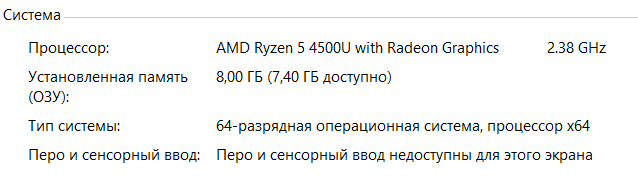
Отчёт 1 лабораторной работы по дисциплине “Инженерия данных” *выполнил Доружинский Дмитрий из группы 6233-010402D.*

# Подготовительный этап

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

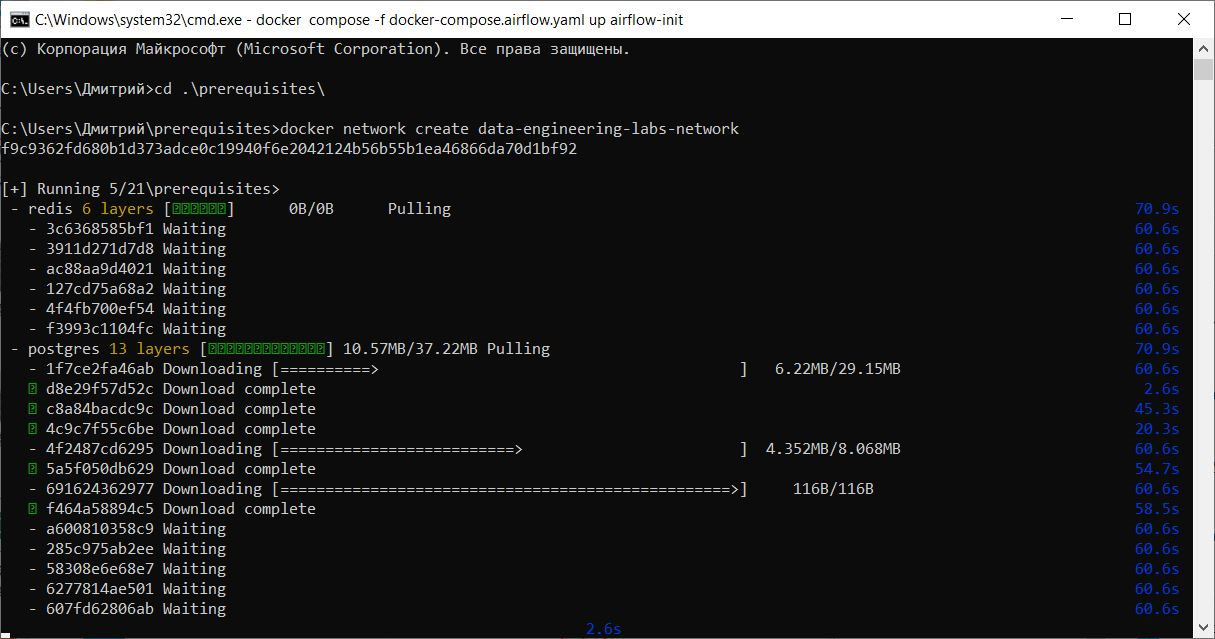
Системные требования следующие: 

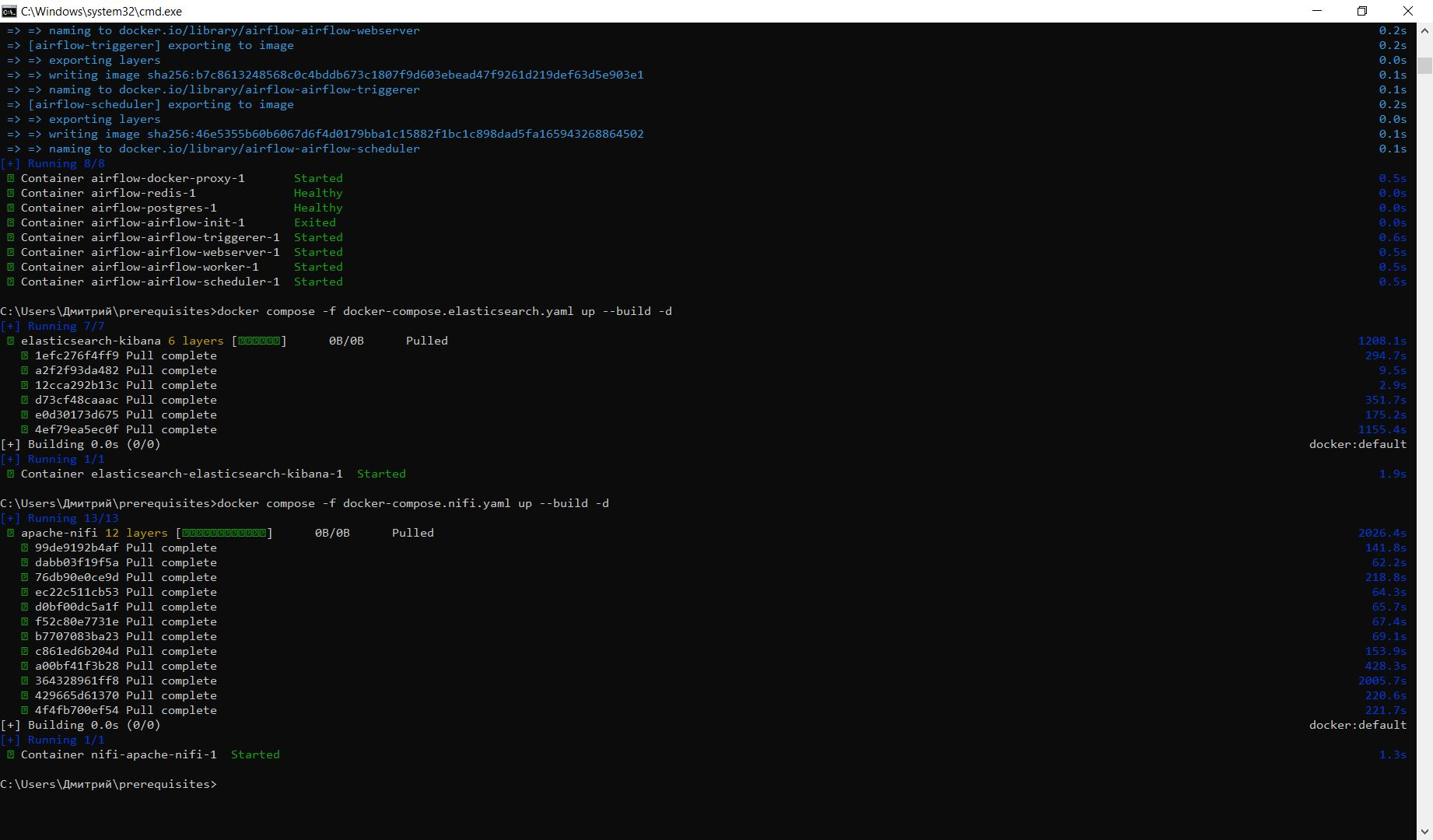
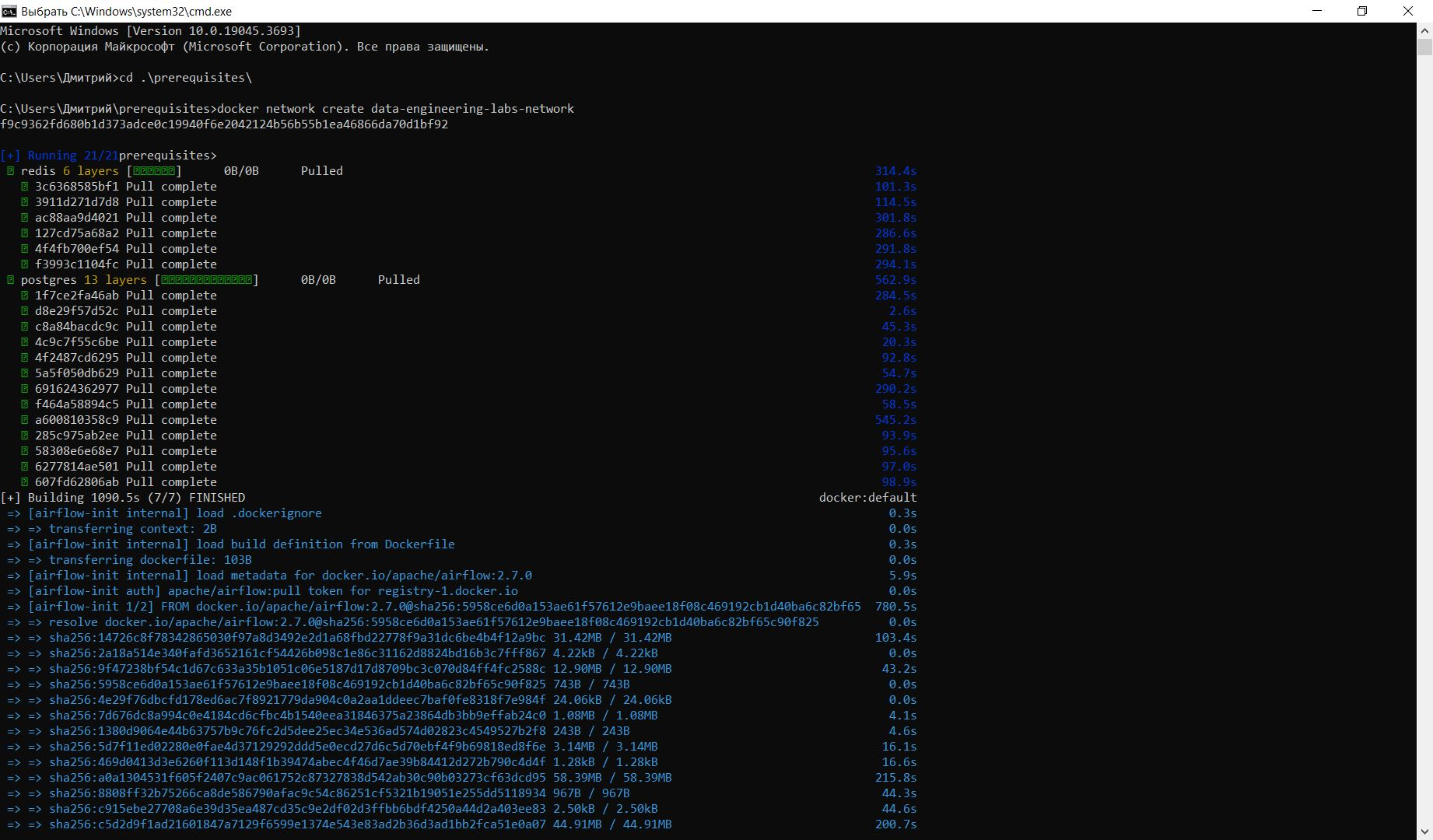
В ходе скачивания необходимых компонентов для выполнения лабораторной работы было выяснено, что нужно освободить более 40 гб свободной памяти, а также после выполнения команд:  
*- docker compose -f docker-compose.airflow.yaml up --build -d*

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

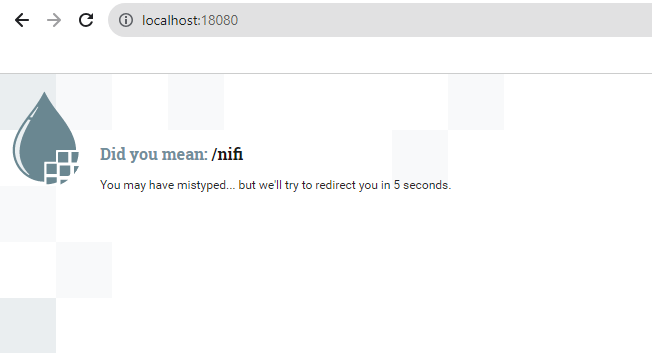
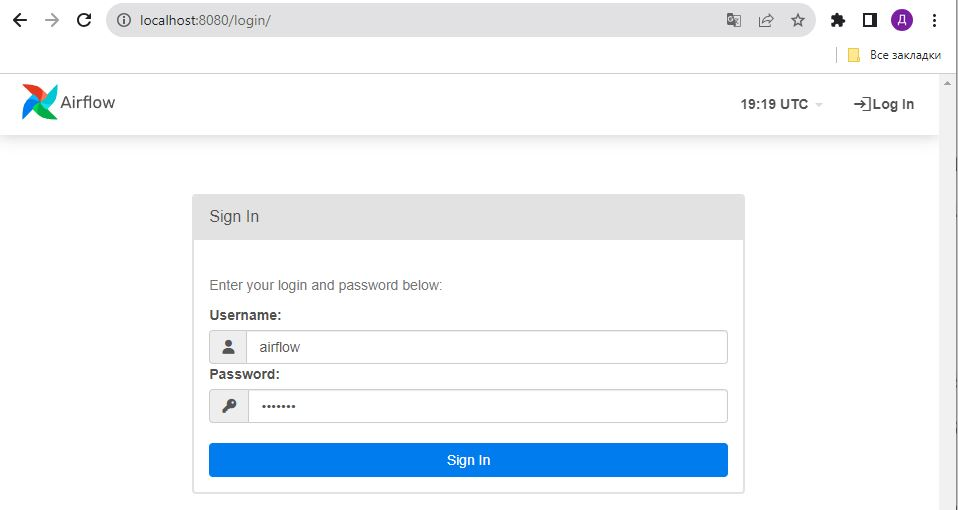
*- docker compose -f docker-compose.elasticsearch.yaml up --build -d*

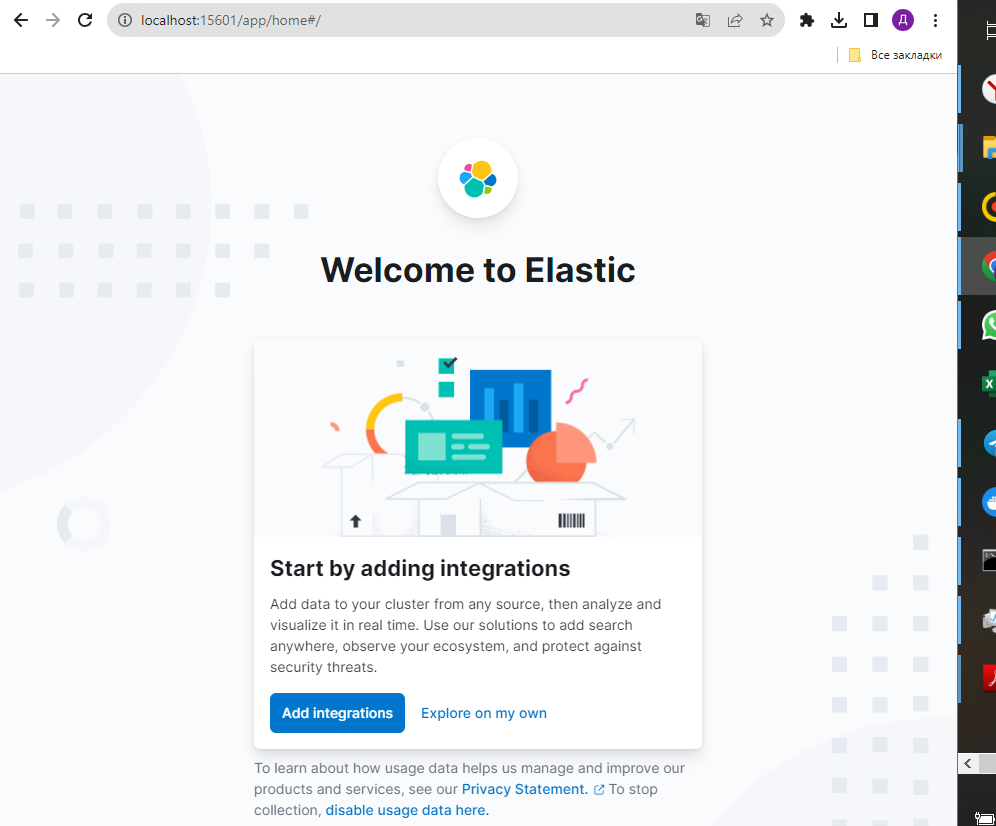
Необходимо вручную остановливать в Docker Desktop контейнеры иначе ресурсов моего компьютера не хватает и происходит зависание с посследующием обнулением Docker. Привожу скриншоты загрузки компонентов.Также хочу отметить, что время загрузки всех компонентов с мобильного интернета составило +- 2 часа.

.



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



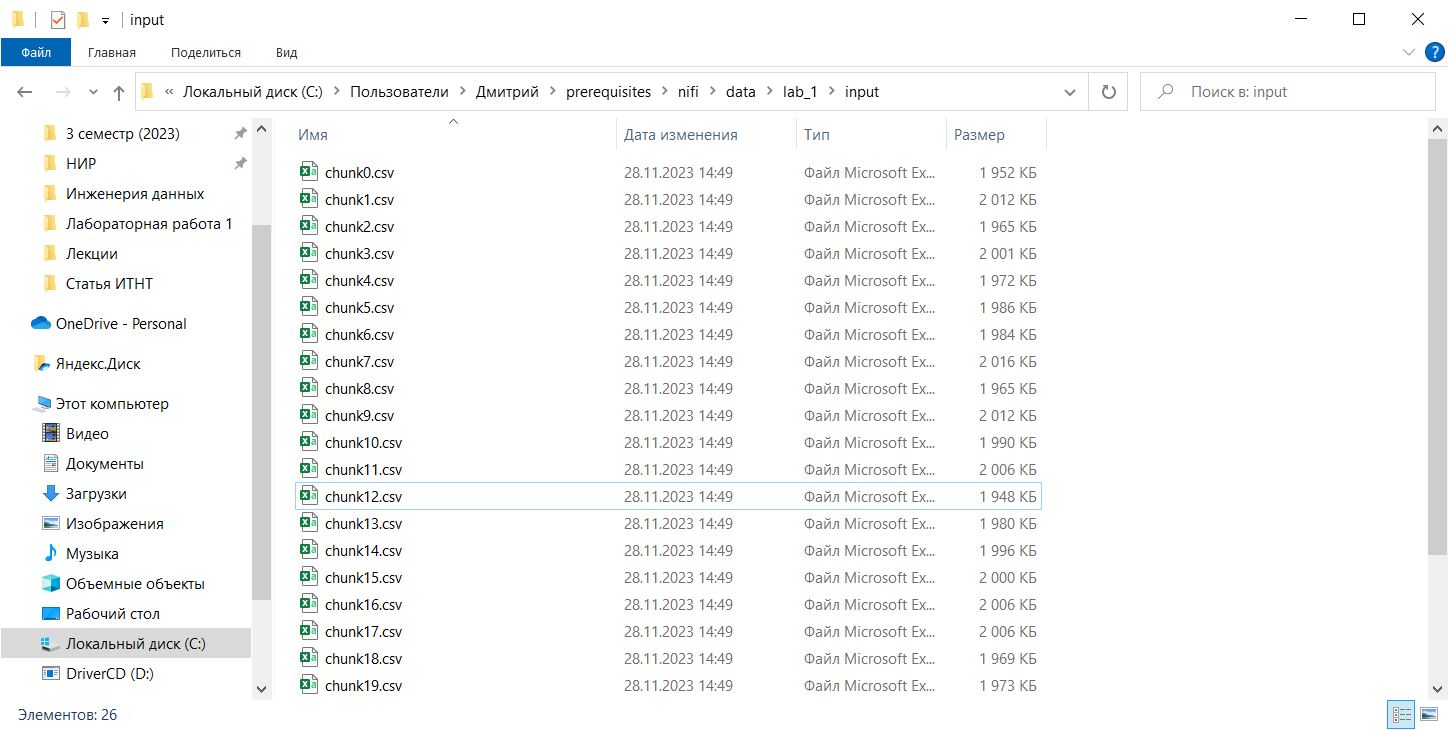


# Построение пайплайна в NiFi

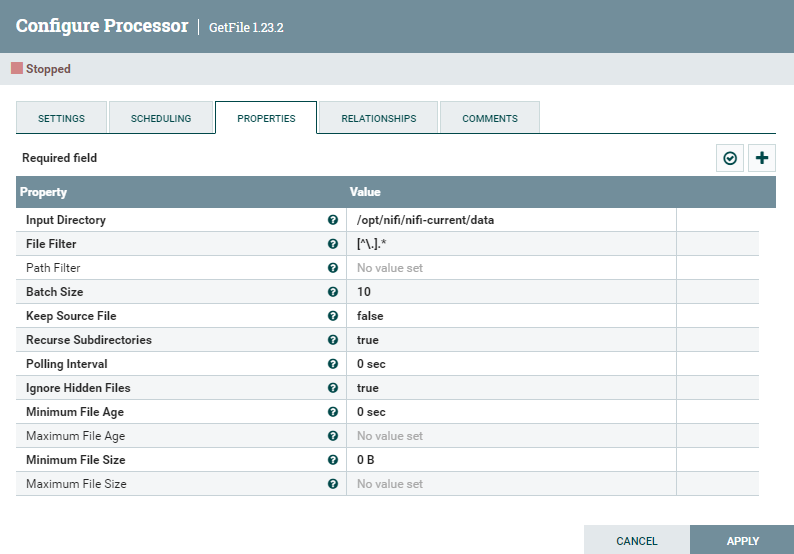
Для построения пайплайна в NiFi использовал рекомендуемые процессы, а именно:

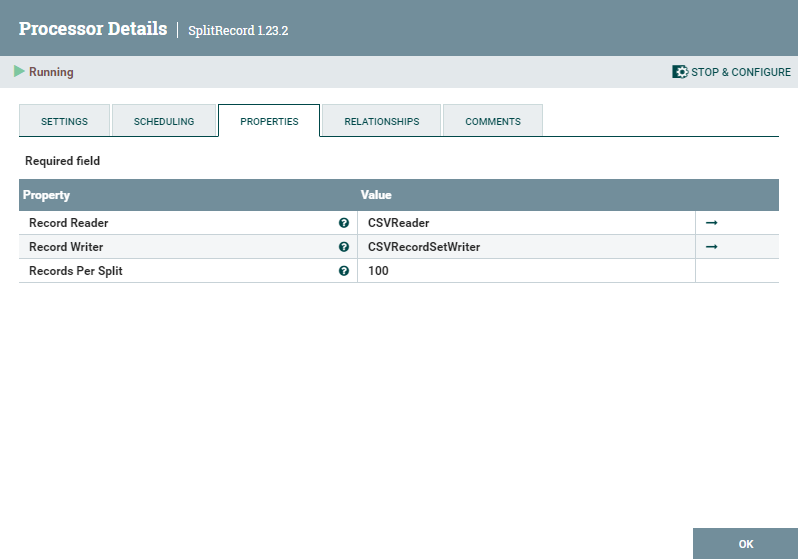
* GetFile – для считывания файлов
* SplitRecord – для разделения данных на батчи
* QueryRecord – необходима для проверки в данных полей designation и region\_1, чтобы они не были пустыми. После нужна для разделения строк с price == null и price != null
* UpdateRecord – изменяет имя получившегося файла, без этого файл будет с названием исходного файла.
* MergeContent – предназначен для объединения всех файлов в один
* UpdateAttribure – необходим для переименование получившегося файла
* PutFile – сохраняет файл в директорию где находятся исходные файлы
* PutElasticsearchHttp – нужна для переброса данных в ElasticSearch для построения графиков

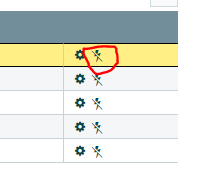
Перед работой перенес предложенный набор данных в папку *nifi/data/lab\_1/input*

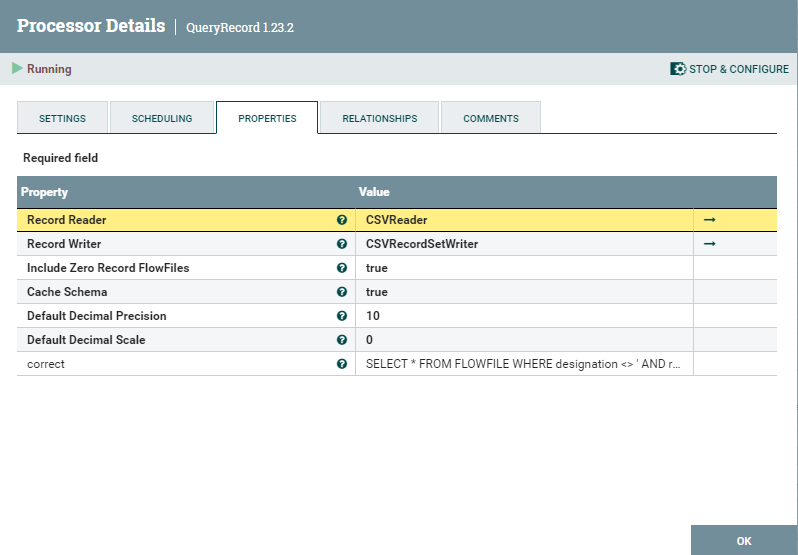


Ниже привожу настройку каждого процесса

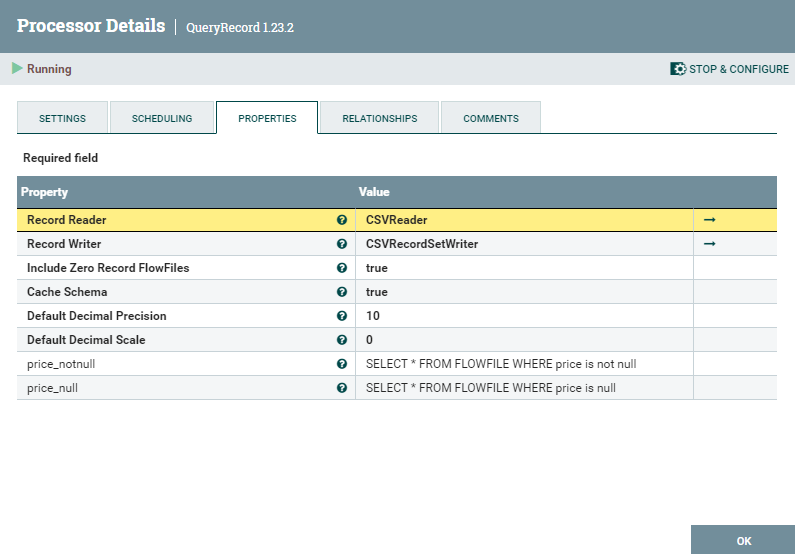


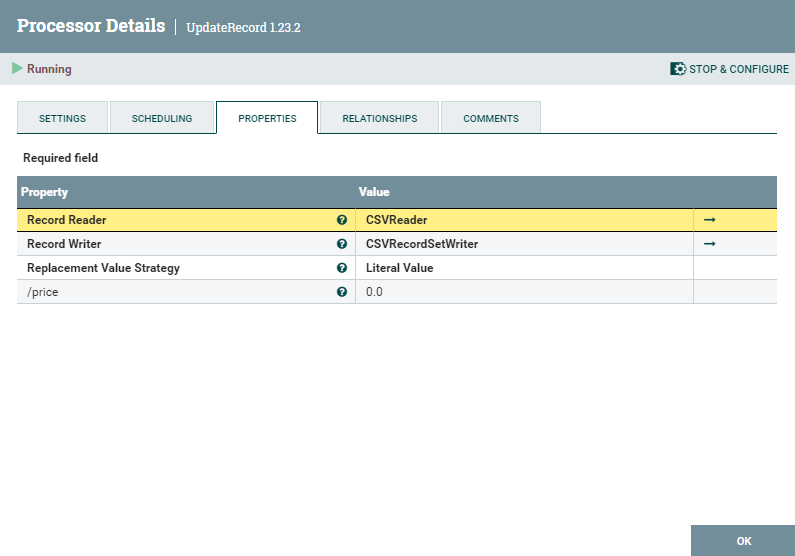


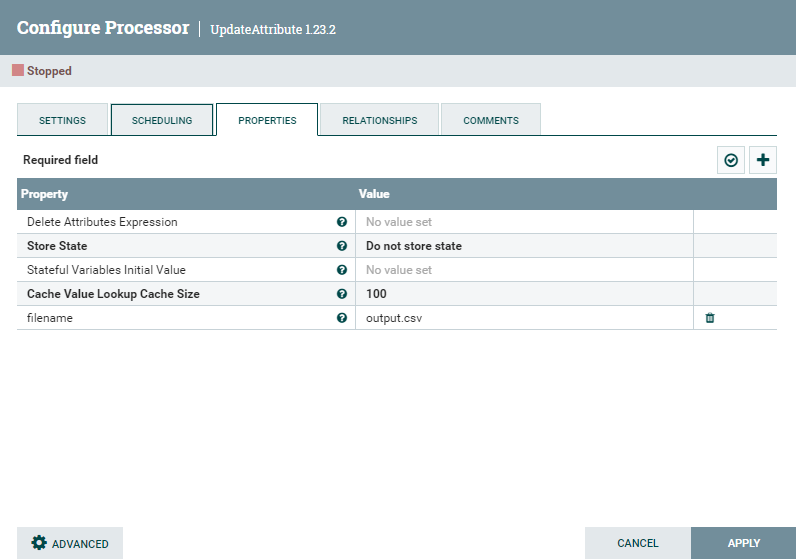
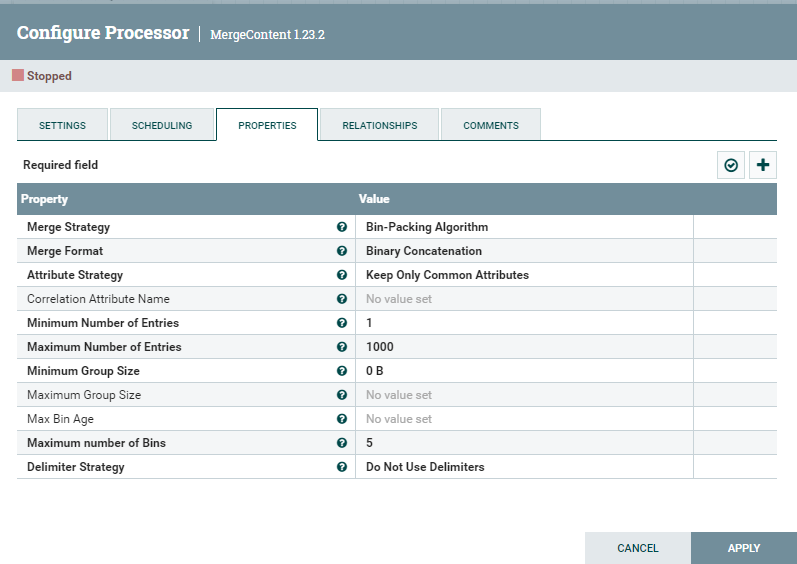
В настройке SplitRecord стоит упомянуть, что после выбора параметров CSVReader и CSVRecordSetWriter необходимо нажать на стрелку и после в появившемся окне нажать на молнию, активировав тем самым Enable. 

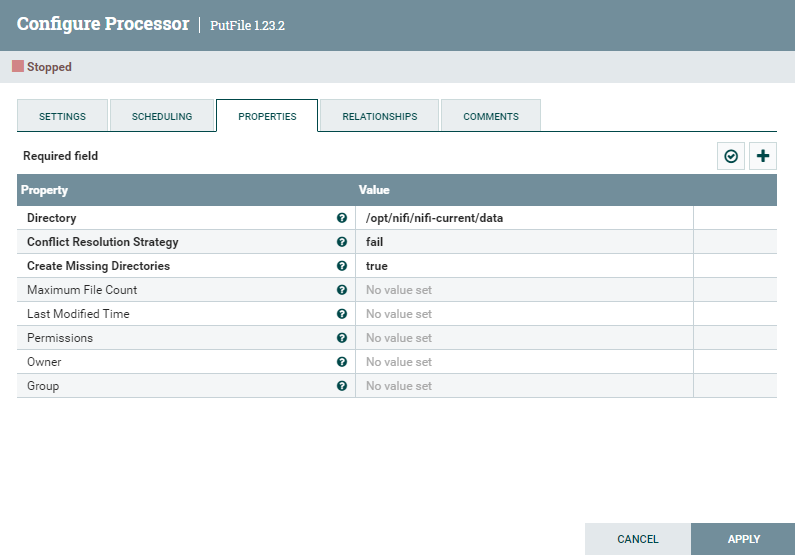


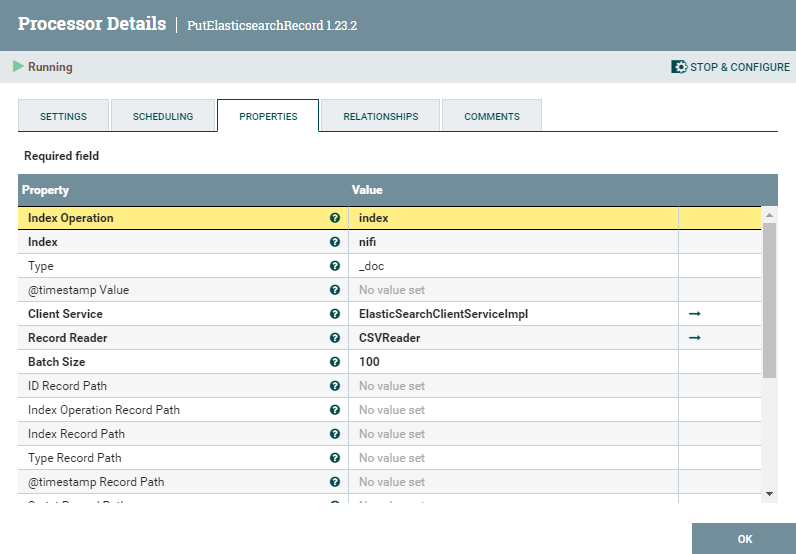
Поле correct содержит SELECT \* FROM FLOWFILE WHERE designation <> ' AND region\_1 <> '

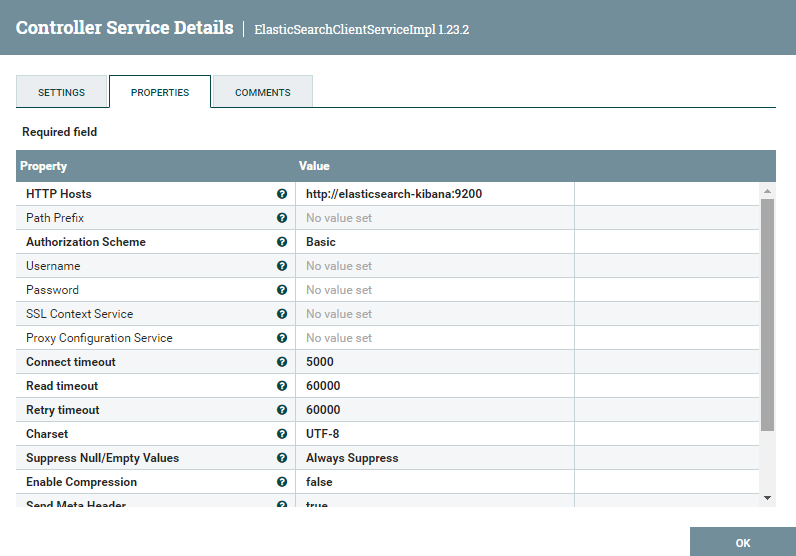


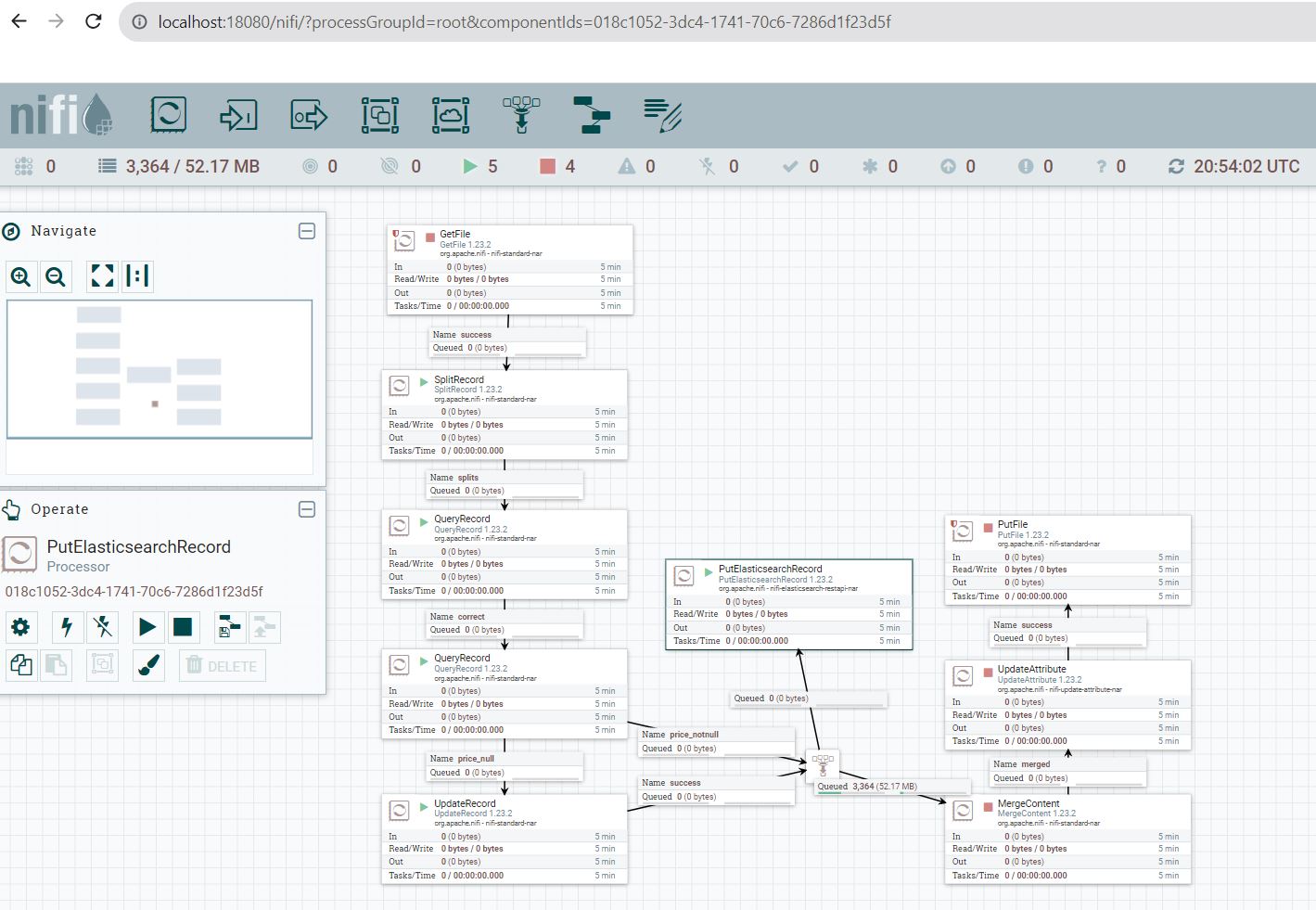




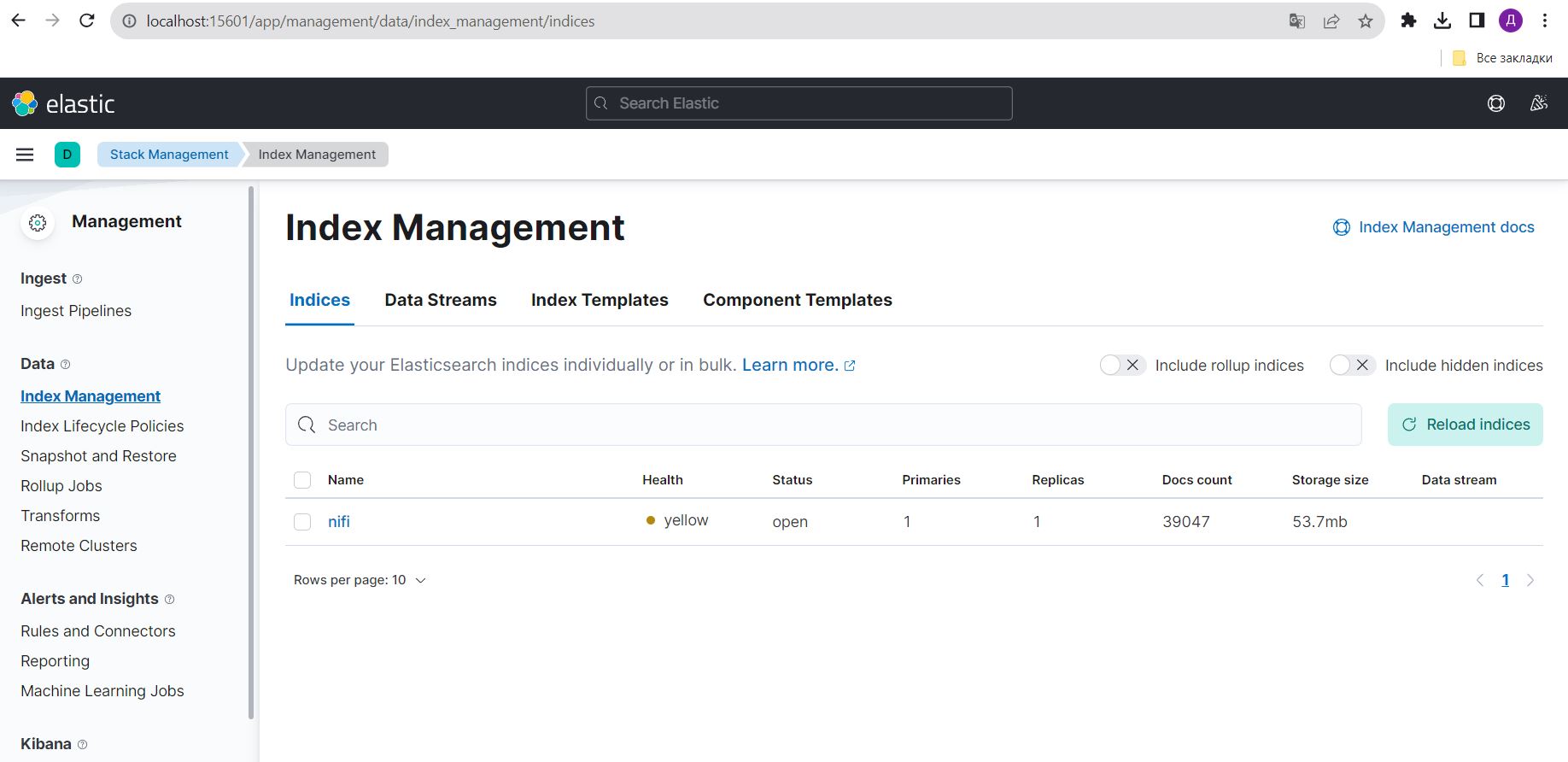




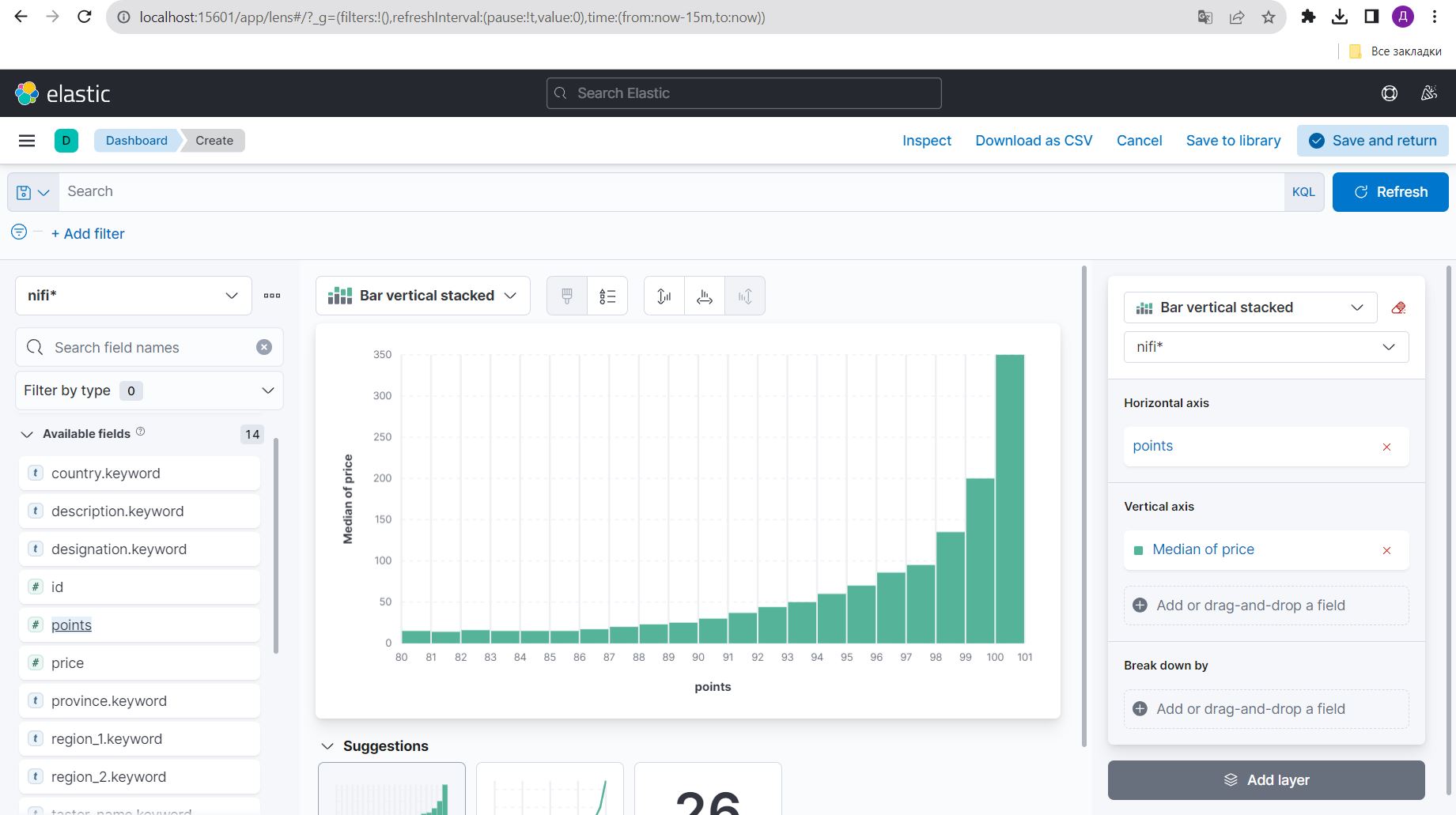
Для корректной загрузки данных в Elasticsearch необходимо добавить параметр HTTP Hosts, для этого нажимаем на стрелочку, далее на шестеренку напротив ElasticSearchClientServiceImpl и в появившемся окне вводим. адрес. 



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



Далее создаём паттерн и строим график



# 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



# 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