задания Бизнес-кейса «StockSense», выполненная в draw.io;
 - архитектура DAG Бизнес-кейса «StockSense» , выполненная в draw.io;
 - диаграмма Ганта DAG в Apache Airflow;
 - ERD-схема базы данных PostgreSQL;
 - SQL-запросы, позволяющие проверить наличие выгруженных агрегированных данных бизнес-задачи.

После проверки преподавателем работоспособности DAG, выгрузить отчет на портал moodle.

ГЛАВА 2. РЕШЕНИЕ ЗАДАЧИ

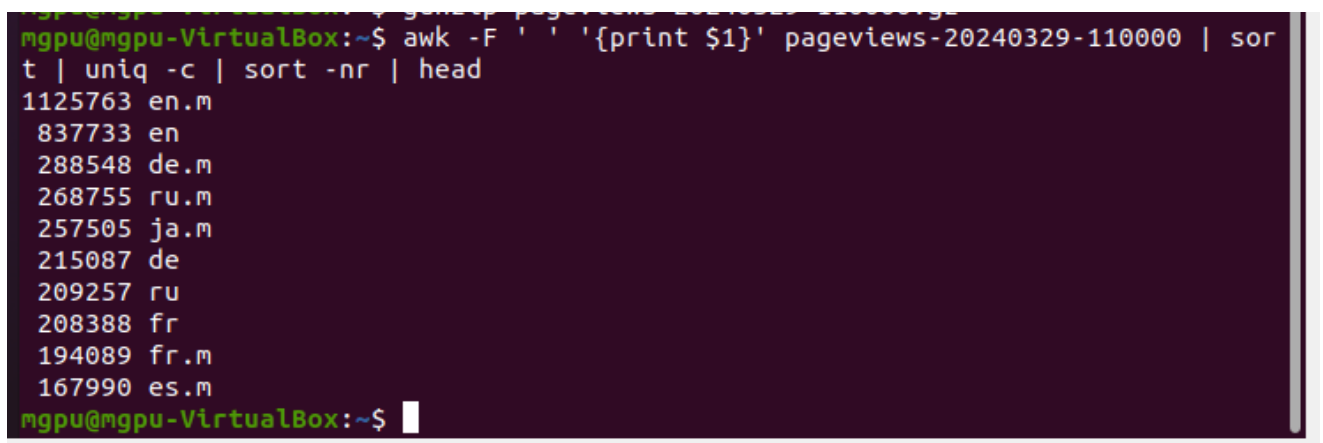
2.1 Начало работы

Для работы сначала нужно клонировать репозиторий. Далее скачиваем и проверяем данные о просмотре страницы «Викимедия». (рисунок 1-2)



```
mgpu@mgpu-VirtualBox: ~  
mgpu@mgpu-VirtualBox:~$ git clone https://github.com/BosenkoTM/workshop-on-ETL.  
git  
Cloning into 'workshop-on-ETL'...  
remote: Enumerating objects: 170, done.  
remote: Counting objects: 100% (60/60), done.  
remote: Compressing objects: 100% (27/27), done.  
remote: Total 170 (delta 32), reused 60 (delta 32), pack-reused 110  
Receiving objects: 100% (170/170), 51.53 KiB | 533.00 KiB/s, done.  
Resolving deltas: 100% (70/70), done.  
mgpu@mgpu-VirtualBox:~$ wget https://dumps.wikimedia.org/other/pageviews/2024/2  
024-03/pageviews-20240329-110000.gz  
--2024-03-31 00:01:27-- https://dumps.wikimedia.org/other/pageviews/2024/2024-  
03/pageviews-20240329-110000.gz  
Resolving dumps.wikimedia.org (dumps.wikimedia.org)... 208.80.154.71, 2620:0:86  
1:3:208:80:154:71  
Connecting to dumps.wikimedia.org (dumps.wikimedia.org)|208.80.154.71|:443... c  
onected.  
HTTP request sent, awaiting response... 200 OK  
Length: 57554544 (55M) [application/octet-stream]  
Saving to: 'pageviews-20240329-110000.gz'  
  
pageviews-20240329- 100%[=====] 54,89M 4,28MB/s in 13s  
  
2024-03-31 00:01:43 (4,08 MB/s) - 'pageviews-20240329-110000.gz' saved [5755454  
4/57554544]  
  
mgpu@mgpu-VirtualBox:~$ gunzip pageviews-20240329-110000.gz  
mgpu@mgpu-VirtualBox:~$
```

Рисунок 1 – Клонирование репозитория



```
mgpu@mgpu-VirtualBox:~$ gunzip pageviews-20240329-110000.gz  
mgpu@mgpu-VirtualBox:~$ awk -F ' ' '{print $1}' pageviews-20240329-110000 | sor  
t | uniq -c | sort -nr | head  
1125763 en.m  
837733 en  
288548 de.m  
268755 ru.m  
257505 ja.m  
215087 de  
209257 ru  
208388 fr  
194089 fr.m  
167990 es.m  
mgpu@mgpu-VirtualBox:~$
```

Рисунок 2 – Данные страницы «Викимедия»

Далее нам необходимо установить DBeaver. (рисунок 3-4)

```
contatiner_business_case_stocksense-webserver-1 started 7.85
mgpu@mgpu-VirtualBox:~/Ovsepyan/workshop-on-ETL/business_case_stocksense$ sudo
apt install snapd
[sudo] password for mgpu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi
  libgstreamer-plugins-bad1.0-0 libva-wayland2
Use 'sudo apt autoremove' to remove them.
The following packages will be upgraded:
  snapd
1 upgraded, 0 newly installed, 0 to remove and 43 not upgraded.
Need to get 24,4 MB of archives.
After this operation, 70,1 MB disk space will be freed.
Get:1 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 snapd amd64
2.61.3+20.04 [24,4 MB]
Fetched 24,4 MB in 7s (3 560 kB/s)
(Reading database ... 181253 files and directories currently installed.)
Preparing to unpack .../snapd_2.61.3+20.04_amd64.deb ...
Unpacking snapd (2.61.3+20.04) over (2.58+20.04.1) ...
Setting up snapd (2.61.3+20.04) ...
Installing new version of config file /etc/apparmor.d/usr.lib.snapd.snap-confi
```

Рисунок 3 – Установка DBeaver

```
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
mgpu@mgpu-VirtualBox:~/Ovsepyan/workshop-on-ETL/business_case_stocksense$ sudo
snap install dbeaver-ce
dbeaver-ce 24.0.1.202403241413 from DBeaver (dbeaver-corp) installed
mgpu@mgpu-VirtualBox:~/Ovsepyan/workshop-on-ETL/business_case_stocksense$
```

Рисунок 4 – Установка DBeaver

Добавляем новое подключение PostgreSQL. (рисунок 5)

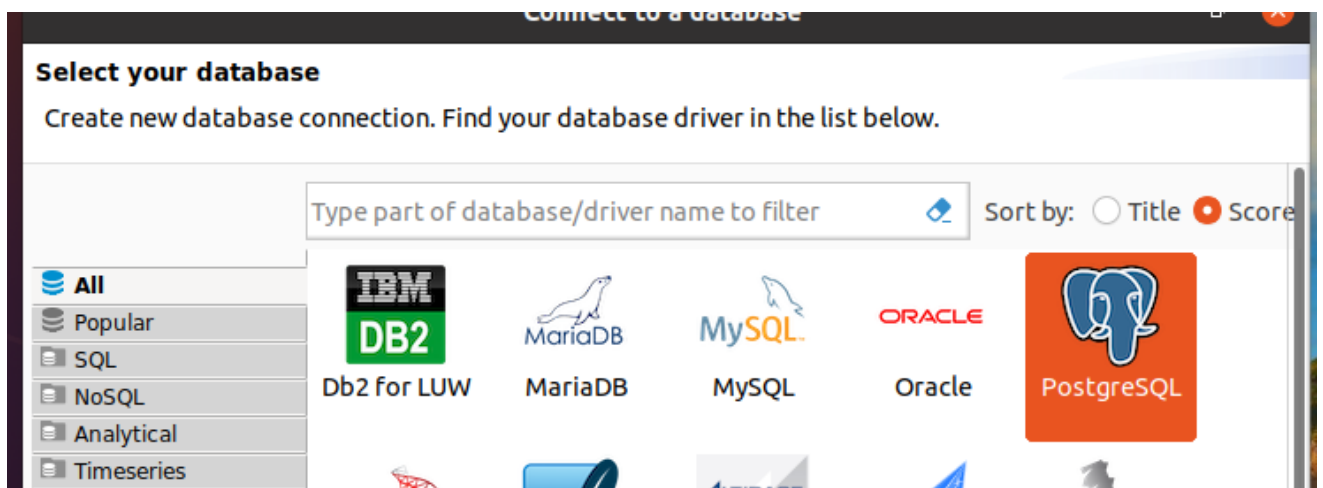


Рисунок 5 – Добавление подключения

Настраиваем базу данных. (рисунок 6)

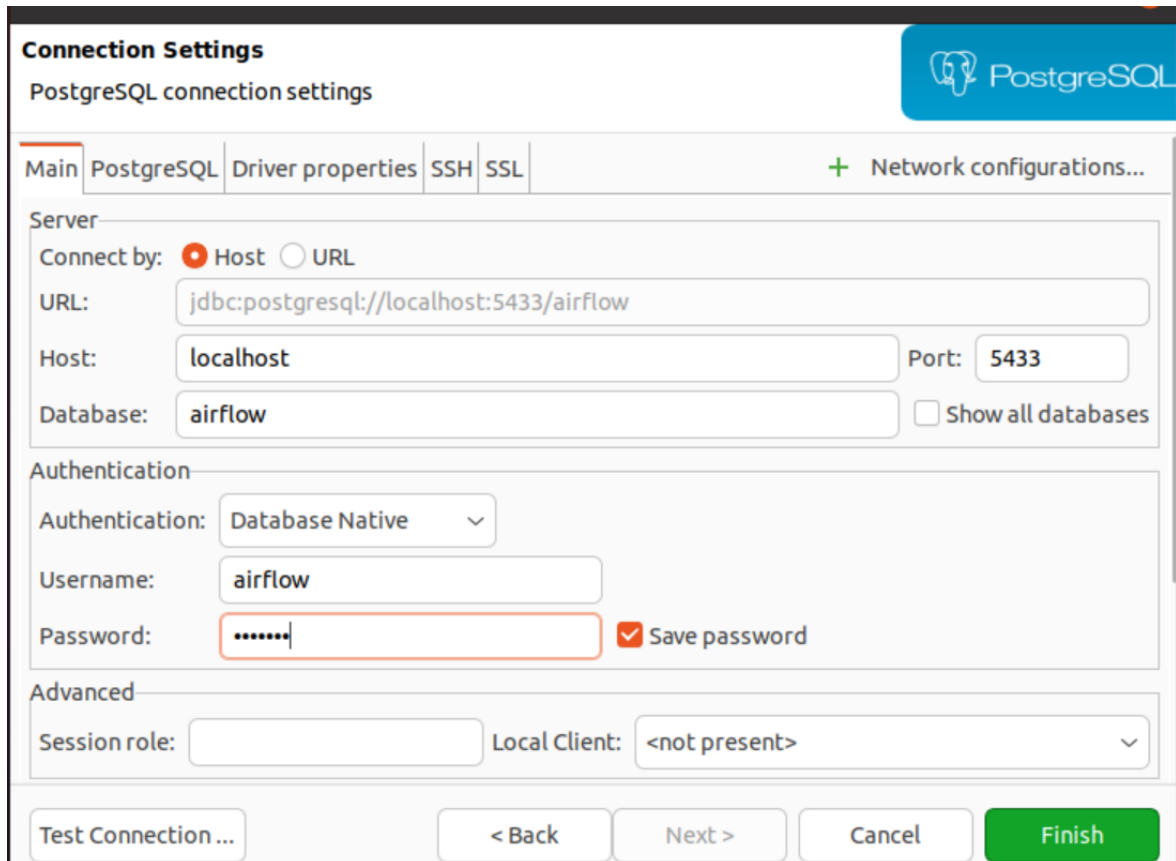


Рисунок 6 – Настройка базы данных PostgreSQL

Далее для начала работы в AirFlow нужно запускать контейнер. (рисунок 7)

```
mgpu@mgpu-VirtualBox:~/ovsepyan/workshop-on-ETL/business_case_stocksense$ sudo
docker compose up -d
[sudo] password for mgpu:
[+] Building 40.7s (12/12) FINISHED                                docker:default
=> [scheduler internal] load build definition from Dockerfile      2.5s
=> == transferring dockerfile: 208B                                0.3s
=> [webserver internal] load build definition from Dockerfile      3.0s
=> == transferring dockerfile: 208B                                0.0s
=> [init internal] load build definition from Dockerfile           2.6s
=> == transferring dockerfile: 208B                                0.0s
=> [webserver internal] load metadata for docker.io/apache/airflow:2.0. 0.0s
=> [scheduler internal] load .dockerignore                         0.9s
=> == transferring context: 2B                                     0.0s
=> [init internal] load .dockerignore                              3.2s
=> == transferring context: 2B                                     0.0s
=> [webserver internal] load .dockerignore                         2.6s
=> == transferring context: 2B                                     0.0s
=> [scheduler 1/2] FROM docker.io/apache/airflow:2.0.0-python3.8 8.6s
=> [webserver 2/2] RUN pip install --user --no-cache-dir apache-ai 24.1s
=> [init] exporting to image                                       1.3s
```

Рисунок 7 – Запуск контейнера

Заходим в AirFlow по ссылке <http://localhost:8080/>.

И запускаем только файл listing_4_20.py. (рисунок 8)

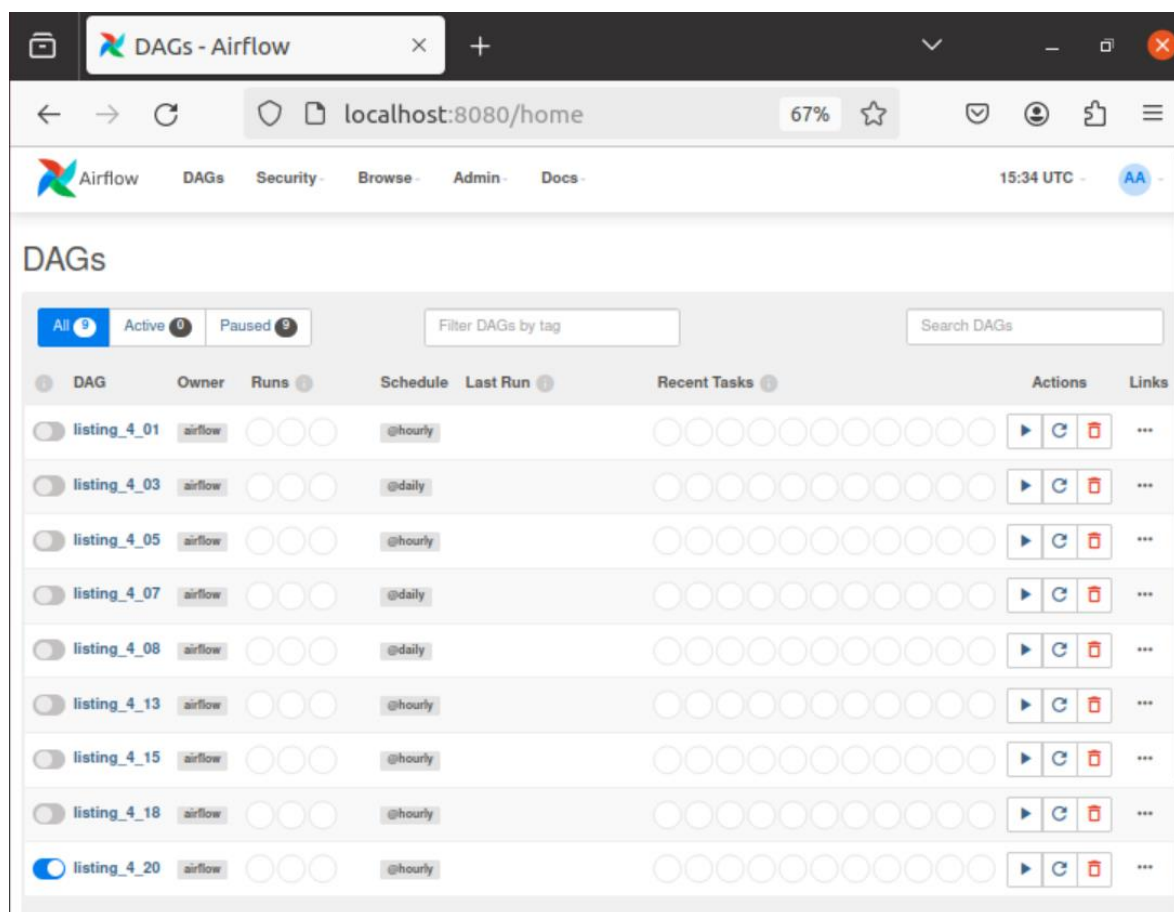
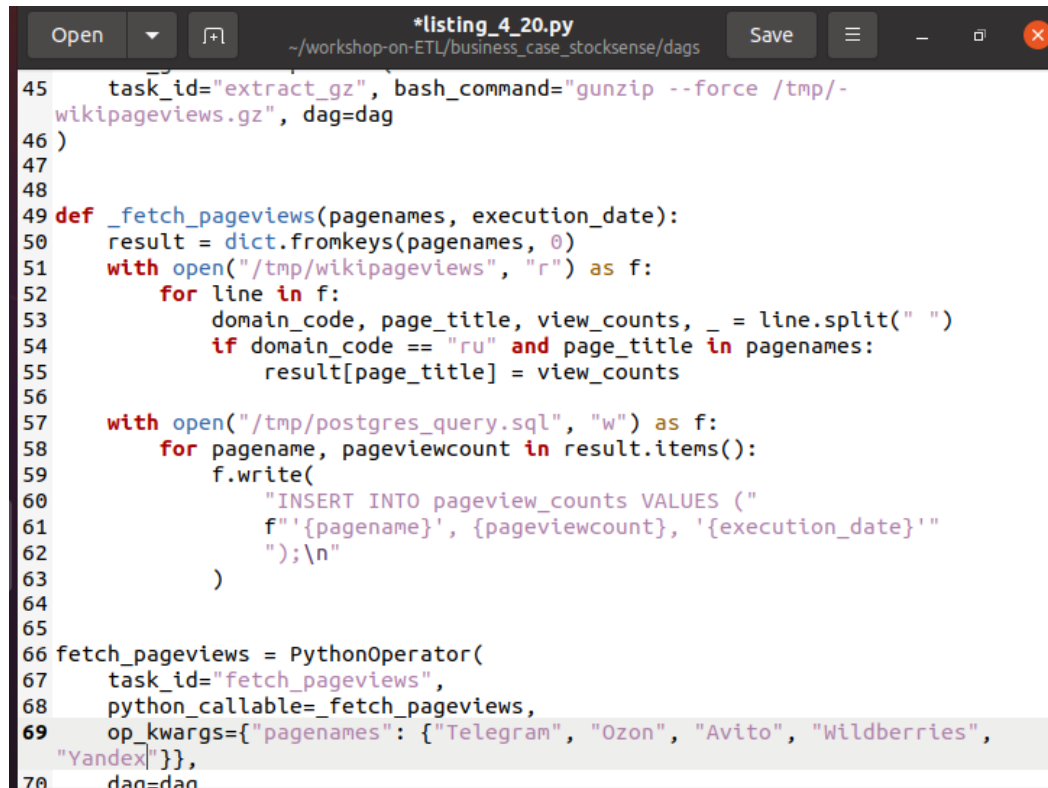


Рисунок 8 – Запуск файла listing_4_20.py

2.2 Исходный код всех DAGs

Исходный код всех DAGs, которые требовались для решения задачи представлен на рисунке 9.



```
45 task_id="extract_gz", bash_command="gunzip --force /tmp/-
    wikipageviews.gz", dag=dag
46 )
47
48
49 def _fetch_pageviews(pagenames, execution_date):
50     result = dict.fromkeys(pagenames, 0)
51     with open("/tmp/wikipageviews", "r") as f:
52         for line in f:
53             domain_code, page_title, view_counts, _ = line.split(" ")
54             if domain_code == "ru" and page_title in pagenames:
55                 result[page_title] = view_counts
56
57     with open("/tmp/postgres_query.sql", "w") as f:
58         for pagename, pageviewcount in result.items():
59             f.write(
60                 "INSERT INTO pageview_counts VALUES ("
61                 f"{pagename}', {pageviewcount}, '{execution_date}'"
62                 ");\n"
63             )
64
65
66 fetch_pageviews = PythonOperator(
67     task_id="fetch_pageviews",
68     python_callable=_fetch_pageviews,
69     op_kwargs={"pagenames": {"Telegram", "Ozon", "Avito", "Wildberries",
70 "Yandex"}},
71     dag=dag
```

Рисунок 9 – Исходный код всех DAGs

2.3 Граф DAG в Apache Airflow

Вкладка Graph View представлен на рисунке 10.

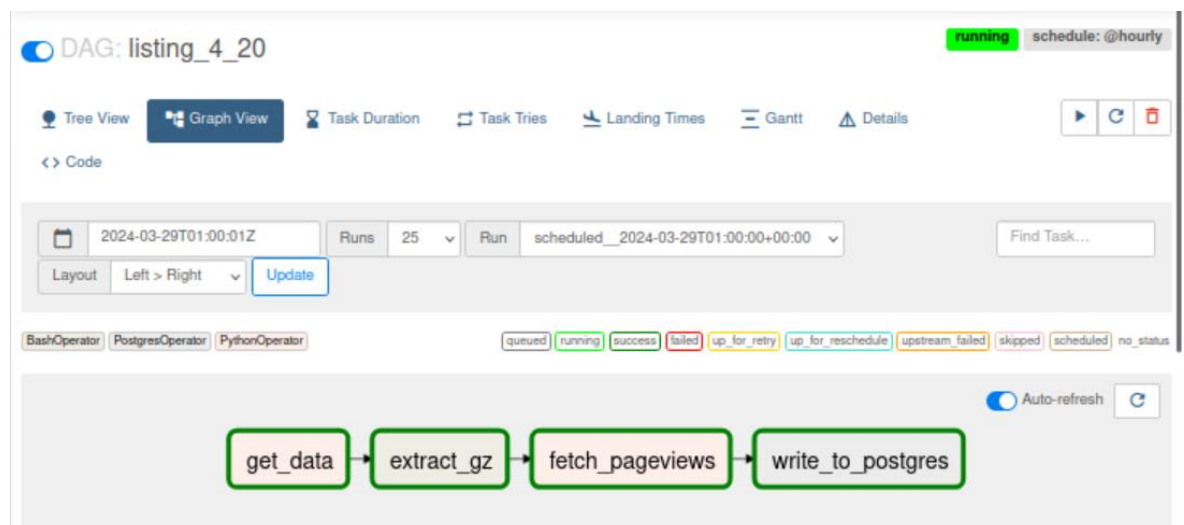


Рисунок 10 – Граф DAG

2.4 Верхнеуровневая архитектура задания Бизнес-кейса «StockSense»

Верхнеуровневая архитектура задания Бизнес-кейса «StockSense», выполненная в draw.io представлена на рисунке 11.

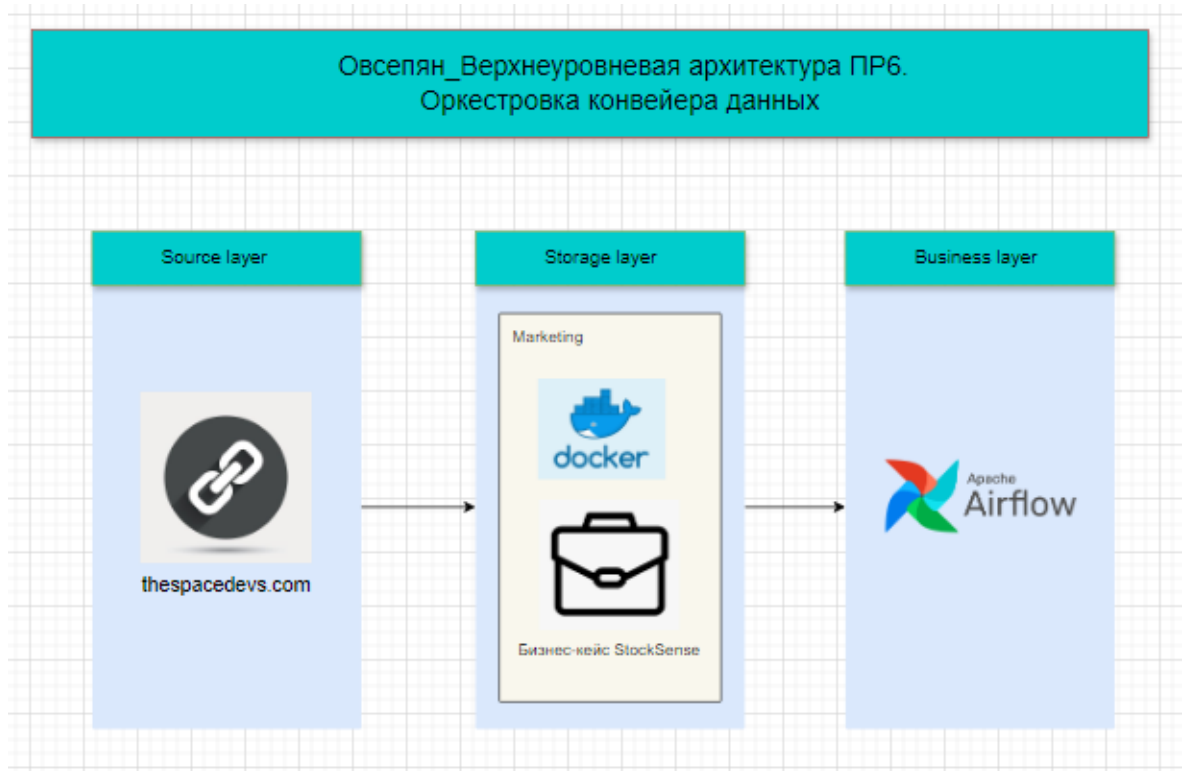


Рисунок 11 – Верхнеуровневая архитектура

Более подробно с верхнеуровневой архитектурой можно ознакомиться по ссылке:

https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1&title=%D0%9E%D0%B2%D1%81%D0%B5%D0%BF%D1%8F%D0%BD_6.drawio#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1irC0s4G2psxkYo0yIac4qzL70I-Wu1Lg%26export%3Ddownload

2.5 Архитектура DAG Бизнес-кейса «StockSense»

Архитектура DAG Бизнес-кейса «StockSense» выполненная в draw.io представлена на рисунке 12.

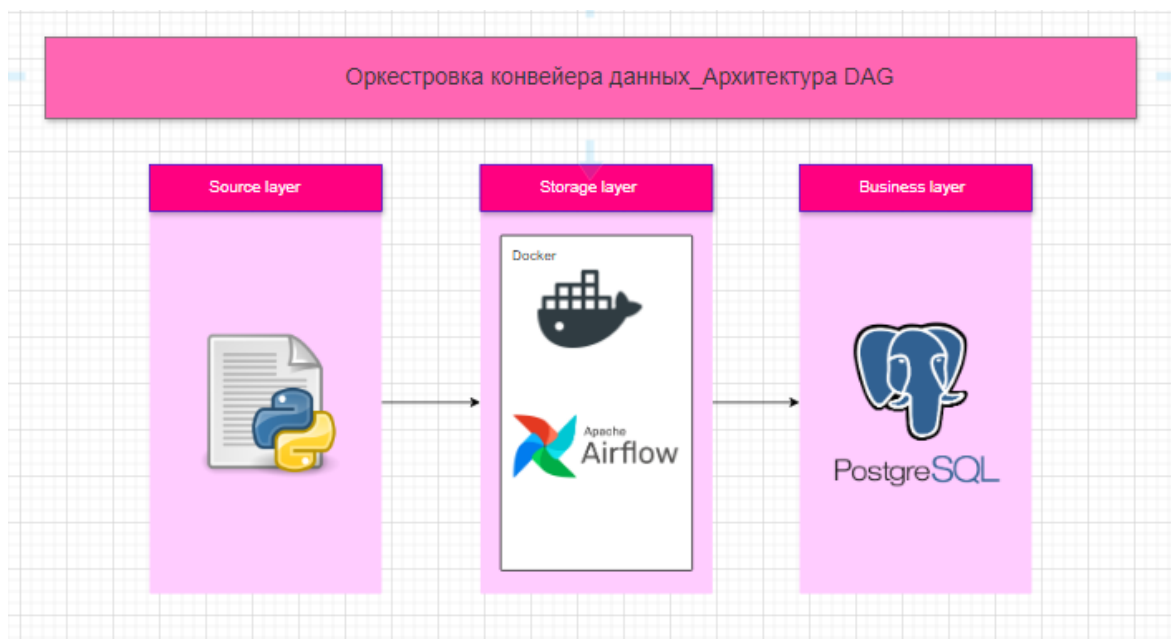


Рисунок 12 – Архитектура DAG Бизнес-кейса «Rocket»

Более подробно с архитектурой можно ознакомиться по ссылке: https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=blank&layers=1&nav=1&title=%D0%9E%D0%B2%D1%81%D0%B5%D0%BF%D1%8F%D0%BD_6%20.drawio#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1zqydzaO5dOU7Z3N0GuOwxNh9BW3w_Oun%26export%3Ddownload

2.6 Диаграмма Ганта DAG в Apache Airflow.

Диаграмма Ганта DAG в Apache Airflow представлена на рисунке 13.

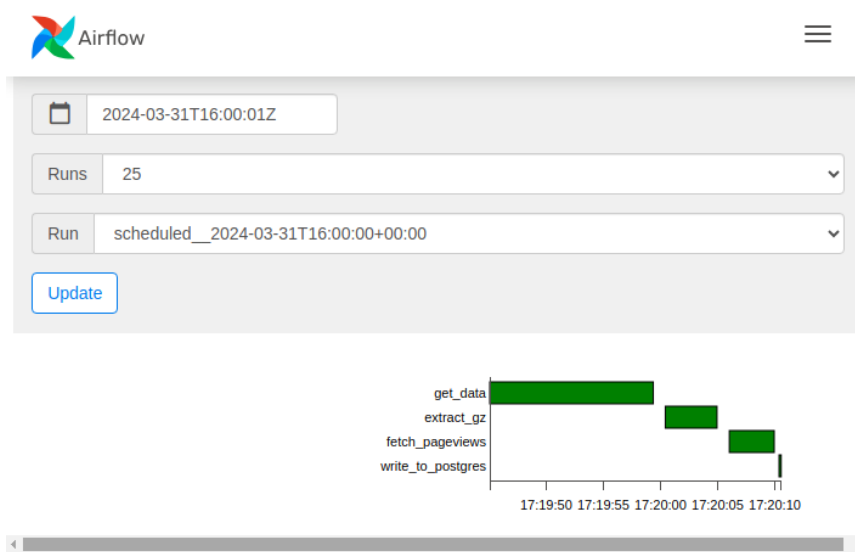


Рисунок 13 – Диаграмма Ганта DAG.

2.7 SQL-запросы, позволяющие проверить наличие выгруженных агрегированных данных бизнес-задачи

Разворачиваем базу данных PostgreSQL. После этого появляется наша таблица с данными. (рисунок 14)

	ABC pagename	123 pageviewcount	datetime
1	Wildberries	41	2024-04-02 00:00:00.000
2	Яндекс	36	2024-04-02 00:00:00.000
3	Telegram	50	2024-04-02 00:00:00.000
4	Ozon	32	2024-04-02 00:00:00.000
5	Sberbank	0	2024-04-02 00:00:00.000
6	Wildberries	50	2024-04-02 01:00:00.000
7	Яндекс	22	2024-04-02 01:00:00.000
8	Telegram	44	2024-04-02 01:00:00.000

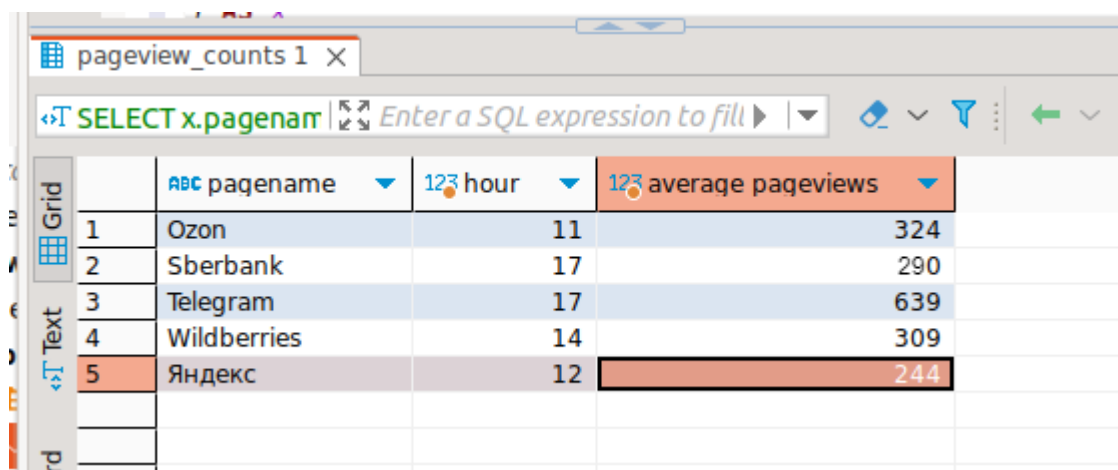
Рисунок 14 – Таблица о данных посещения страниц

Далее необходимо написать SQL-запрос, позволяющий проверить наличие выгруженных агрегированных данных бизнес-задачи. (рисунок 15)

```
SELECT x.pagename, x.hr AS "hour", x.average AS "average pageviewcount"
FROM (
  SELECT
    pagename,
    date_part('hour', datetime) AS hr,
    AVG(pageviewcount) AS average,
    ROW_NUMBER() OVER (PARTITION BY pagename ORDER BY AVG(pageviewcount)) AS row_number
  FROM pageview_counts
  GROUP BY pagename, hr
) AS x
WHERE row_number=1;
```

Рисунок 15 – SQL-запрос

Выполняем запрос. Результат запроса представлен на рисунке 16.



	ABC pagename	123 hour	123 average pageviews
1	Ozon	11	324
2	Sberbank	17	290
3	Telegram	17	639
4	Wildberries	14	309
5	Яндекс	12	244

Рисунок 16 – Результат SQL-запроса

ЗАКЛЮЧЕНИЕ

В рамках выполнения задания по анализу Бизнес-кейса "StockSense" был проделан значительный объем работы, включающий развертывание виртуальной машины, работу с Apache Airflow, проектирование архитектурных решений и агрегацию данных бизнес-процесса.

1. Виртуальная машина ubuntu_mgpr.ova была успешно развернута в VirtualBox, а задача Бизнес-кейса "StockSense" была клонирована на ПК для дальнейшего анализа.
2. С использованием Apache Airflow был запущен контейнер с кейсом, создан DAG в соответствии с предоставленным алгоритмом, и изучены основные элементы DAG. Логи выполненного DAG были изучены и успешно скачаны в основную операционную систему.
3. Агрегированные данные бизнес-процесса были выгружены в PostgreSQL для последующего анализа и обработки.
4. В draw.io была спроектирована верхнеуровневая архитектура аналитического решения задания "StockSense" и архитектура DAG для Бизнес-кейса "StockSense", с учетом Source Layer, Storage Layer и Business Layer.
5. Была построена диаграмма Ганта работы DAG в Apache Airflow для визуализации хода выполнения задач.

Эти шаги позволили успешно выполнить задание и представить все необходимые результаты исследований для анализа Бизнес-кейса "StockSense".