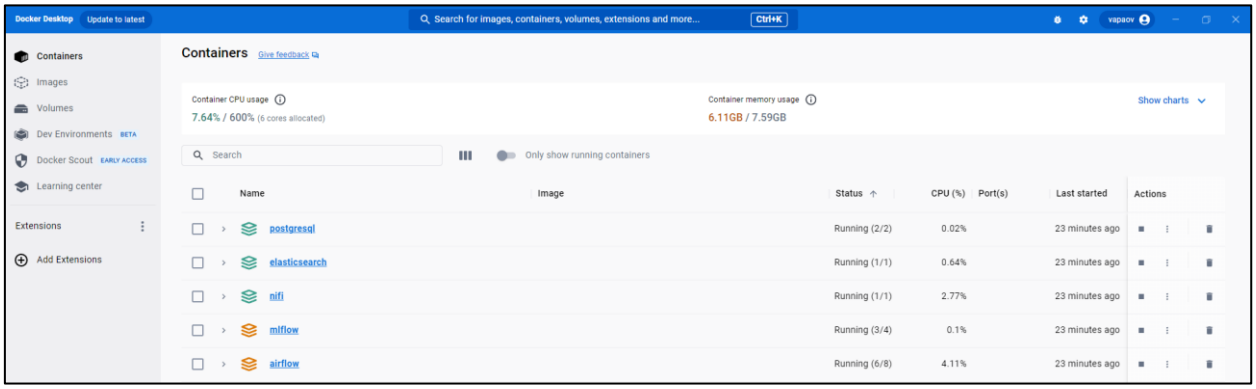
БУЛДАКОВ ДМИТРИЙ

6233-010402D

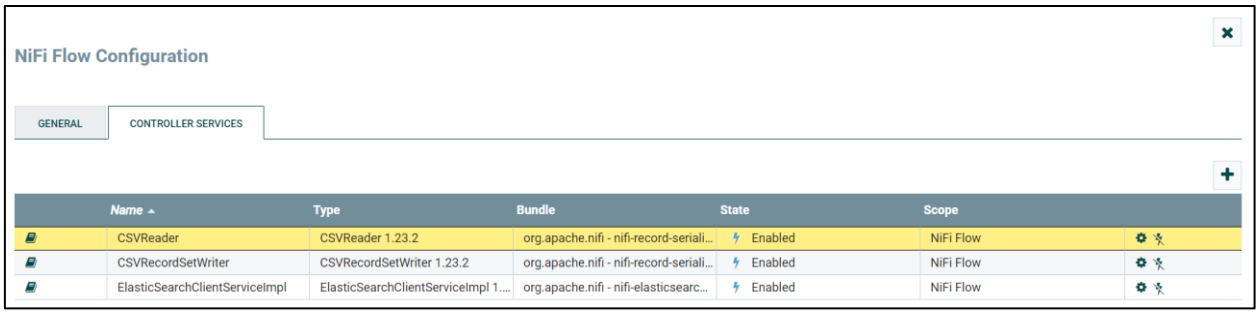
ХОД РАБОТЫ

По инструкции из репозитория Prerequisites были подняты необходимые контейнеры.

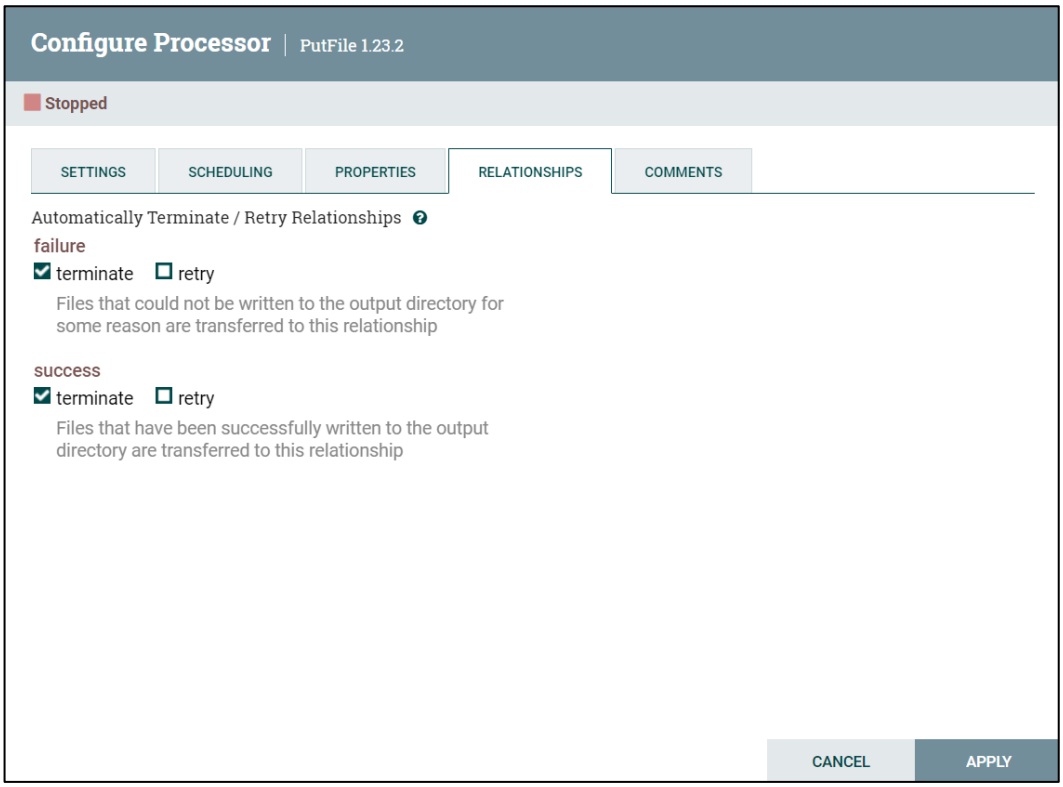


APACHE NIFI

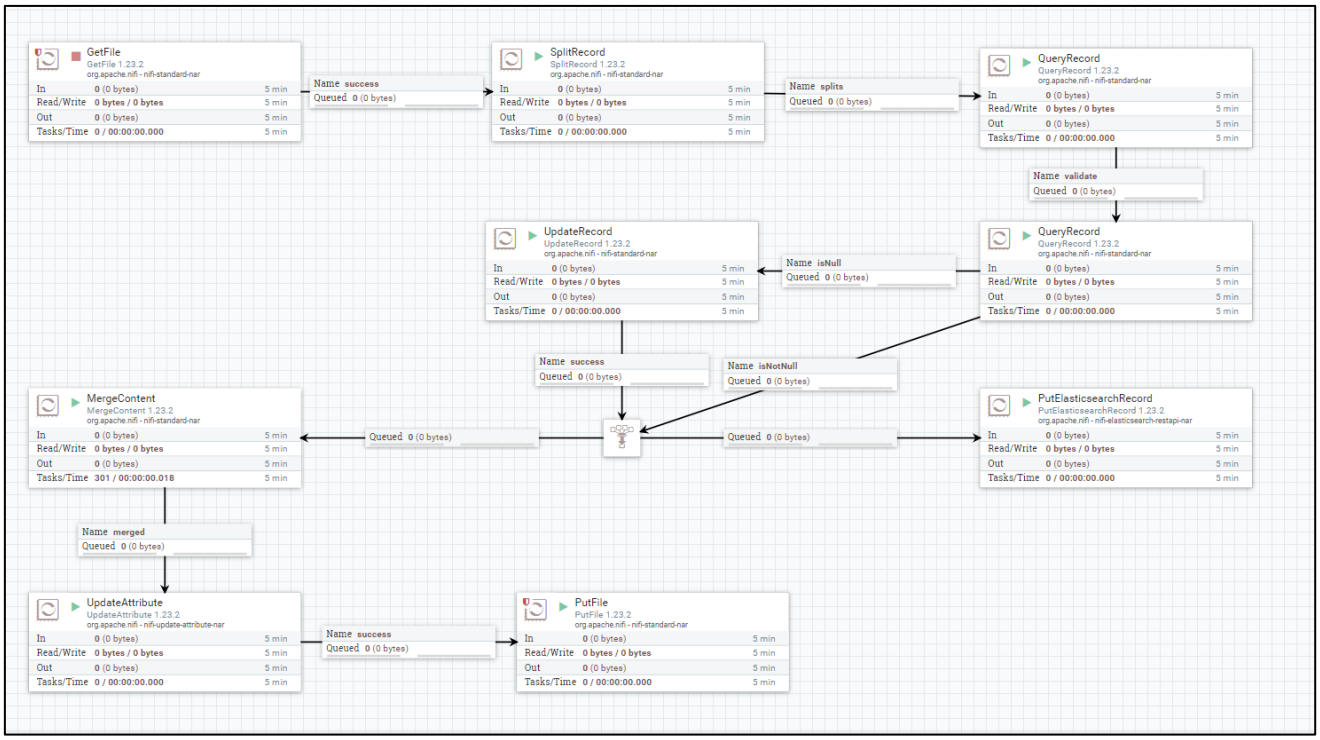
В Apache Nifi реализовали пайплайн. Использовались следующие процессоры: GetFile, SplitRecord, QueryRecord, UpdateRecord, MergeContent, PutFile и PutElasticsearchHttp. В некоторых процессорах использовались вручную добавленные сервисы CSVReader, CSVRecordSetWriter и ElasticSearchClient. Необходимо было включить перед первым запуском пайплайна.



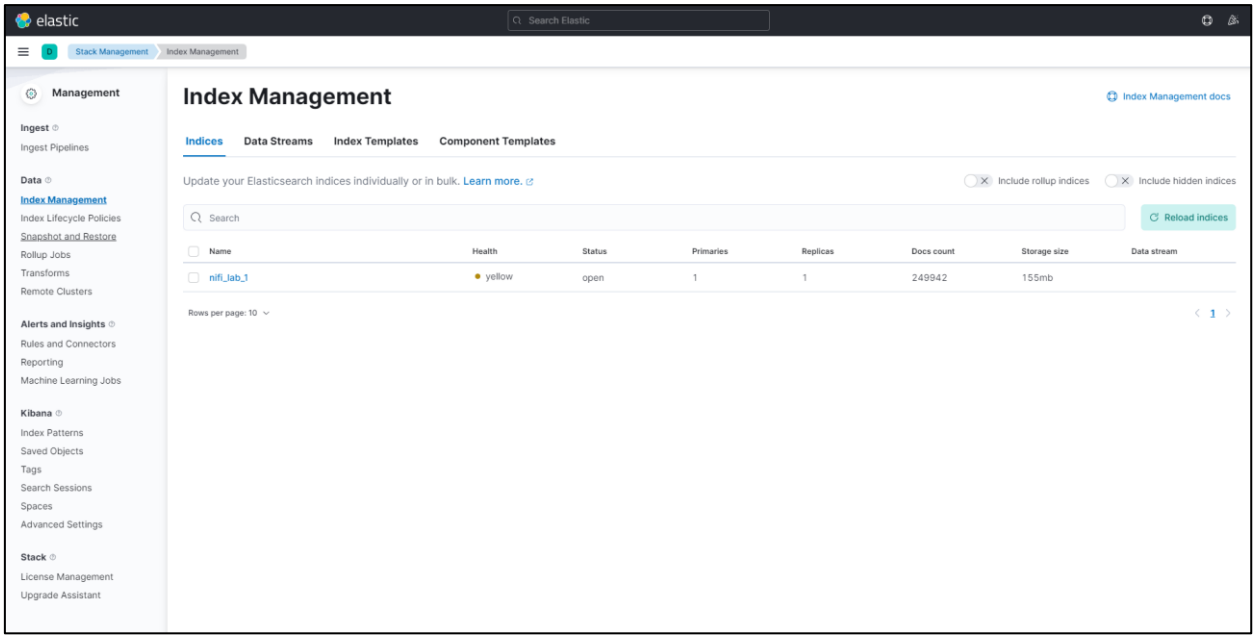
Так же были проблемы с ошибками связанными с стрелками отношений. Они решались указанием поведения при успешной\ неуспешной передачи данных.



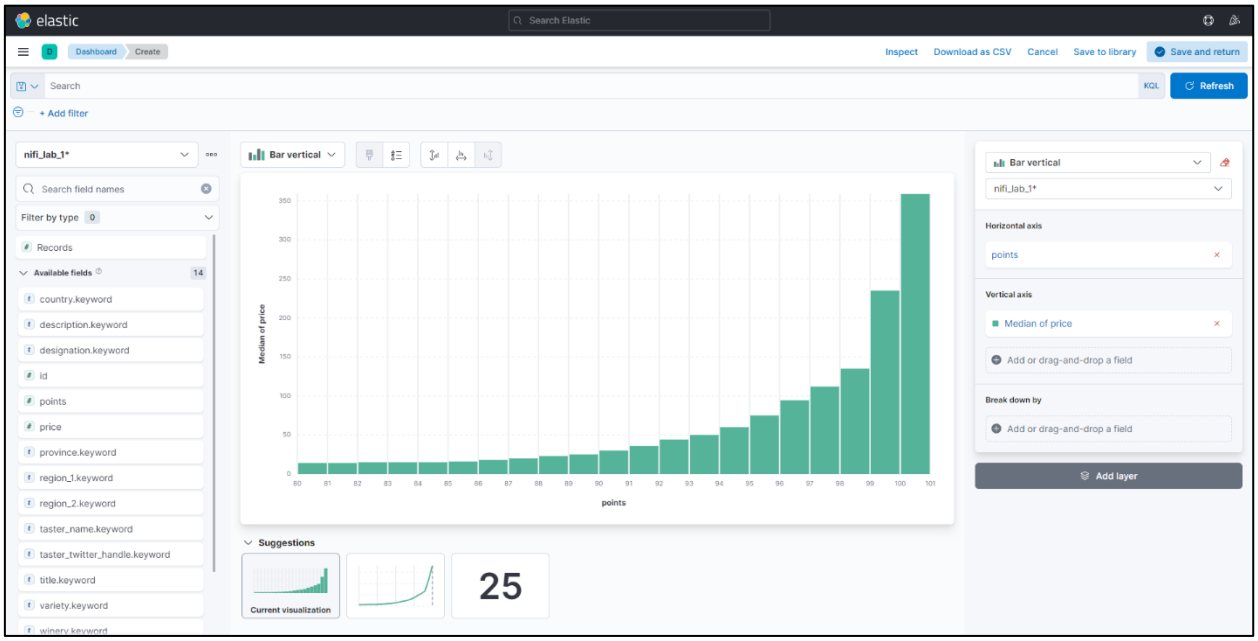
Что касается других настроек – особых проблем не возникло. Итоговая схема:



После полного прохода пайплайна в ElasticSearch и директорию (указанную в PutFile) отправился наш объединенный файлик.



В качестве теста в ElasticSearch получилось создать простенькую гистограмму



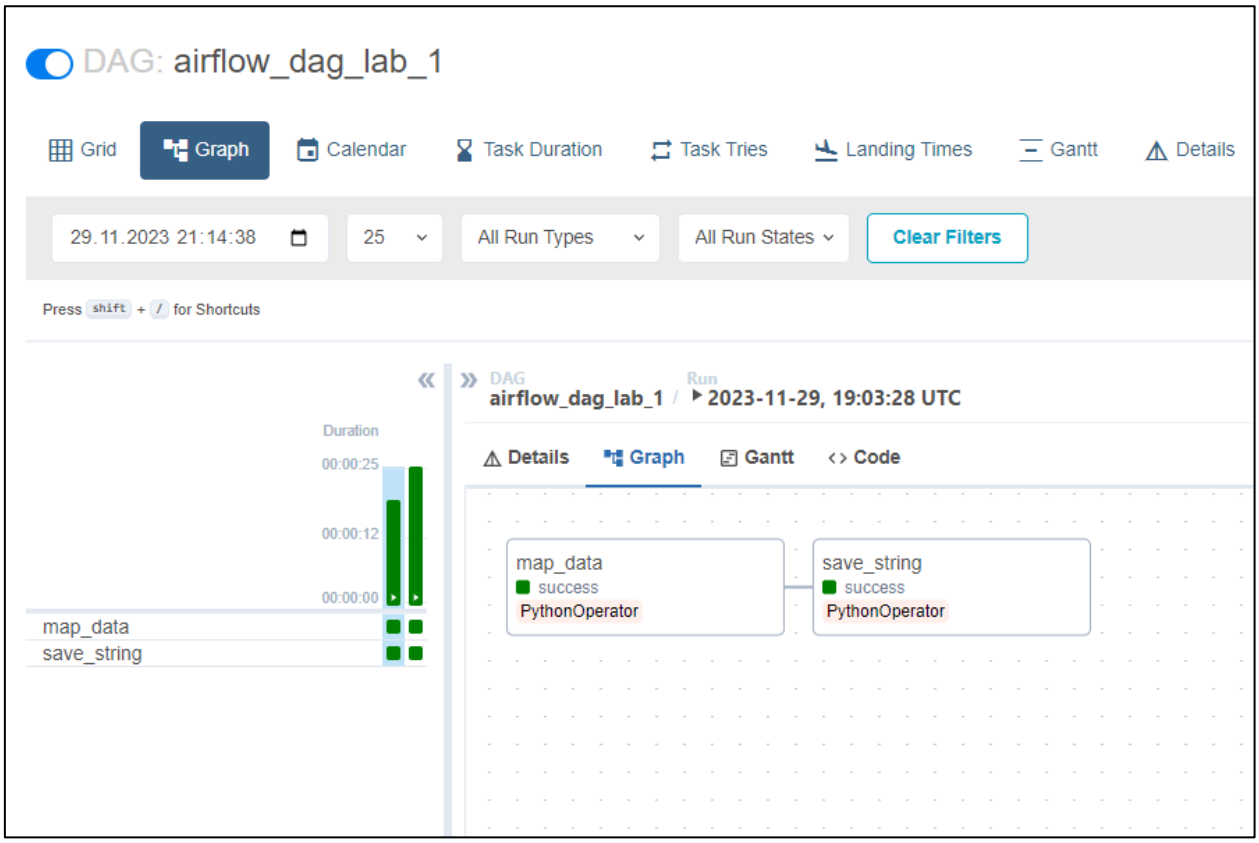
APACHE AIRFLOW

Для Apache Airflow был сформирован следующий DAG файл:

```
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('airflow_dag_lab_1',

         start_date=datetime(2023, 11, 28),

         schedule_interval=None,

         catchup=False) as dag:

    def map_data():

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

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

```
def save_string():
    path = Elasticsearch("http://elasticsearch-kibana:9200")
    data = pd.read_csv("/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_2"],
            "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:
        path.index(index="wines", id=i, document=doc)

map_data = PythonOperator(

    task_id='map_data',
    python_callable=map_data,
    dag=dag)

save_string = PythonOperator(

    task_id='save_string',
    python_callable=save_string,
    dag=dag)

map_data >> save_string
```

Файл .py был помещён в директорию \prerequisites\airflow\dags, затем пойман и успешно запущен в Airflow


DAG: airflow_dag_lab_1
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
29.11.2023 21:14:38
25
All Run Types
All Run States
Clear Filters
Press shift + / for Shortcuts
Duration
00:00:25
00:00:12
00:00:00
map_data
save_string
DAG
airflow_dag_lab_1
Run
2023-11-29, 19:03:28 UTC
Details
Graph
Gantt
Code
map_data
success
PythonOperator
save_string
success
PythonOperator