

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
Факультет информатики
Кафедра технической кибернетики

Отчет по лабораторной работе №1

Дисциплина: «Инженерия данных»

Тема: «Знакомство с основными инструментами построения
пайплайнов: Apache Airflow и Apache NiFi»

Выполнил: Иванов И.А.

Группа: 6231-010402D

Самара 2023

Содержание

Часть 1. Настройка программ для выполнения лабораторной работы.....	3
Шаг 1. Настройка Docker Desktop.	3
Шаг 2. Настройка Visual Studio Code	3
Шаг 3. Клонирование репозиториев.....	4
Шаг 4. Разворачивание образов и контейнеров.....	5
Шаг 4.1 Создание соединения.....	5
Шаг 4.2 Установка необходимых компонентов	5
Шаг 4.3 Развёртывание airflow, nifi, elasticsearch, postgresql, mlflow.....	6
Часть 2. Выполнение лабораторной работы №1: Apache airflow.....	10
Шаг 1. Авторизация в Apache airflow.....	10
Шаг 2. Создание собственного Directed Acyclic Graphs (DAG).....	11
Шаг 3. Получение данных для запуска собственного DAG-а.....	14
Шаг 4. Запуск DAG-а.....	15
Часть 3. Выполнение лабораторной работы №1: Apache Nifi.....	18
Шаг 1. Построение и запуск DAG-а в Apache Nifi	18
Шаг 2. Построение графиков на основе полученных данных.	18
Заключение	22

Часть 1. Настройка программ для выполнения лабораторной работы.

Шаг 1. Настройка Docker Desktop.

1.1 Клиент программы Docker Desktop доступен на сайте по ссылке <https://www.docker.com/products/docker-desktop>

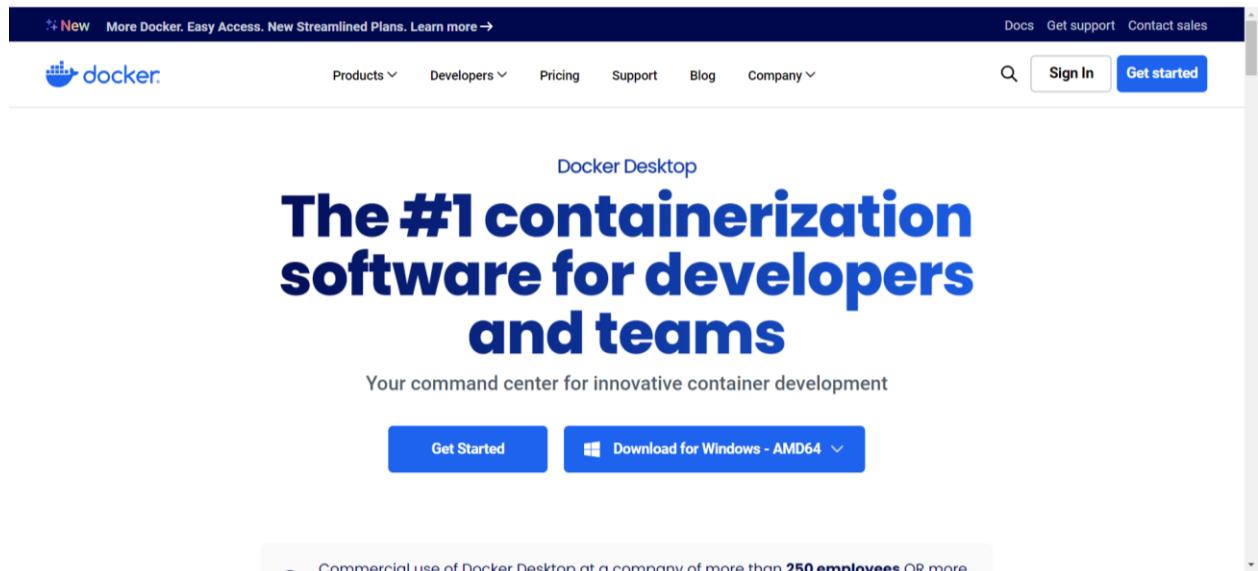


Рисунок 1 – Получение клиента программы Docker Desktop

1.2 На данном ПК потребуется произвести настройку для запуска программы Docker Desktop, более подробно об этом можно почитать на сайте по ссылке <https://docs.microsoft.com/en-us/windows/wsl/install-manual>.

```
$ dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart  
$ dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

1.3 После чего потребуется перезапуск машины. Теперь Docker Desktop установлен и запущен.

Шаг 2. Настройка Visual Studio Code

2.1. Для работы с Docker воспользуемся средой разработки Visual Studio Code, скачать которую можно на сайте по ссылке <https://code.visualstudio.com/insiders/>. Для работы буду использовать версию Insiders. В качестве installer-а использую **system installer**.

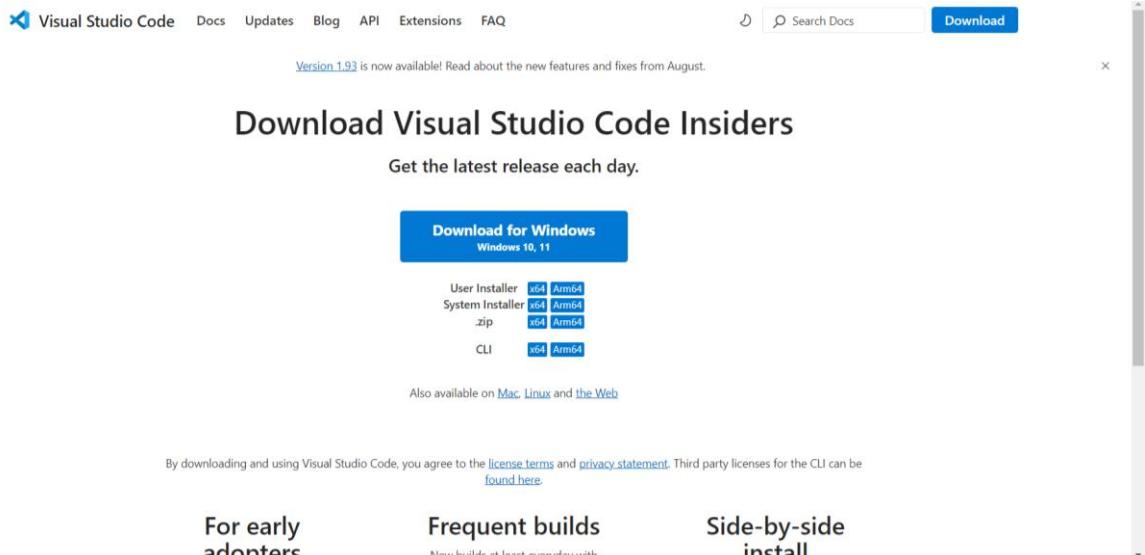


Рисунок 2 – Получение клиента программы Visual Studio Code

2.2 После установки программы, произведем ее настройку. Установим необходимые расширения для работы:

```
ms-python.python  
ms-toolsai.jupyter  
ms-vscode-remote.vscode-remote-extensionpack  
ms-azuretools.vscode-docker
```

Шаг 3. Клонирование репозиториев.

Контейнеры и образы для выполнения лабораторных работ расположены по ссылке <https://github.com/ssau-data-engineering/Prerequisites>.

Для их получения воспользуемся терминалом, и клонируем в выбранную папку необходимый репозиторий:

```
git clone https://github.com/ssau-data-engineering/Prerequisites.git  
cd .\prerequisites\
```

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS C:\Users\ivano> cd d:\
PS D:> cd study_de
PS D:\study_de> git clone https://github.com/ssau-data-engineering/Prerequisites.git
Cloning into 'Prerequisites'...
remote: Enumerating objects: 69, done.
remote: Counting objects: 100% (69/69), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 69 (delta 28), reused 56 (delta 19), pack-reused 0 (from 0)
Receiving objects: 100% (69/69), 90.39 KiB | 833.00 KiB/s, done.
Resolving deltas: 100% (28/28), done.
PS D:\study_de> cd .\prerequisites\
cd.\prerequisites\ : Имя "cd.\prerequisites\" не распознано как имя командлета, функции, файла сценария или выполняемой
программы. Проверьте правильность написания имени, а также наличие и правильность пути, после чего повторите попытку.
строка:1 знак:1
+ cd.\prerequisites\
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (cd.\prerequisites\:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS D:\study_de> cd .\prerequisites\
PS D:\study_de\prerequisites> |
```

Рисунок 5 – Клонирование репозитория.

Шаг 4. Разворачивание образов и контейнеров.

Шаг 4.1 Создание соединения

Перед разворачиванием контейнеров поднимем соединение с Docker, при помощи команды:

```
$ docker network create data-engineering-labs-network
```

```
PS D:\study_de> cd .\prerequisites\
PS D:\study_de\prerequisites> docker network create data-engineering-labs-network
cc7212c839c52ec1c28326bb10f30708a72c842580cf0eb4bc1abfa549b25351
PS D:\study_de\prerequisites> |
```

Рисунок 6 – создание соединения с Docker

Шаг 4.2 Установка необходимых компонентов

Для работы контейнеров скачаем и установим для этого необходимые компоненты при помощи команды:

```
$ docker compose -f docker-compose.airflow.yaml up airflow-init
```

```

PS D:\study_de\prerequisites> docker network create data-engineering-labs-network
c5ea9f71b9e5b3963031edc80aba4d7651bd3fa5d5d6d9d97313e55f4d7bbff
PS D:\study_de\prerequisites> docker compose -f docker-compose.airflow.yaml up airflow-init
time="2024-09-22T15:54:58+04:00" level=warning msg="D:\\study_de\\prerequisites\\docker-compose.airflow.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[*] Running 23/23
  redis Pulled
    ✓ a2318d6c47ec Pull complete
    ✓ ed7fd66f27f2 Pull complete
    ✓ 410a3d5b3155 Pull complete
    ✓ 9312cfc3fb63e Pull complete
    ✓ c39877ab23d0 Pull complete
    ✓ 01394fffc7248 Pull complete
    ✓ 4f4fb700ef54 Pull complete
    ✓ 5a03cb6163ab Pull complete
  postgres Pulled
    ✓ f29ecd7d1618 Pull complete
    ✓ 60c9fd6649d4 Pull complete
    ✓ cab8740fdd56 Pull complete
    ✓ 84ea3c162c46 Pull complete
    ✓ 9633ee13b142 Pull complete
    ✓ 0b395d80892 Pull complete
    ✓ 7d6567b63997 Pull complete
    ✓ da499088c178 Pull complete
    ✓ 85d5c9c188d3 Pull complete
    ✓ 4aa58d1adb9 Pull complete
    ✓ 4129bfcc48c3 Pull complete
    ✓ 21f4381d94e7 Pull complete
    ✓ e833a193ea76 Pull complete
[*] Building 187.7s (8/8) FINISHED
=> [airflow-init internal] load build definition from Dockerfile
=> transferring dockerfile: 103B
=> [airflow-init internal] load metadata for docker.io/apache/airflow:2.7.0
=> [airflow-init auth] apache/airflow:pull token for registry-1.docker.io
=> [airflow-init internal] load .dockerrcignore
=> transferring context: 2B
=> [airflow-init 1/2] FROM docker.io/apache/airflow:2.7.0@sha256:7a1d5453d5511e662d093c534dbab9417a3210af162e81
=> resolve docker.io/apache/airflow:2.7.0@sha256:7a1d5453d5511e662d093c534dbab9417a3210af162e81e4c51056dd2c01
=> sha256:7d676dc8a99c0e4184cd66fc4b4b1540eae31846375a23864db3bb9effab20c0 1.08MB / 1.08MB
=> sha256:9f4f7238bf54c1d67c633a35b1051c06e5187d17d8789bc3c070d84ff4fc2588c 12.90MB / 12.90MB

```

Рисунок 7 – Установка необходимых компонентов

Шаг 4.3 Развёртывание airflow, nifi, elasticsearch, postgresql, mlflow.

Шаг 4.3.1 Развёртывание airflow

Для развертывания airflow используем комнаду:

```
$ docker compose -f docker-compose.airflow.yaml up --build -d
```

```

PS D:\study_de\prerequisites> docker compose -f docker-compose.airflow.yaml up --build -d
time="2024-09-22T15:59:04+04:00" level=warning msg="D:\\study_de\\prerequisites\\docker-compose.airflow.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[*] Running 17/17
  docker-proxy Pulled
    ✓ c6a83fedfae6 Pull complete
    ✓ 3cb0c5068d02 Pull complete
    ✓ 4f4fb700ef54 Pull complete
    ✓ c26a89106685 Pull complete
    ✓ 9fc3e39771f4 Pull complete
    ✓ c93f0fd2a2a8c Pull complete
    ✓ 40eb2cf48e46 Pull complete
    ✓ cb6ad0fbabff Pull complete
    ✓ 5afcecc5c4253 Pull complete
    ✓ 10fcfa464d7a8 Pull complete
    ✓ 50814ed1ee34 Pull complete
    ✓ e2700e97a1b Pull complete
    ✓ 69387216db1 Pull complete
    ✓ c002566d79d9 Pull complete
    ✓ 7cc7640b5de98 Pull complete
    ✓ 6ec4f22cd29c Pull complete
[*] Building 2.6s (23/23) FINISHED
=> [airflow-init internal] load build definition from Dockerfile
=> transferring dockerfile: 103B
=> [airflow-scheduler internal] load metadata for docker.io/apache/airflow:2.7.0
=> [airflow-init internal] load .dockerrcignore
=> transferring context: 2B
=> [airflow-triggerer 1/2] FROM docker.io/apache/airflow:2.7.0@sha256:7a1d5453d5511e662d093c534dbab9417a3210af162e81e4c51056dd2c01a84e
=> CACHED [airflow-triggerer 2/2] RUN pip install --no-cache-dir mlflow
=> [airflow-init] exporting to image
=> exporting layers
=> writing image sha256:1fcf6725f2e3d29fa8544135daca1020b3602de23078508e3997dee91754c411
=> naming to docker.io/library/airflow-airflow-init
=> [airflow-init] resolving provenance for metadata file
=> [airflow-webserver internal] load build definition from Dockerfile
=> transferring Dockerfile: 103B
=> [airflow-scheduler internal] load build definition from Dockerfile
=> transferring Dockerfile: 103B
=> [airflow-triggerer internal] load build definition from Dockerfile

```

Рисунок 8 - Развёртывание airflow

Шаг 4.3.2 Развёртывание nifi

Для развертывания nifi используем комнаду:

```
$ docker compose -f docker-compose.nifi.yaml up --build -d
```

```
PS D:\study_de\prerequisites> docker compose -f docker-compose.nifi.yaml up --build -d
time="2024-09-22T16:10:00+04:00" level=warning msg="D:\\study_de\\prerequisites\\docker-compose.nifi.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 8/8
  Container airflow-docker-proxy-1      Started          0.4s
  Container airflow-redis-1            Healthy          0.1s
  Container airflow-postgres-1        Healthy          0.0s
  Container airflow-airflow-init-1    Exited           0.0s
  Container airflow-airflow-scheduler-1 Started          0.0s
  Container airflow-airflow-webserver-1 Started          0.0s
  Container airflow-airflow-worker-1   Started          0.0s
  Container airflow-airflow-triggerer-1 Started          0.0s
[+] Running 8/8
  ✓ Container airflow-docker-proxy-1      Started          0.6s
  ✓ Container airflow-redis-1            Healthy          1.6s
  ✓ Container airflow-postgres-1        Healthy          1.6s
  ✓ Container airflow-airflow-init-1    Exited           18.1s
  ✓ Container airflow-airflow-scheduler-1 Started          19.1s
  ✓ Container airflow-airflow-webserver-1 Started          19.5s
  ✓ Container airflow-airflow-worker-1   Started          19.0s
  ✓ Container airflow-airflow-triggerer-1 Started          19.0s
PS D:\study_de\prerequisites> docker compose -f docker-compose.nifi.yaml up --build -d
time="2024-09-22T16:10:00+04:00" level=warning msg="D:\\study_de\\prerequisites\\docker-compose.nifi.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 13/13
  - apache-nifi [=====] 1.265GB / 1.721GB Pulling          18.0s
    ✓ 99de9192b4af Pull complete          6.4s
    ✓ dabb03f19f5a Pull complete          7.9s
    ✓ 76db96e0ce9d Pull complete          9.0s
    ✓ ec22c511cb53 Pull complete          9.0s
    ✓ d0bf00dc5a1f Pull complete          9.0s
    ✓ f52c88e7731e Pull complete          9.1s
    ✓ b7707083ba23 Pull complete          9.1s
    ✓ c861ed6b204d Pull complete          9.1s
    ✓ a00bf41fb2b8 Pull complete          30.1s
    - 364328961ff8 Downloading [=====] 1.021GB/1.477GB          177.8s
    ✓ 429665d61370 Download complete          14.5s
    ✓ 4f4fb700ef54 Download complete          15.9s
  |
```

Рисунок 9 - Развёртывание nifi

Шаг 4.3.3 Развёртывание elasticsearch

Для развёртывания elasticsearch используем комнаду:

```
$ docker compose -f docker-compose.elasticsearch.yaml up --build -d
```

```
PS D:\study_de\prerequisites> docker compose -f docker-compose.elasticsearch.yaml up --build -d
time="2024-09-22T16:15:01+04:00" level=warning msg="D:\\study_de\\prerequisites\\docker-compose.elasticsearch.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 4/7
  - elasticsearch-kibana [=====] 143.7MB / 536.4MB Pulling          9.2s
    ✓ lefc2764ff9 Pull complete          3.7s
    ✓ a2f2f93da482 Pull complete          3.8s
    ✓ 12cca292b13c Pull complete          3.8s
    ✓ d73cf48caac Pull complete          6.7s
  - e0d30173d675 Extracting [====>] 1.311MB/15.34MB          6.9s
  - 4ef79ea5ec0f Downloading [=====] 49.35MB/442.1MB          6.9s
  |
```

Рисунок 10 - Развёртывание elasticsearch

Шаг 4.3.4 Развёртывание posgresql

Для развёртывания posgresql используем комнаду:

```
$ docker compose -f docker-compose.postgresql.yaml up --build -d
```

```

PS D:\study_de\prerequisites> docker compose -f docker-compose.postgresql.yaml up --build -d
time="2024-09-22T16:16:25+04:00" level=warning msg="D:\\study_de\\prerequisites\\docker-compose.postgresql.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 1/1
  Container elasticsearch-elasticsearch-kibana-1 Started
PS D:\study_de\prerequisites>

```

The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command `docker compose -f docker-compose.postgresql.yaml up --build -d` is being run. The output lists several Docker images being pulled, including `elasticsearch`, `kibana`, and `postgresql`. A warning message is displayed about the 'version' attribute being obsolete. The PowerShell interface includes a taskbar at the bottom with icons for File Explorer, Task View, and various applications.

Рисунок 11 - Развёртывание postgresql

Шаг 4.3.5 Развёртывание mlflow

Для развертывания mlflow используем комнаду:

```
$ docker compose -f docker-compose.mlflow.yaml up --build -d
```

```

PS D:\study_de\prerequisites> docker compose -f docker-compose.mlflow.yaml up --build -d
time="2024-09-22T16:17:08+04:00" level=warning msg="D:\\study_de\\prerequisites\\docker-compose.mlflow.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 19/21
  db [██████] 109.2MB / 109.4MB Pulling
    a316717fc6ee Pull complete
    b64762744f75 Extracting [=====]
      ] 32.31MB/66.81MB
    a1f742e3aa43 Download complete
    f71a5f0dcc26 Download complete
  minio Pulled
  f2f8f30a646a Pull complete
  4fc5fcbe9ca Pull complete
  53140b7af7f0 Pull complete
  2bbc182be72c Pull complete
  4363be478988 Pull complete
  de7c323e2901 Pull complete
  7eb10b7b511b Pull complete
  3d82d69301e2 Pull complete
  mc Pulled
    55360c0b72d6 Pull complete
    029556cb2748 Pull complete
    5f9665cb7cf0 Pull complete
    2e47722e1257 Pull complete
    ee70e41b4c95 Pull complete
    d77f5efee8837 Pull complete

```

The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command `docker compose -f docker-compose.mlflow.yaml up --build -d` is being run. The output lists several Docker images being pulled, including `db`, `minio`, `mc`, and `mlflow`. A warning message is displayed about the 'version' attribute being obsolete. The PowerShell interface includes a taskbar at the bottom with icons for File Explorer, Task View, and various applications.

Рисунок 12 – Развёртывание mlflow

Шаг 4.3.6 Проверка развертывания

После выполнения команды из пункта 4.3.4 по развертыванию mlflow.

Посмотрим в докер, и убедимся, что все запущено и работает

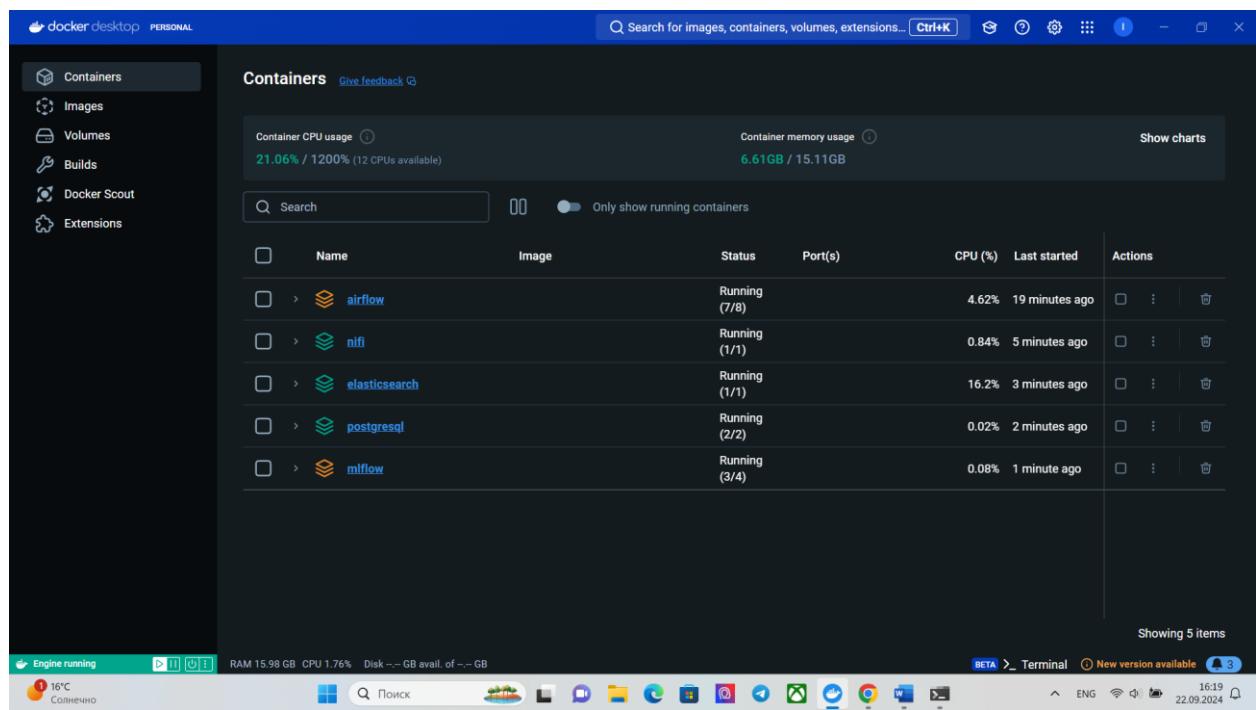
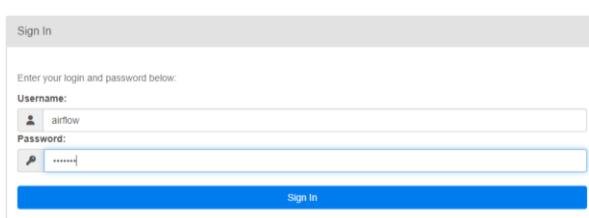


Рисунок 13 – Запущенные контейнеры в Docker.

Часть 2. Выполнение лабораторной работы №1: Apache airflow

Шаг 1. Авторизация в Apache airflow.

После того как мы развернули необходимые контейнеры, переходим по ссылке <http://127.0.0.1:8080/login/>. После чего мы попадаем страницу входа в Apache airflow. Вводим регистрационные данные airflow/airflow и выполняем вход в систему.



The screenshot shows a 'Sign In' dialog box. At the top, it says 'Enter your login and password below:'. Below that, there are two input fields: 'Username:' with 'airflow' typed into it, and 'Password:' with a masked password. At the bottom right of the dialog is a blue 'Sign In' button.

Рисунок 14 – Страница регистрации Apache airflow

После выполнения входа в Apache airflow, мы попадаем на главную страницу.

The screenshot shows the Apache Airflow web interface at the URL 127.0.0.1:8080/home. The title bar includes tabs for various documentation and setup guides. The main navigation bar has links for Course, SkillFactory, Gmail, YouTube, Карты, Илья Иванов, Курс: Искусствен..., GitHub, and Adobe Acrobat. The top right shows the time as 12:47 UTC and a user icon. The page title is "DAGs". Below it, there are filters for "All 53", "Active 0", "Paused 53", "Running 0", and "Failed 0". A search bar and an "Auto-refresh" button are also present. The main content area displays a table of DAGs with columns for Name, Owner, Runs, Schedule, Last Run, Next Run, and Recent Tasks. The table lists several DAGs, each with a status indicator (e.g., green for active, grey for paused), owner (airflow), and various scheduling details. The interface is set against a light blue background with white and grey text.

Рисунок 16 – Главная страница Apache airflow

Шаг 2. Создание собственного Directed Acyclic Graphs (DAG).

Используя Directed Acyclic Graphs (DAG) реализуем пайплайн обработки данных, в файлах CSV. Схема пайплайна приведена на рисунке ниже.

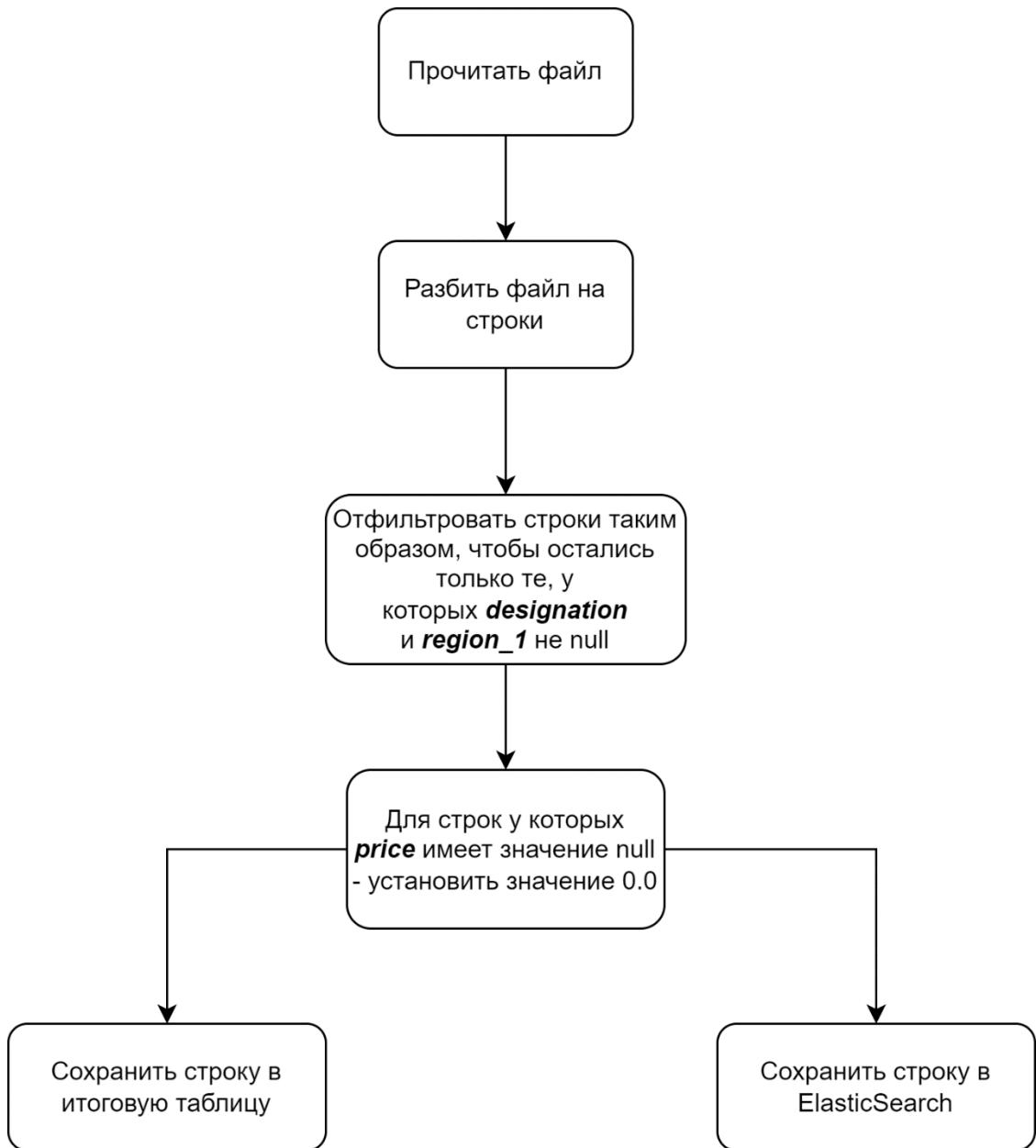


Рисунок 17 – Схема, описывающая пайплайн.

После того как определились с алгоритмом решения задачи, напишем код, который и будет решать ее. В процессе написания кода, мы определили DAG и его параметры, описали функции `extract_load_transform_data()` и `load_in_elastic()`.

`extract_load_transform_data()` – функция, обеспечивающая предварительную обработку данных:

Считывает данные из файлов;

Объединяет данные в единый датафрейм;

Применяем условие на датафрейм: оставляем строки, где designation и region_1 не NULL;

В колонке price значения NULL заменяем на числовое значение “0.0”;

Сохраняем обработанную таблицу в CSV-файл.

`load_in_elastic()` – функция, обеспечивающая загрузку данных в elastic:

Получаем данные

Преобразовываем данные в json-формат

Отправляем полученные данные в elastic

Часть кода представлена на рисунке ниже, полная версия решения представлена в репозитории лабораторной работы.

```
D:\study_de>Prerequisites>airflow>dags>◆ my_dag.py > ...
1  from airflow import DAG
2  from airflow.operators.python import PythonOperator
3  from elasticsearch import Elasticsearch
4  from datetime import datetime, timedelta
5  import pandas as pd
6  import numpy as np
7  import uuid
8
9  default_args = {
10      'owner': 'IlyaSwallow',
11      'start_date': datetime(2024, 9, 22),
12  }
13
14 dag = DAG(
15     'Joe_Peach',
16     default_args=default_args,
17     catchup=False,
18 )
19
20 def extract_transform_load_data():
21     result_dataframe = pd.DataFrame()
22     for i in range(26):
23         current_chunk = pd.read_csv(f"/opt/airflow/data/chunk({i}).csv")
24         result_dataframe = pd.concat([result_dataframe, current_chunk])
25
26     result_dataframe = result_dataframe[
27         ~(result_dataframe['designation'].isnull())
28         &
29         ~(result_dataframe['region_1'].isnull())
30     ]
31     result_dataframe['price'] = result_dataframe['price'].replace(np.nan, 0)
32     result_dataframe = result_dataframe.drop(['id'], axis=1)
33
34     result_dataframe.to_csv('/opt/airflow/data/data.csv', index=False)
35
36 etl_task = PythonOperator(
37     task_id='etl_task',
```

Рисунок 18 – Реализация DAG-а

Готовый код сохраняем, и копируем в папку, в которой развернули контейнер, в моем случае путь был такой: `../Prerequisites/airflow/dags/dags.py`

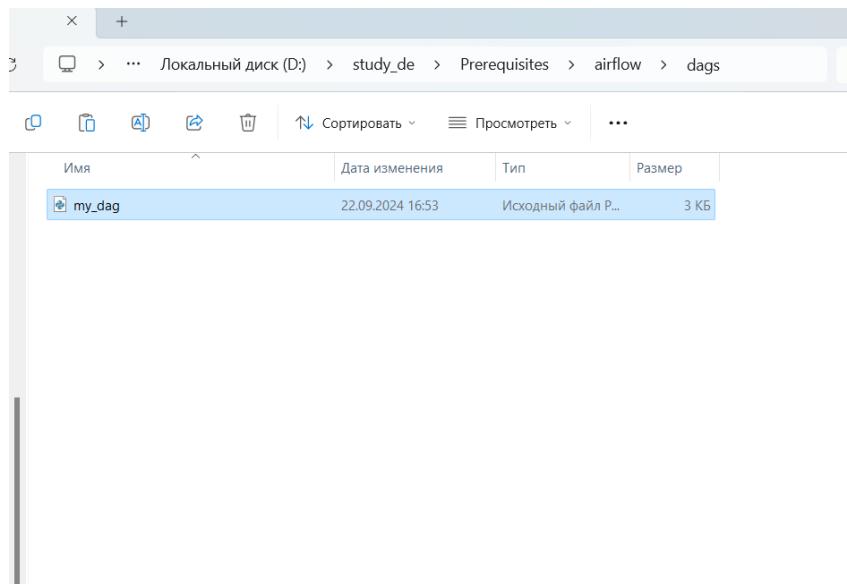


Рисунок 19 – Сохранение кода DAG-а

Шаг 3. Получение данных для запуска собственного DAG-а.

Для продолжения выполнения лабораторной работы, нам необходимы данные. Их можно получить из репозитория лабораторной работы №1. Поэтому клонируем, в соседнюю от подготовительной папки, где развернуты контейнеры, папку с репозиторием первой лабораторной работы.

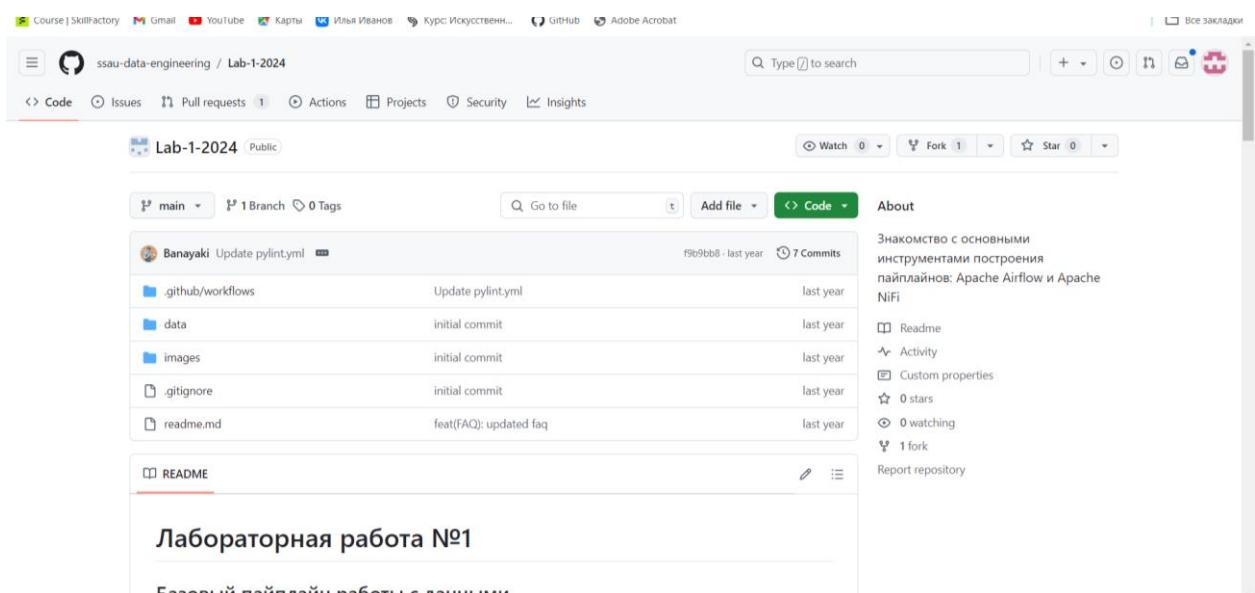


Рисунок 20 – Клонирование репозитория лабораторной работы №1

После клонирования репозитория, переходим в папку data, где расположены необходимые нам данные и копируем их в аналогичную папку data, но расположенную директории Airflow, в моем случае это директория/Prerequisites/airflow/data/.

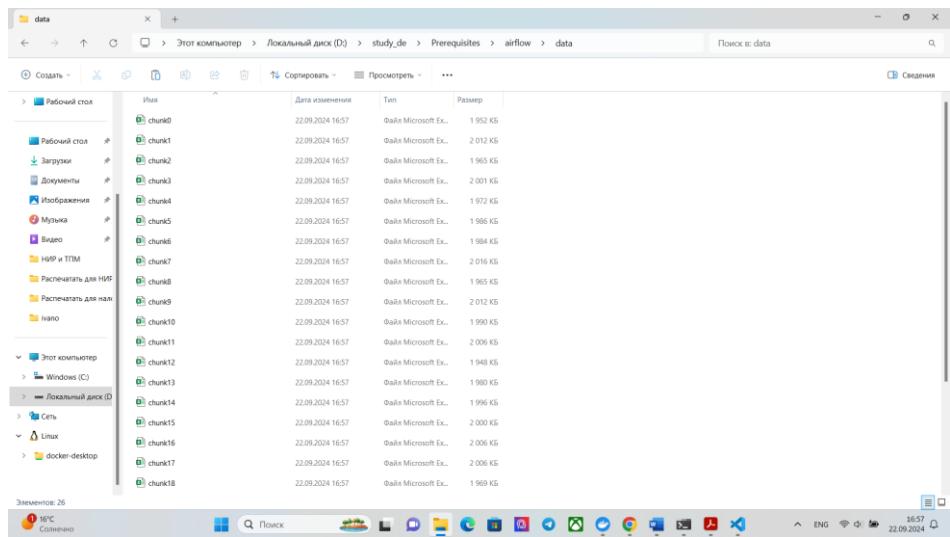


Рисунок 21 – Копирование данных для работы DAG-а

Шаг 4. Запуск DAG-а.

По окончании копирования данных в пункте 3, возвращаемся в браузер.

Ожидаем пару минут, после чего сможем найти свой собственный DAG.

Рисунок 22 – Поиск собственного DAG-а.

После того как мы нашли собственный DAG, запускаем.

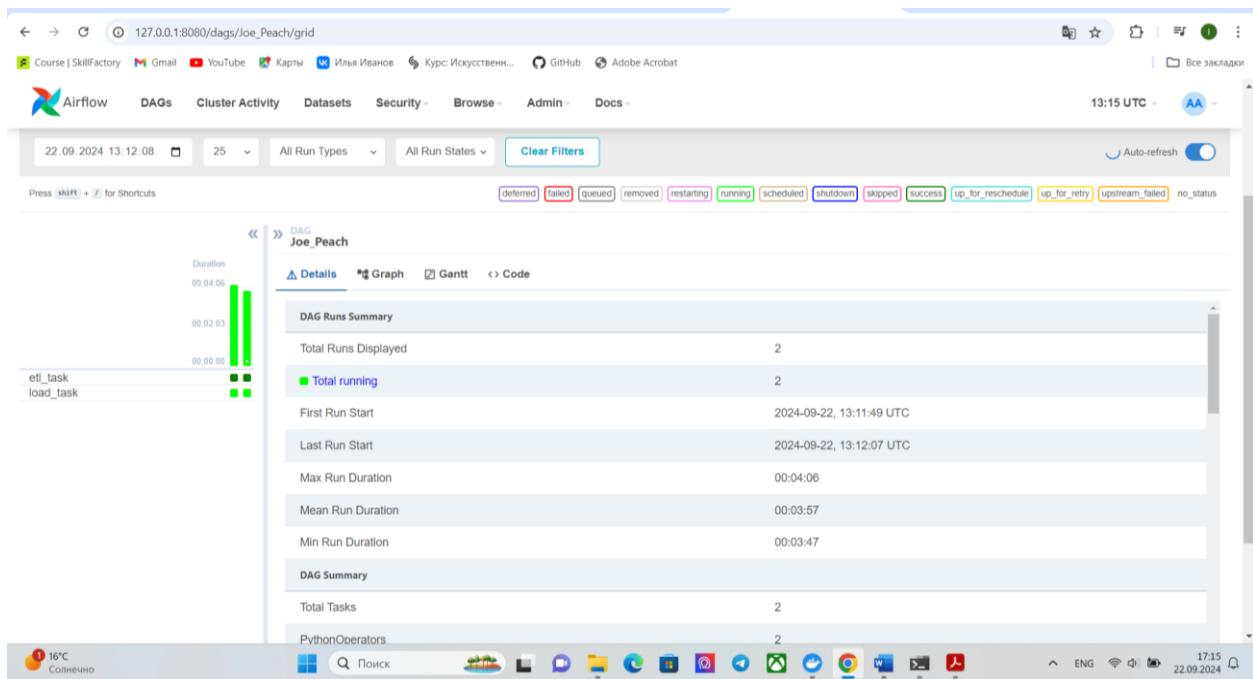


Рисунок 23 – Запуск DAG-а.

Ожидаем окончание работы DAG-а. Мы должны получить success на весь DAG.

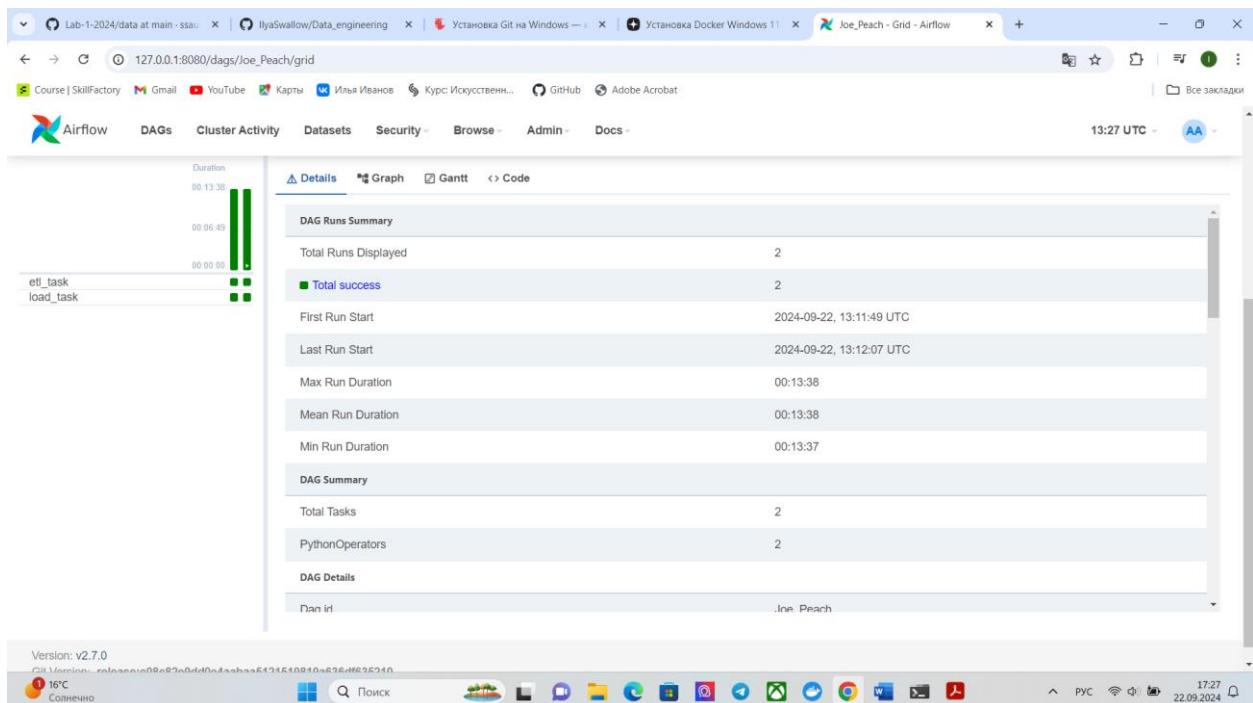


Рисунок 24 – Результат работы DAG-а

По окончании работы DAG-а, выводим все данные в ElasticSearch.

```

1 GET /dag_wines/_search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }

1 #! Elasticsearch built-in security features are not enabled. Without
2 authentication, your cluster could be accessible to anyone. See https://www
3 .elastic.co/guide/en/elasticsearch/reference/7.17/security-minimal-setup.html
4 enable security.
5
6 {
7   "took": 2255,
8   "timed_out": false,
9   "shards": {
10    "total": 1,
11    "successful": 1,
12    "skipped": 0,
13    "failed": 0
14  },
15  "hits": {
16    "total": {
17      "value": 10000,
18      "relation": "gte"
19    },
20    "max_score": 1.0,
21    "hits": [
22      {
23        "_index": "dag_wines",
24        "_type": "_doc",
25        "_id": "5fe8db72-cc70-4a4e-83f0-4ee289c7d560",
26        "_score": 1.0,
27        "_source": {
28          "country": "Argentina",
29          "description": "Smashed berry aromas are backed by earth and toasty notes. Fairly hard and tannic on the palate, this shows herbal plum flavors in front of a bold finish with dry tannins and a hint of spice.",
30          "designation": "Grand Reserve",
31          "points": 87,
32          "price": 20.0,
33          "province": "Other",
34          "region_1": "San Juan",
35        }
36      }
37    ]
38  }
39}

```

Рисунок 25 – Загрузка данных в ElasticSearch

Теперь же можем построить график, визуализируя зависимость цены вина от количества его очков.

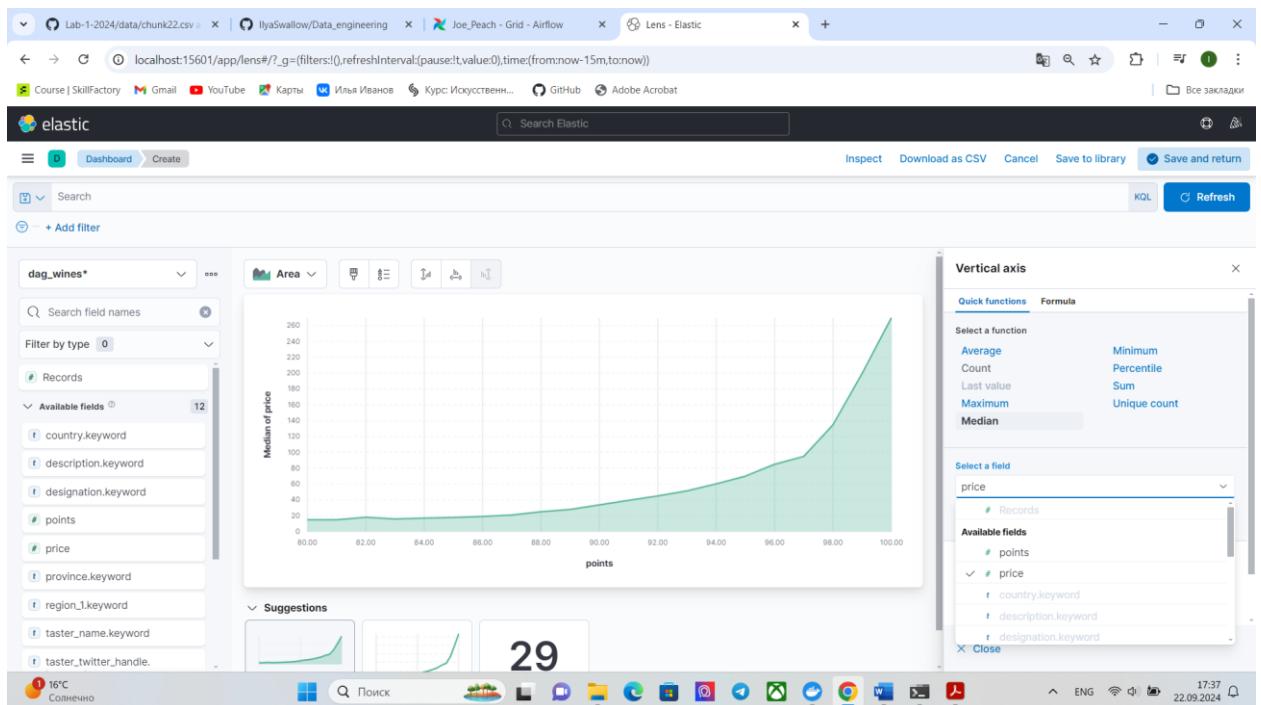


Рисунок 26 – Построение графика.

Продолжим работу в APACHE NIFI.

Часть 3. Выполнение лабораторной работы №1: Apache Nifi

Шаг 1. Построение и запуск DAG-а в Apache Nifi

В Apache Nifi пайплин реализован при помощи построения схемы.

Общая схема всего пайплайна представлена на рисунке ниже.

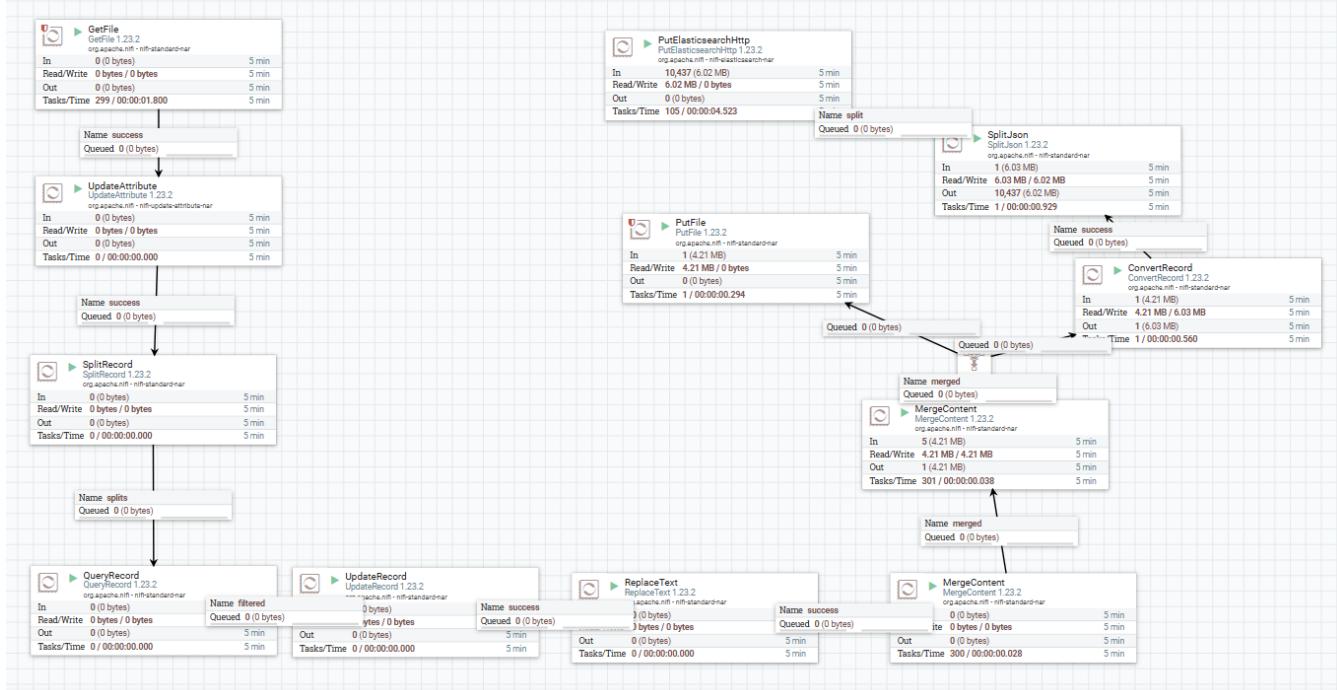


Рисунок 27 – Общая схема пайплайна Apache Nifi

Основными узлами схемы являются процессоры - GetFile, UpdateAttribute, SplitRecord, QueryRecord, UpdateRecord, MergeContent(x2), PutFile, ConertRecord, SplitJson, PutElasticsearchHttp

Далее запускаем пайплин, и дожидаемся окончания его работы. Здесь стоит уточнить, что при запуске необходимо в начале включить только GetFile, дождаться его окончания работы выключить его, только после этого запускать оставшуюся часть пайплайна.

По окончании работы, сохраняем шаблон в XML. Шаблон размещен в репозитории с решением лабораторной работы.

Шаг 2. Построение графиков на основе полученных данных.

По окончании работы DAG-а перейдем в Elasticsearch

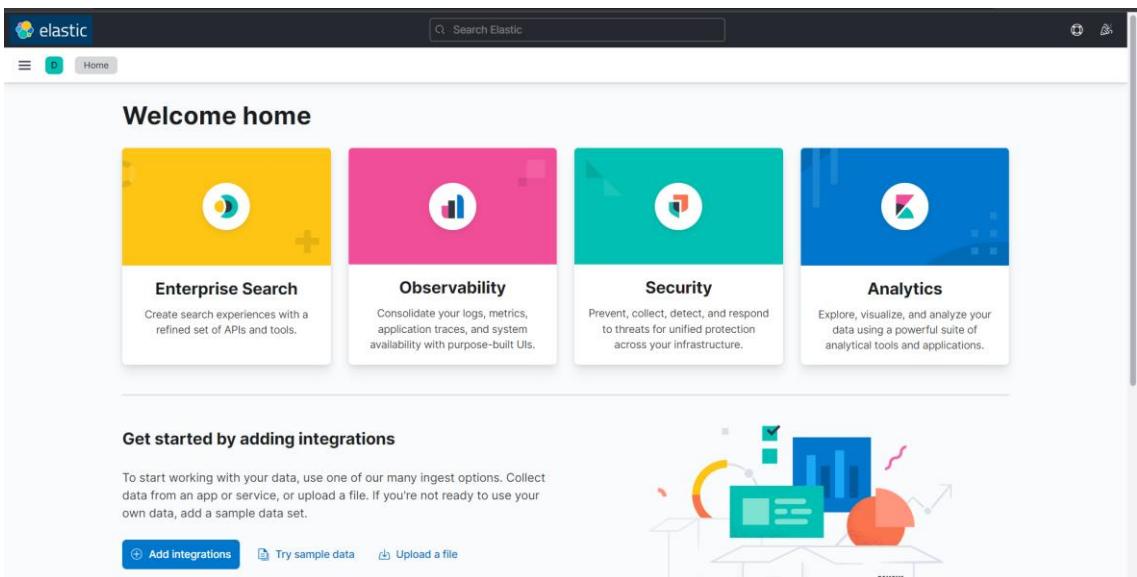


Рисунок 46 – Главная страница Elasticsearch

Проверим полученные результаты: для этого переходим по ссылкам:
Stack Management >> Index Management, где видим наши данные.

The screenshot shows the 'Index Management' page under 'Stack Management'. The left sidebar includes links for Management, Ingest, Data, Index Management, Alerts and Insights, Kibana, and more. The main area has tabs for Indices, Data Streams, Index Templates, and Component Templates. The Indices tab is selected, showing a table with one entry: 'nifi'. The table columns are Name, Health, Status, Primaries, Replicas, Docs count, Storage size, and Data stream. The 'nifi' entry shows a yellow health status, open status, 1 primary, 1 replica, 44441 documents, and 41.9mb storage size. There are also 'Include rollup indices' and 'Include hidden indices' checkboxes, a 'Search' input field, and a 'Reload indices' button. The bottom of the screen shows a Windows taskbar with various icons and system status.

Рисунок 47 – Проверка результатов

Далее создадим Index patterns. Для этого перейдем по ссылкам: Stack Management >> Index patterns, и нажмем Create index pattern:

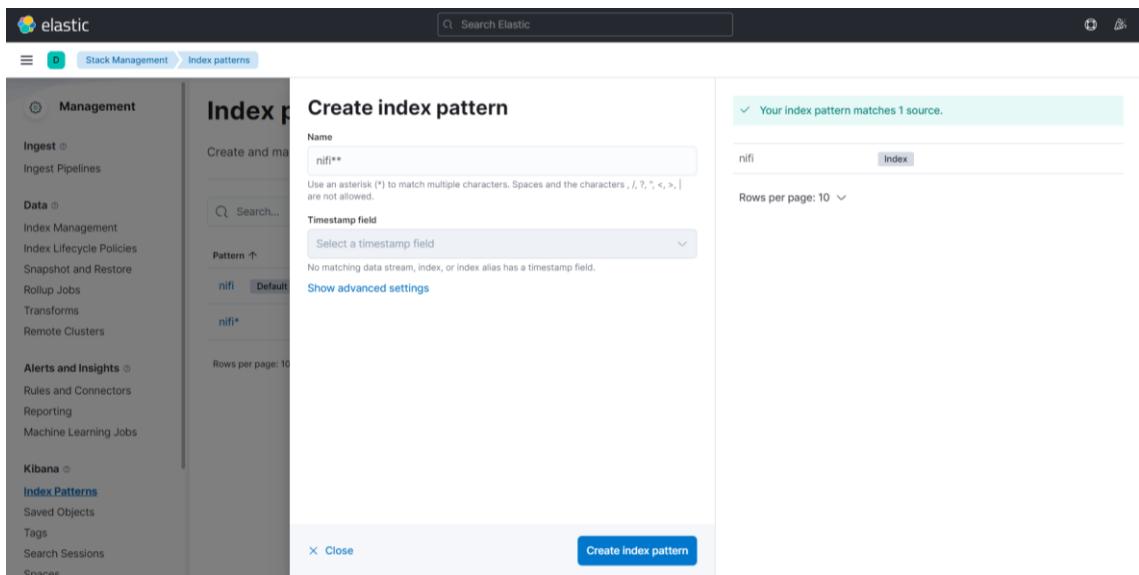


Рисунок 48 – Create index pattern

На основе полученных данных построим график. Переходим по ссылке Dashboard >> Editing New Dashboard и нажимаем на кнопку Create visualization

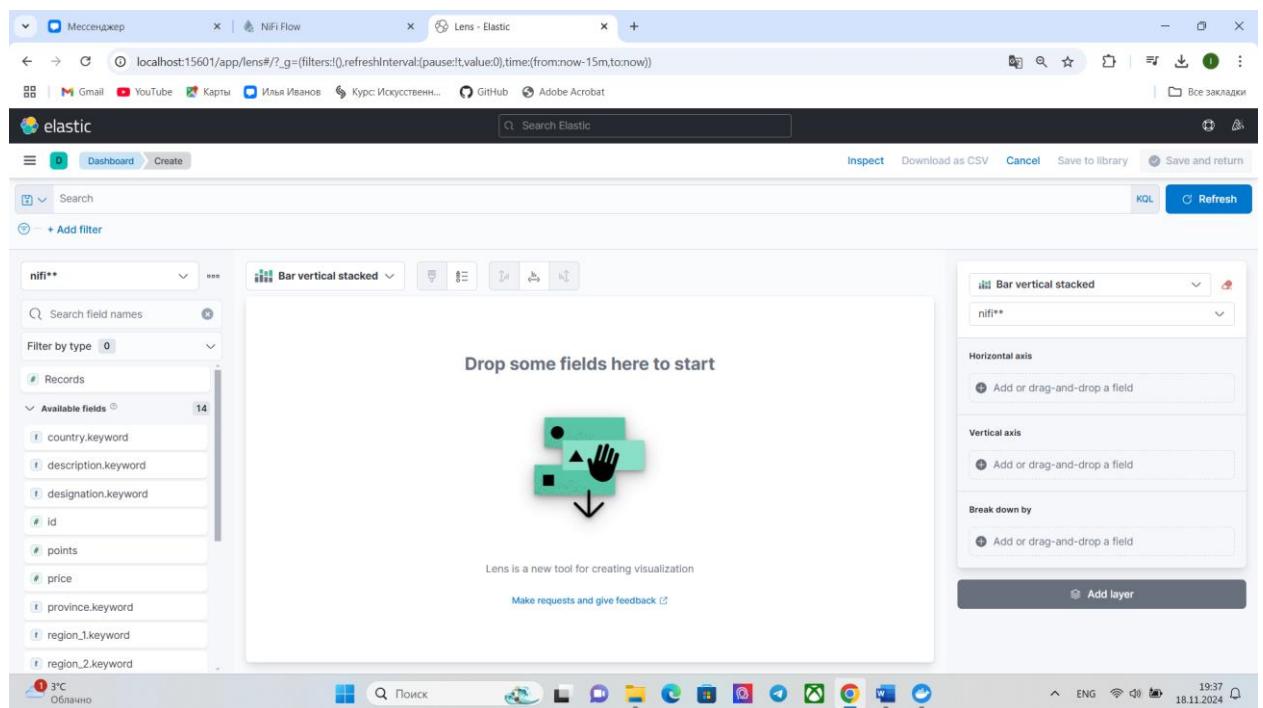


Рисунок 49 - Создание нового Dashboard-a

Передаем на вертикальную ось поле «points» на вертикальную «price». В итоге получаем следующий график:

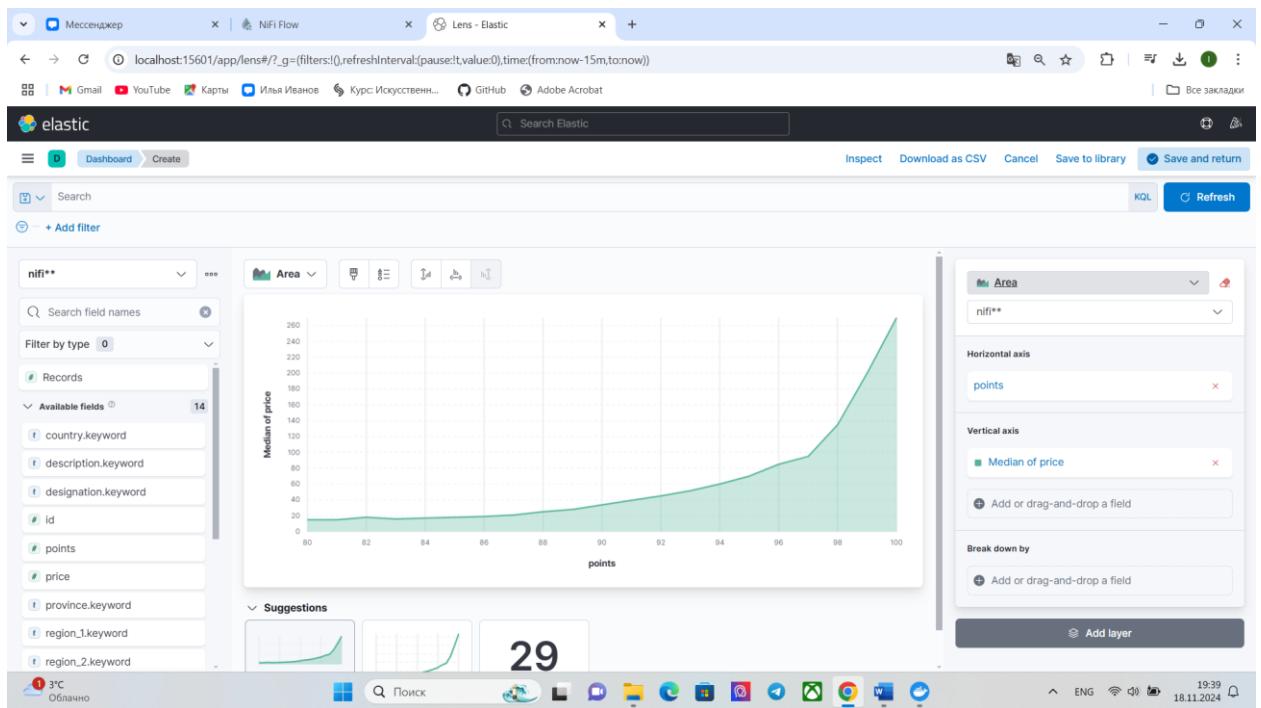


Рисунок 50 – Построенный график.

Шаблон DAG-а будет приложен в папку репозитория решения лабораторной работы №1.

Заключение

В ходе выполнения лабораторной работы получены навыки работы фреймворками Apache Airflow и Apache Nifi. Удалось построить пайпайн в Apache NiFi с помощью графического интерфейса, а в Apache Airflow с помощью кода Airflow.

