Отчёт 1 лабораторной работы по дисциплине “Инженерия данных”

выполнил Доружинский Дмитрий из группы 6233-010402D.

1. Подготовительный этап

Выполнил все рекомендации из репозитория https://github.com/ssau-data-engineering/Prerequisites

Системные требования следующие:

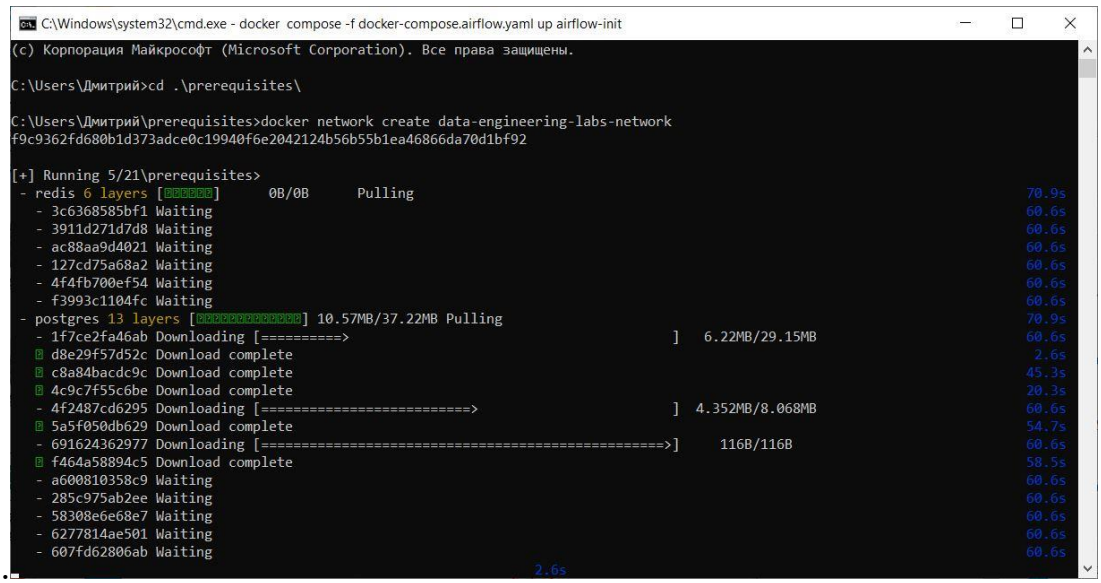
Система

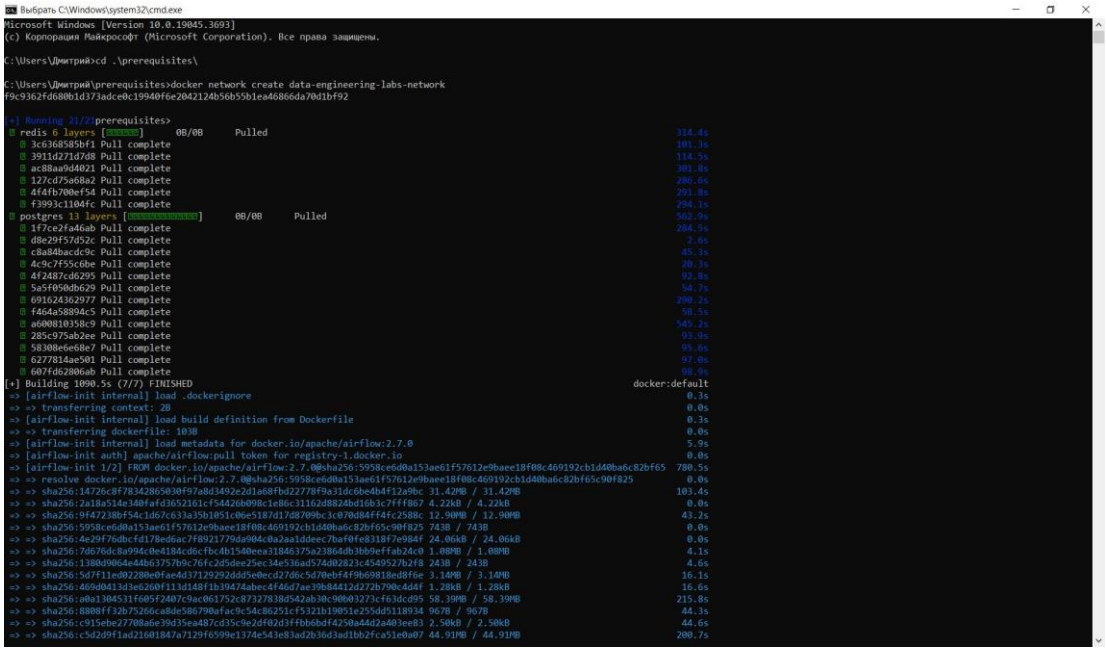
Процессор:	AMD Ryzen 5 4500U with Radeon Graphics	2.38 GHz
Установленная память (ОЗУ):	8,00 ГБ (7,40 ГБ доступно)	
Тип системы:	64-разрядная операционная система, процессор x64	
Перо и сенсорный ввод:	Перо и сенсорный ввод недоступны для этого экрана	

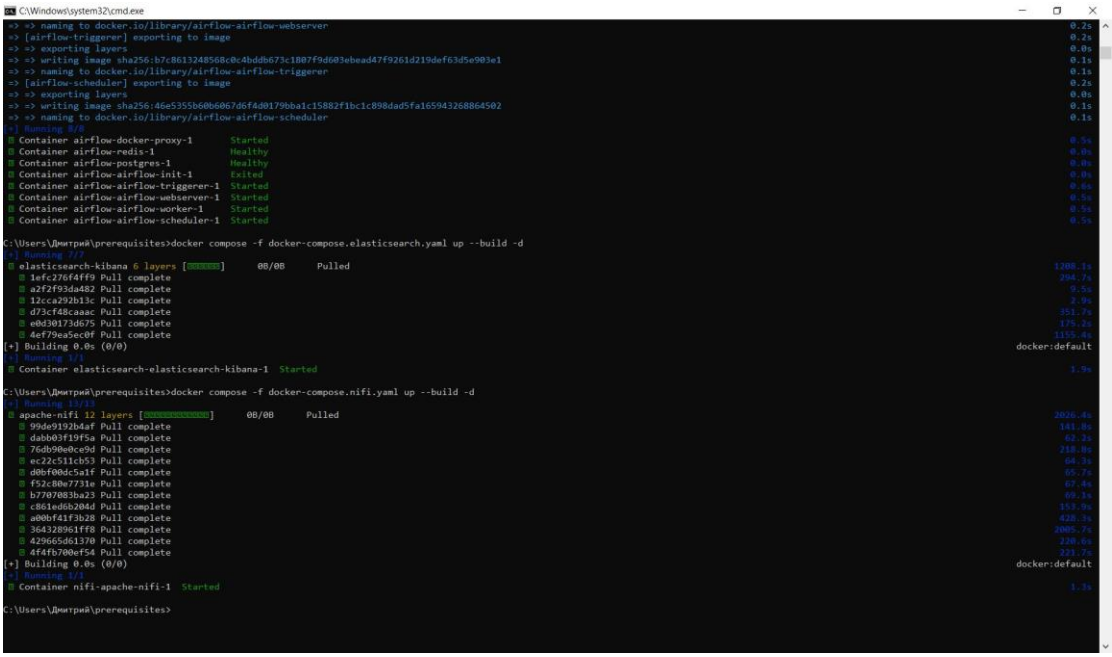
В ходе скачивания необходимых компонентов для выполнения лабораторной работы было выяснено, что нужно освободить более 40 гб свободной памяти, а также после выполнения команд:

- *docker compose -f docker-compose.airflow.yaml up --build -d*
- *docker compose -f docker-compose.nifi.yaml up --build -d*
- *docker compose -f docker-compose.elasticsearch.yaml up --build -d*

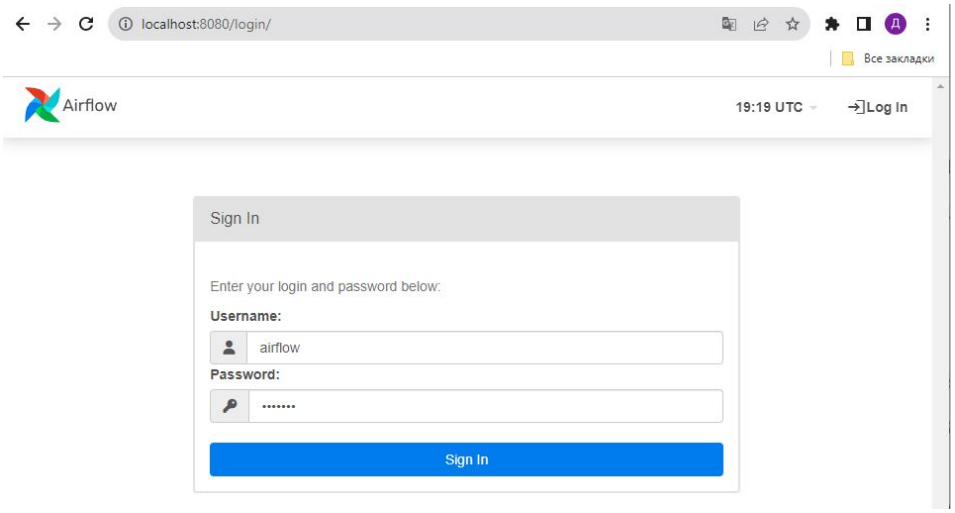
Необходимо вручную остановливать в Docker Desktop контейнеры иначе ресурсов моего компьютера не хватает и происходит зависание с посследующием обнулением Docker. Привожу скриншоты загрузки компонентов.Также хочу отметить, что время загрузки всех компонентов с мобильного интернета составило +- 2 часа.

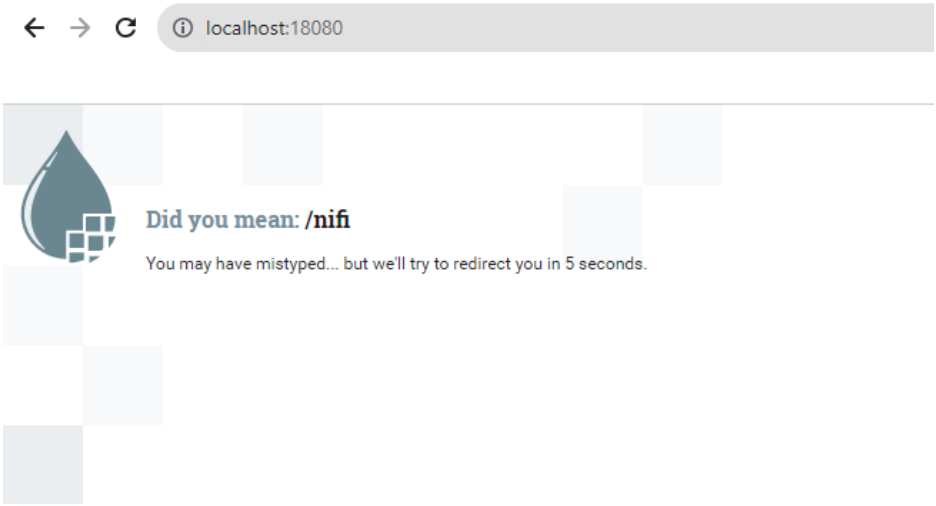

C:\Windows\system32\cmd.exe - docker compose -f docker-compose.airflow.yaml up airflow-init
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\Дмитрий>cd .\prerequisites\
C:\Users\Дмитрий\prerequisites>docker network create data-engineering-labs-network
f9c9362fd680b1d373adce0c19940f6e2042124b56b55b1ea46866da70d1bf92
[+] Running 5/21\prerequisites>
- redis 6 layers [] 0B/0B Pulling 70.9s
- 3c6368585bf1 Waiting 60.6s
- 3911d271d7d8 Waiting 60.6s
- ac88aa9d4021 Waiting 60.6s
- 127cd75a68a2 Waiting 60.6s
- 4f4fb700ef54 Waiting 60.6s
- f3993c1104fc Waiting 60.6s
- postgres 13 layers [] 10.57MB/37.22MB Pulling 70.9s
- 1f7ce2fa46ab Downloading [==========>] 6.22MB/29.15MB 60.6s
- d8e29f57d52c Download complete 2.6s
- c8a84bacdc9c Download complete 45.3s
- 4c9c7f55c6be Download complete 20.3s
- 4f2487cd6295 Downloading [==========>] 4.352MB/8.068MB 60.6s
- 5a5f050db629 Download complete 54.7s
- 691624362977 Downloading [==========>] 116B/116B 60.6s
- f464a58894c5 Download complete 58.5s
- a600810358c9 Waiting 60.6s
- 285c975ab2ee Waiting 60.6s
- 58308e6e68e7 Waiting 60.6s
- 6277814ae501 Waiting 60.6s
- 607fd62806ab Waiting 60.6s
2.6s

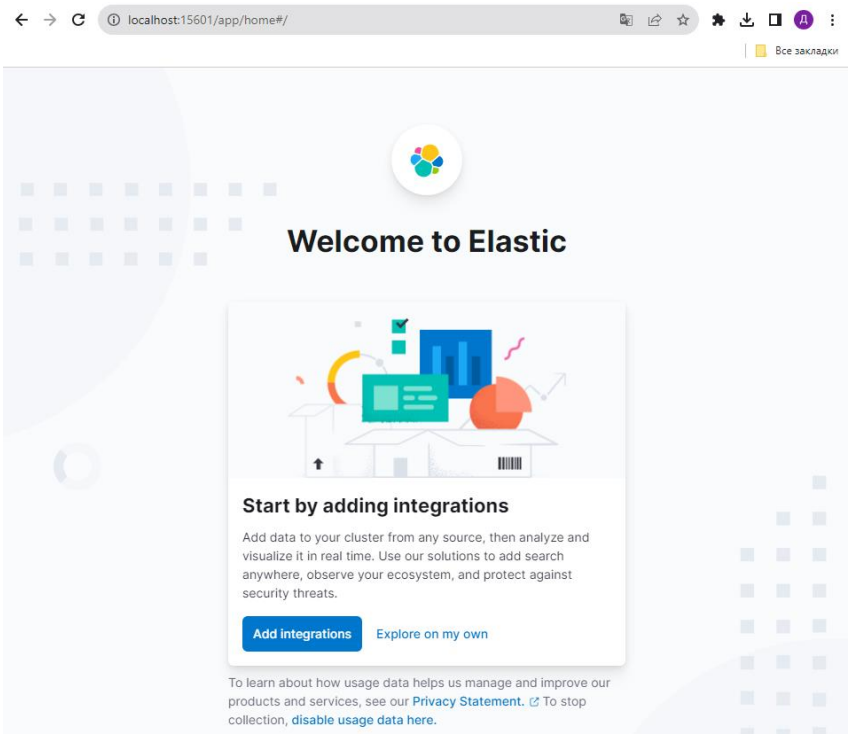

Выбрать C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3693]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\Дмитрий>cd .\prerequisites\
C:\Users\Дмитрий\prerequisites>docker network create data-engineering-labs-network
f9c9362fd680b1d373adce0c19940f6e2042124b56b55b1ea46866da70d1bf92
[+] Running 21/21prerequisites>
redis 6 layers [] 0B/0B Pulled 314.4s
3c6368585bf1 Pull complete 101.3s
3911d271d7d8 Pull complete 114.5s
ac88aa9d4021 Pull complete 301.8s
127cd75a68a2 Pull complete 286.6s
4f4fb700ef54 Pull complete 291.8s
f3993c1104fc Pull complete 294.1s
postgres 13 layers [] 0B/0B Pulled 562.9s
1f7ce2fa46ab Pull complete 284.5s
d8e29f57d52c Pull complete 2.6s
c8a84bacdc9c Pull complete 45.3s
4c9c7f55c6be Pull complete 20.3s
4f2487cd6295 Pull complete 92.8s
5a5f050db629 Pull complete 54.7s
691624362977 Pull complete 290.2s
f464a58894c5 Pull complete 58.5s
a600810358c9 Pull complete 545.2s
285c975ab2ee Pull complete 93.9s
58308e6e68e7 Pull complete 95.6s
6277814ae501 Pull complete 97.0s
607fd62806ab Pull complete 98.9s
[+] Building 1090.5s (7/7) FINISHED docker:default
=> [airflow-init internal] load .dockerignore 0.3s
=> => transferring context: 2B 0.0s
=> [airflow-init internal] load build definition from Dockerfile 0.3s
=> => transferring dockerfile: 103B 0.0s
=> [airflow-init internal] load metadata for docker.io/apache/airflow:2.7.0 5.9s
=> [airflow-init auth] apache/airflow:pull token for registry-1.docker.io 0.0s
=> [airflow-init 1/2] FROM docker.io/apache/airflow:2.7.0@sha256:5958ce6d0a153ae61f57612e9baee18f08c469192cb1d40ba6c82bf65 780.5s
=> => resolve docker.io/apache/airflow:2.7.0@sha256:5958ce6d0a153ae61f57612e9baee18f08c469192cb1d40ba6c82bf65c90f825 0.0s
=> => sha256:14726c8f783428650308f97a8d3492e2d1a68fbd22778f9a31dc6be4b4f12a9bc 31.42MB / 31.42MB 103.4s
=> => sha256:2a18a514e340fafd3652161cf54426b098c1e86c31162d8824bd16b3c7fff867 4.22kB / 4.22kB 0.0s
=> => sha256:9f47238bf54c1d67c633a35b1051c00e5187d17d8709bc3c070d84ff4fc2588c 12.90MB / 12.90MB 43.2s
=> => sha256:5958ce6d0a153ae61f57612e9baee18f08c469192cb1d40ba6c82bf65c90f825 743B / 743B 0.0s
=> => sha256:4e29f76dbcfd178ed6ac7f8921779da904c0a2aa1ddeec7baf0fe8318f7e984f 24.06kB / 24.06kB 0.0s
=> => sha256:7d676dc8a994c0e4184cd6cfbc8b1540eea31846375a2386d4b3bb9eff4ab24c0 1.08MB / 1.08MB 4.1s
=> => sha256:1380d9064e44bb3757b9c76fc2d5dee25ec34e536ad57d0282 3cc4549527b2f8 243B / 243B 4.6s
=> => sha256:5d7f11ed00280e0fae4d3712929ddd5e0ecd27d6c5d70ebf4f9b69818ed8f6e 3.14MB / 3.14MB 16.1s
=> => sha256:469d0413d3e6260f113d148f1b19474abec4f46d7ae39b84412d272b790c4d4f 1.28kB / 1.28kB 16.6s
=> => sha256:a0a1304531f605f2407c9ac061752c87327838d542ab30c90b03273cf63dcd95 58.39MB / 58.39MB 215.8s
=> => sha256:8808ff32b75266ca8de586790afac9c54c86251cf5321b19051e255dd5118934 967B / 967B 44.3s
=> => sha256:c915ebe27708a6e39d35ea487cd35c9e2df02d3ffbb6bdf4250a44d2a403ee83 2.50kB / 2.50kB 44.6s
=> => sha256:c5d2d9f1ad21601847a7129f6599e1374e543e83ad2b36d3ad1bb2fca51e0a07 44.91MB / 44.91MB 200.7s


C:\Windows\system32\cmd.exe
=> => naming to docker.io/library/airflow-airflow-webserver 0.2s
=> [airflow-triggerer] exporting to image 0.2s
=> => exporting layers 0.0s
=> => writing image sha256:b7c861324856 8c0c4bddb673c1807f9d603ebead47f9261d219def63d5e903e1 0.1s
=> => naming to docker.io/library/airflow-airflow-triggerer 0.1s
=> [airflow-scheduler] exporting to image 0.2s
=> => exporting layers 0.0s
=> => writing image sha256:46e5355b60b6067d6f4d0179ba1c15882f1bc1c898dad5fa165943268864502 0.1s
=> => naming to docker.io/library/airflow-airflow-scheduler 0.1s
[+] Running 8/8
Container airflow-docker-proxy-1 Started 0.5s
Container airflow-redis-1 Healthy 0.0s
Container airflow-postgres-1 Healthy 0.0s
Container airflow-airflow-init-1 Exited 0.0s
Container airflow-airflow-triggerer-1 Started 0.6s
Container airflow-airflow-webserver-1 Started 0.5s
Container airflow-airflow-worker-1 Started 0.5s
Container airflow-airflow-scheduler-1 Started 0.5s
C:\Users\Дмитрий\prerequisites>docker compose -f docker-compose.elasticsearch.yaml up --build -d
[+] Running 7/7
elastic search-kibana 6 layers [] 0B/0B Pulled 1208.1s
1efc276f4ff9 Pull complete 294.7s
a2f2f93da482 Pull complete 9.5s
12cca292b13c Pull complete 2.9s
d73cf48caaac Pull complete 351.7s
e0d30173d675 Pull complete 175.2s
4ef79ea5ec0f Pull complete 183.4s
[+] Building 0.0s (0/0) docker:default
[+] Running 1/1
Container elasticsearch-elasticsearch-kibana-1 Started 1.9s
C:\Users\Дмитрий\prerequisites>docker compose -f docker-compose.nifi.yaml up --build -d
[+] Running 1/21
apache-nifi 12 layers [] 0B/0B Pulled 2026.4s
99de9192b4af Pull complete 141.0s
dabb03f19f5a Pull complete 62.2s
76db90e0ce9d Pull complete 218.8s
ec22c511cb53 Pull complete 64.3s
d0bf006dc5a1f Pull complete 65.7s
f52c80e7731e Pull complete 67.4s
b7707083ba23 Pull complete 68.1s
c861ed6b204d Pull complete 153.9s
a00bf41f3b28 Pull complete 428.3s
364328961ff8 Pull complete 2005.7s
429665d61170 Pull complete 220.6s
4f4fb700ef54 Pull complete 221.7s
[+] Building 0.0s (0/0) docker:default
[+] Running 1/1
Container nifi-apache-nifi-1 Started 1.3s
C:\Users\Дмитрий\prerequisites>

После проверил корректно ли запускаются контейнеры с NiFi, Airflow и Elastic.





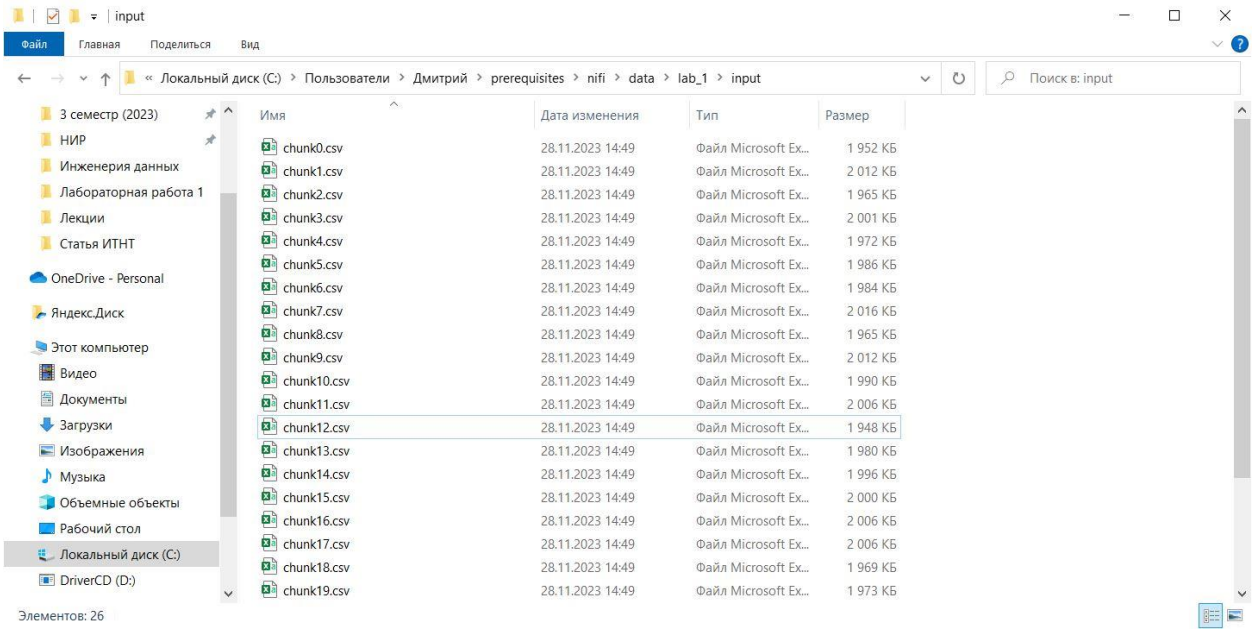


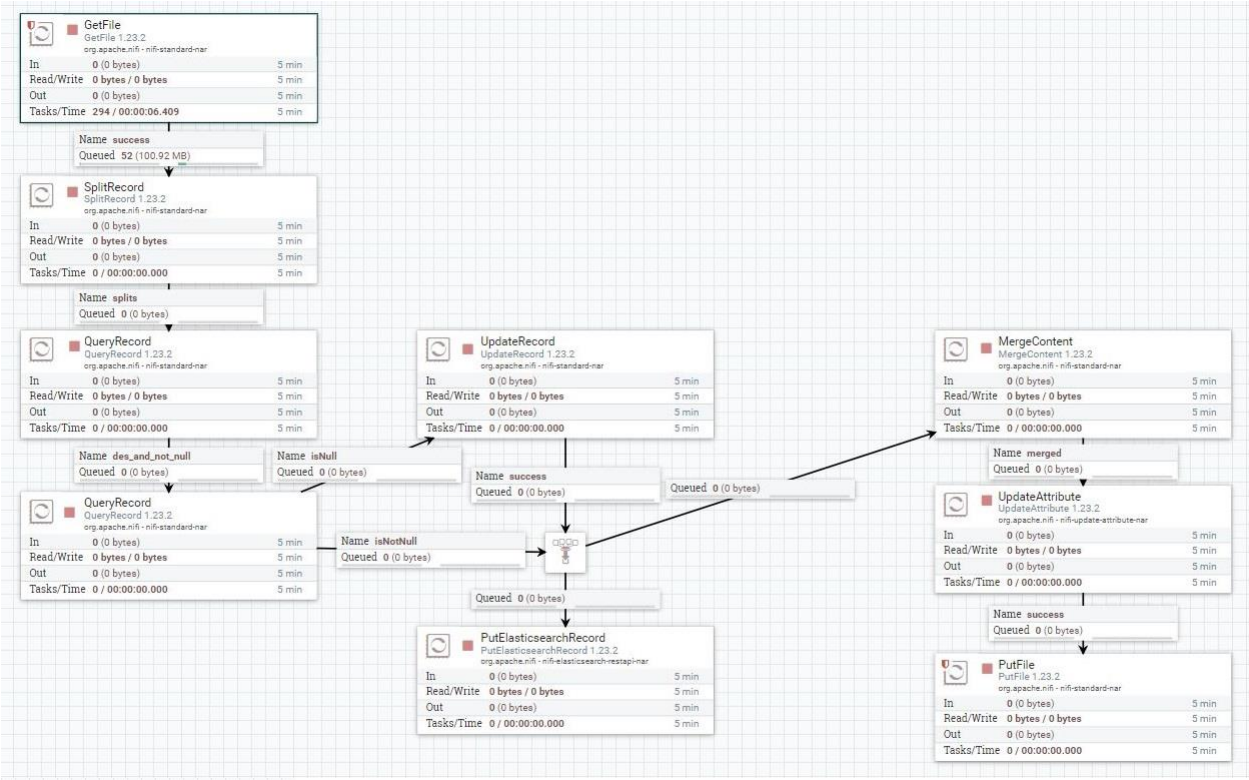
2. Построение пайплайна в NiFi

Для построения пайплайна в NiFi использовал рекомендуемые процессы, а именно:

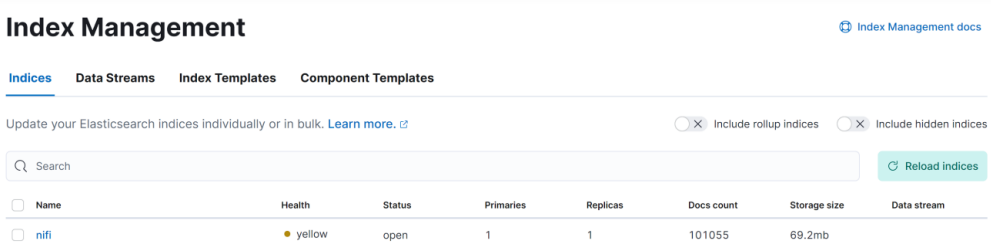
- GetFile – для считывания файлов
- SplitRecord – для разделения данных на батчи
- QueryRecord – необходима для проверки в данных полей designation и region_1, чтобы они не были пустыми. После нужна для разделения строк с price == null и price != null
- UpdateRecord – изменяет имя получившегося файла, без этого файл будет с названием исходного файла.
- MergeContent – предназначен для объединения всех файлов в один
- PutFile – сохраняет файл в директорию где находятся исходные файлы
- PutElasticsearchHttp – нужна для переброса данных в ElasticSearch для построения графиков

Перед работой перенес предложенный набор данных в папку *nifi/data/lab_1/input*

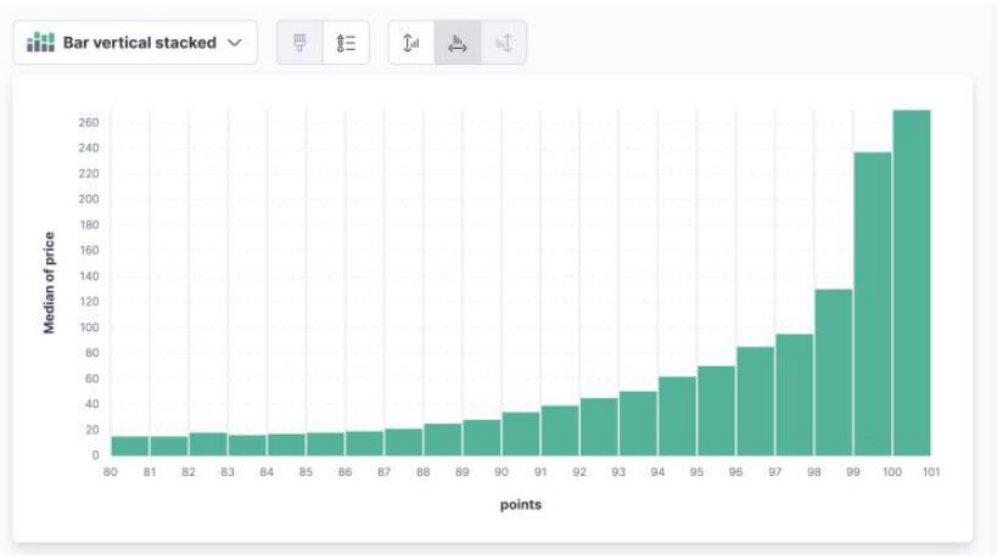




После убеждаемся, что данные пришли в ElasticSearch, для этого переходим в Index Management



Далее создаём паттерн и строим график



3. Построение пайплайна в Airflow

Построение пайплайна в Airflow необходимо написать код на Python, использовал для этого VS Code со следующими расширениями:

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

Был написан следующий код для обработки предложенного датасета

```
C: > Users > Дмитрий > Desktop > Учеба > 3 семестр (2023) > Инженерия данных > 1 лаба > lab1.py > ...
from airflow import DAG
from airflow.operators.python import PythonOperator
from elasticsearch import Elasticsearch
from datetime import datetime
import pandas as pd
import numpy as np

with DAG('basic_etl_dag',

         schedule_interval=None,

         start_date=datetime(2023, 28, 11),

         catchup=False) as dag:

    def read_data():
        result = pd.DataFrame()
        for i in range(26):
            result = pd.concat([result, pd.read_csv(f"/opt/airflow/data/chunk{i}.csv")])
        return result

    def transform_data():

        result = read_data()
        result = result[(result['designation'].str.len() > 0)]
        result = result[(result['region_1'].str.len() > 0)]
        result['price'] = result['price'].replace(np.nan, 0)
        result = result.drop(['id'], axis=1)
        result.to_csv('/opt/airflow/data/data.csv', index=False)

    def load_to_elastic():
     patch = Elasticsearch("http://elasticsearch-kibana:9200")
     data = pd.read_csv(f"/opt/airflow/data/data.csv")

     for i, row in data.iterrows():
        doc = {
        "country": row["country"],
        "description": row["description"],
        "designation": row["designation"],
        "points": row["points"],
        "price": row["price"],
        "province": row["province"],
        "region_1": row["region_1"],
        "region_2": row["region_1"],
        "taster_name": row["taster_name"],
        "taster_twitter_handle": row["taster_twitter_handle"],
        "title": row["title"],
        "variety": row["variety"],
        "winery": row["winery"],
     }

     if i < data.shape[0] - 1:
        patch.index(index="wines", id=i, document=doc)

    transform_data = PythonOperator(

         task_id='transform_data',

         python_callable=transform_data,

         dag=dag)

    load_to_elastic = PythonOperator(

         task_id='load_to_elastic',

         python_callable=load_to_elastic,

         dag=dag)

     #task order
    transform_data >> load_to_elastic
```

4. Вывод

В ходе проделанной работы были получены практические навыки работы с такими фреймворками как Apache Airflow и Nifi. Удалось построить пайплайн в NiFi с помощью графического интерфейса, а в Airflow с помощью кода Airflow. Так же получен полезный опыт работы с контейнерами в Docket Desktop