

Ход работы

Apache Nifi

Полная схема в системе NiFi представлена на рисунке 1.

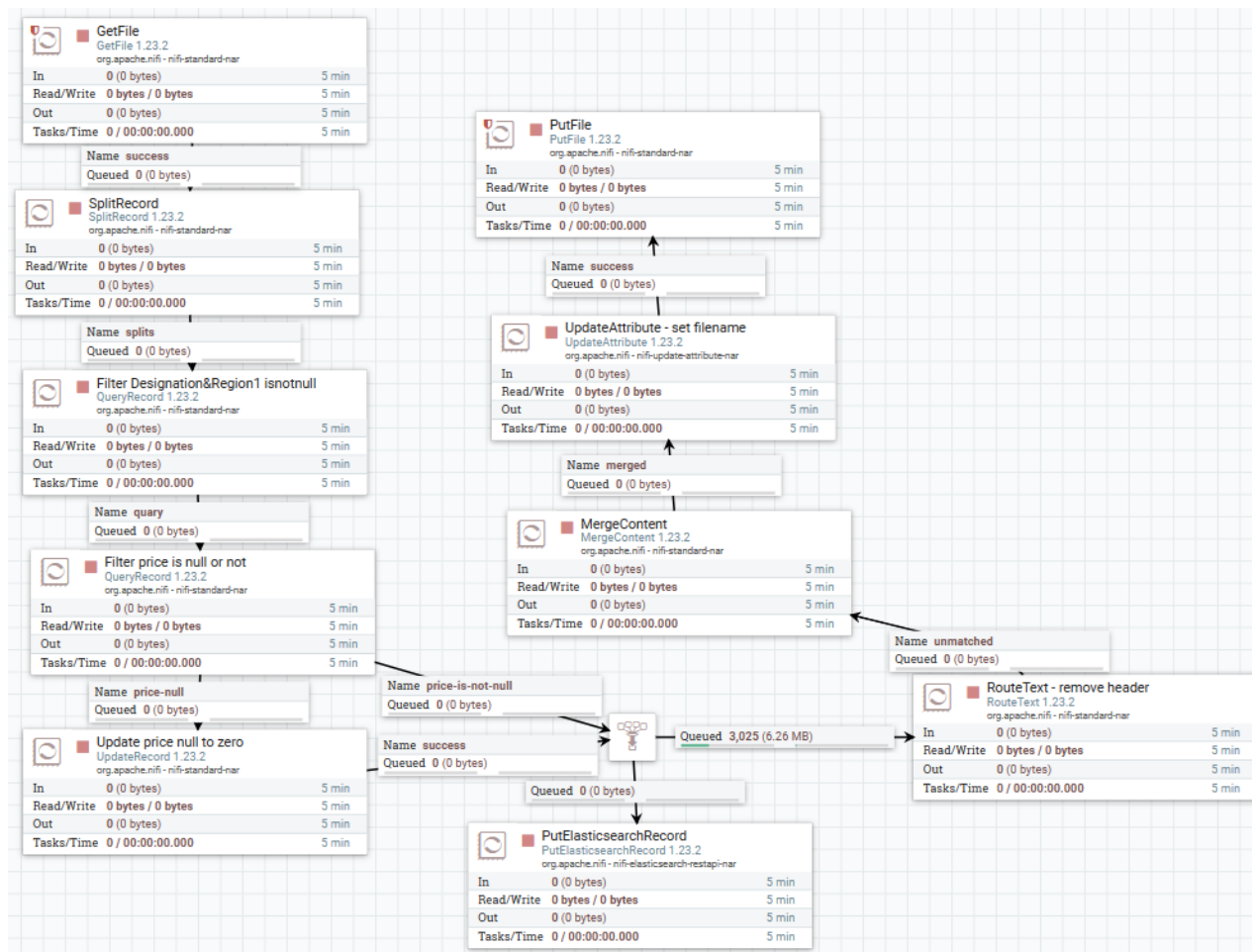


Рисунок 1 – Схема в системе NiFi

По схеме и с учетом названий смысл компонентов описывается. Однако есть момент с RouteText и MergeContent. Одна из проблем, что MergeContent соединяет csv данные дублируя шапку (header), поэтому итоговый csv файл имеет дубликаты шапки внутри файла как итог.

Основная задача RouteText удалить шапку везде, и при этом в MergeContent шапка вставляется сама, через следующие настройки на рисунке 2. MergeContent позволяет вручную вставить шапку, дабы нивелировать данную проблему. Однако, когда в UpdateAttribute указывается filename и далее PutFile сохраняет файл, происходит немного странное с тем, что шапка соединяется с первой строкой записи (см. рисунок 3). Основное решение данной проблемы, это вставление новой строки в шапку, указанной на рисунке 2. Однако я не понял, как это сделать, и такие вставки в конец как `${'\n'}`, `'\n'`, `${'\n'}` – не помогли.

Также про сохранение файла с именем. У файла filename берется из GetFile операции, но необходимо задавать ему какое-то более значимое имя. Я сделал так: идентификатор имени и случайное число (см. рисунок 4). Была проблема с тем, что файл мог создаваться несколько раз с одним именем, из-за чего NiFi выдавал ошибку, ибо такое делать нельзя. То есть имя должно быть уникальным. При этом идентификатор для разных сегментов одного файла – одинаковый, поэтому я присоединяю еще случайное число дабы была некоторая уникальность.

Configure Processor | MergeContent 1.23.2

Stopped

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

+

Property	Value
Attribute Strategy	Keep Only Common Attributes
Correlation Attribute Name	No value set
Minimum Number of Entries	1
Maximum Number of Entries	1000
Minimum Group Size	0 B
Maximum Group Size	No value set
Max Bin Age	No value set
Maximum number of Bins	5
Delimiter Strategy	Text
Header	id,country,description,designation,points,price,province,region_1,region_2,taster_name,taster_twitter_title,variety,winery,100 France Concentra Vieilles Vi
Footer	No value set
Demarcator	No value set

CANCEL

APPLY

Рисунок 2 – Настройки MergeContent

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	id	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_title	variety	winery	100 France	Concentra Vieilles Vi	90	0	Loire Valley	Muscadet S.A.	
2	100060	France	Ripe and fleshy	serve	87	20	Burgundy	Bourgogne		Roger Vos	@vossrog	Bouchard	Chardonnay	Bouchard P.A.	re & Fils					
3	100061	Italy	This open Campo	Lai	87	30	Veneto	Valpolicella Superiore				Ca' Rugate	Corvina, R	Ca' Rugate						
4	100062	US	Lightly scented	Lonesome	87	20	Washington	Columbia	Columbia	Paul Gregi	@paulgwi	Tildio 201	Sauvignon	Tildio						
5	100064	Argentina	Bright and Oak	Cask	87	10	Mendoza	Mendoza		Michael Si	@winesch	Trapiche 2	Syrah	Trapiche						
6	100065	US	Made from	Blanco	87	20	California	Edna Valley	Central Coast			Trenza 20	White	Ble Trenza						
7	100066	Italy	Lean and	Paverno	87	35	Veneto	Amarone della	Valpolicella Classico			Vaona 20	Corvina, R	Vaona						
8	100067	France	Soft and	Buissonni	87	21	Burgundy	Montagne		Roger Vos	@vossrog	Vignerons	Chardonnay	Vignerons de Buxy						
9	100068	France	From the	Marie-Ant	87	25	Burgundy	Pouilly-Fuiss	A	Roger Vos	@vossrog	Vincent 2	Chardonnay	Vincent						
10	100069	US	Showing	r Clone 777	87	35	California	Russian River	Sonoma			Windsor C	Pinot Noir	Windsor Oaks						

Рисунок 3 – Отсутствие новой строки при соединении шапки и строк файла

Processor Details | UpdateAttribute 1.23.2

Running

STOP & CONFIGURE

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property	Value
Delete Attributes Expression	No value set
Store State	Do not store state
Stateful Variables Initial Value	No value set
Cache Value Lookup Cache Size	100
filename	\${fragment.identifier}_\${random()}.csv

Рисунок 4 – Установка нового имени файла для сохранения

Nibana elastic-search

В NiFi создается соответствующий компонент PutElasticsearchRecord для отправки отфильтрованных данных (см. рисунок 5). Также необходима настройка сервиса для отправки (см. рисунок 6).

Configure Processor | PutElasticsearchRecord 1.23.2

Stopped

SETTINGS | SCHEDULING | **PROPERTIES** | RELATIONSHIPS | COMMENTS

Required field 🔍 +

Property	Value	
Index Operation	? index	
Index	? fromifi	
Type	? _doc	
@timestamp Value	? No value set	
Client Service	? ElasticSearchClientServiceImpl	→
Record Reader	? CSVReader	→
Batch Size	? 100	
ID Record Path	? No value set	
Index Operation Record Path	? No value set	
Index Record Path	? No value set	
Type Record Path	? No value set	
@timestamp Record Path	? No value set	

CANCEL APPLY

Рисунок 5 – Создание PutElasticsearchRecord

Controller Service Details | ElasticSearchClientServiceImpl 1.23.2

SETTINGS PROPERTIES COMMENTS

Required field

Property	Value
HTTP Hosts	? http://elasticsearch-kibana:9200
Path Prefix	? No value set
Authorization Scheme	? Basic
Username	? No value set
Password	? No value set
SSL Context Service	? No value set
Proxy Configuration Service	? No value set
Connect timeout	? 5000
Read timeout	? 60000
Retry timeout	? 60000
Charset	? UTF-8
Suppress Null/Empty Values	? Always Suppress
Enable Compression	? false
Send Meta Header	? true

OK

Рисунок 6 – Настройка сервиса для отправки

Теперь, отправим данные в данный сервис. В сервисе увидим, что данные пришли (см. рисунок 7).

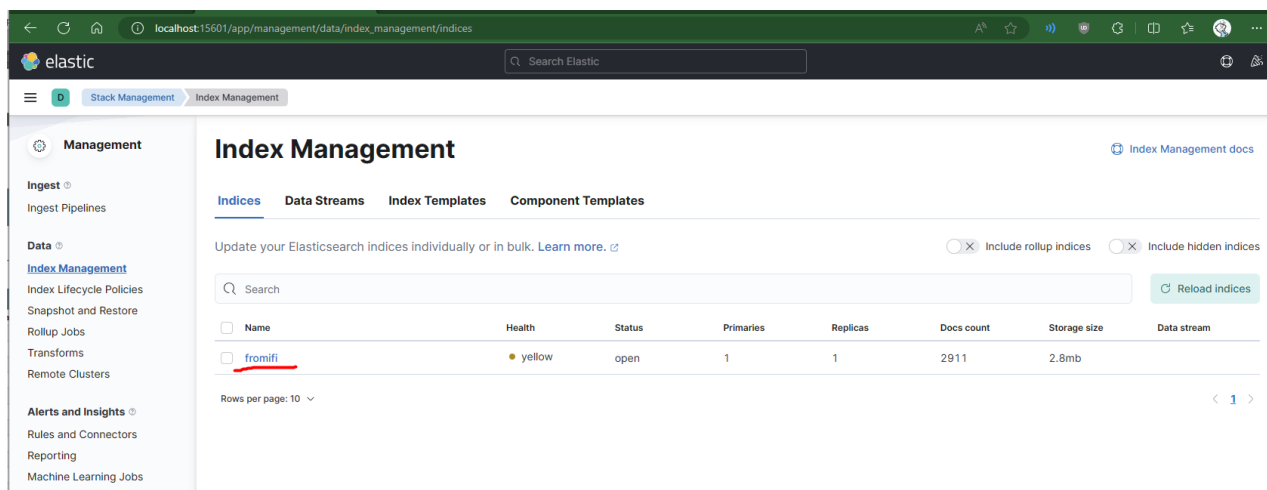


Рисунок 7 – Присланные данные из NiFi

Теперь в вкладке Kibana нужно для этого индекса создать индекс паттерн (см. рисунок 8), где указаны поля данных.

Теперь с данными созданным индексом паттернов можно приступать к визуализации (см. рисунок 9).

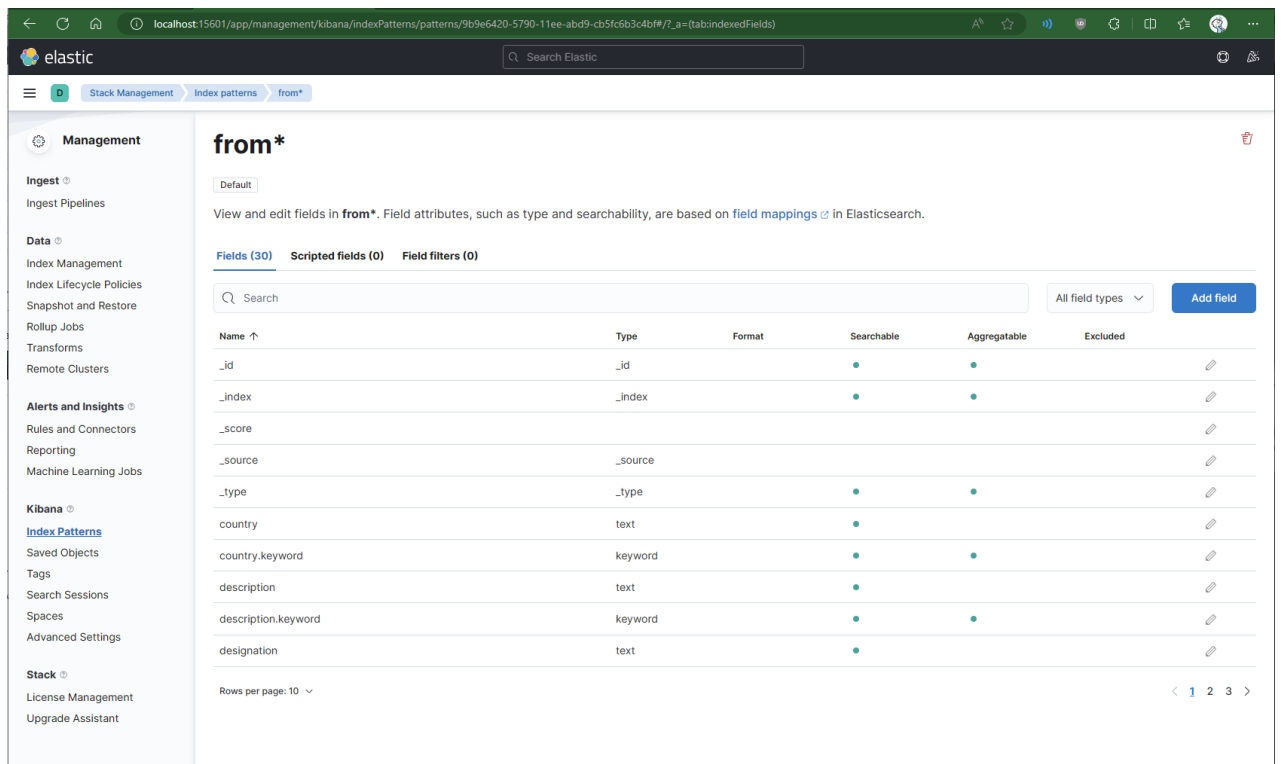


Рисунок 8 – Индекс паттерн для присланных данных

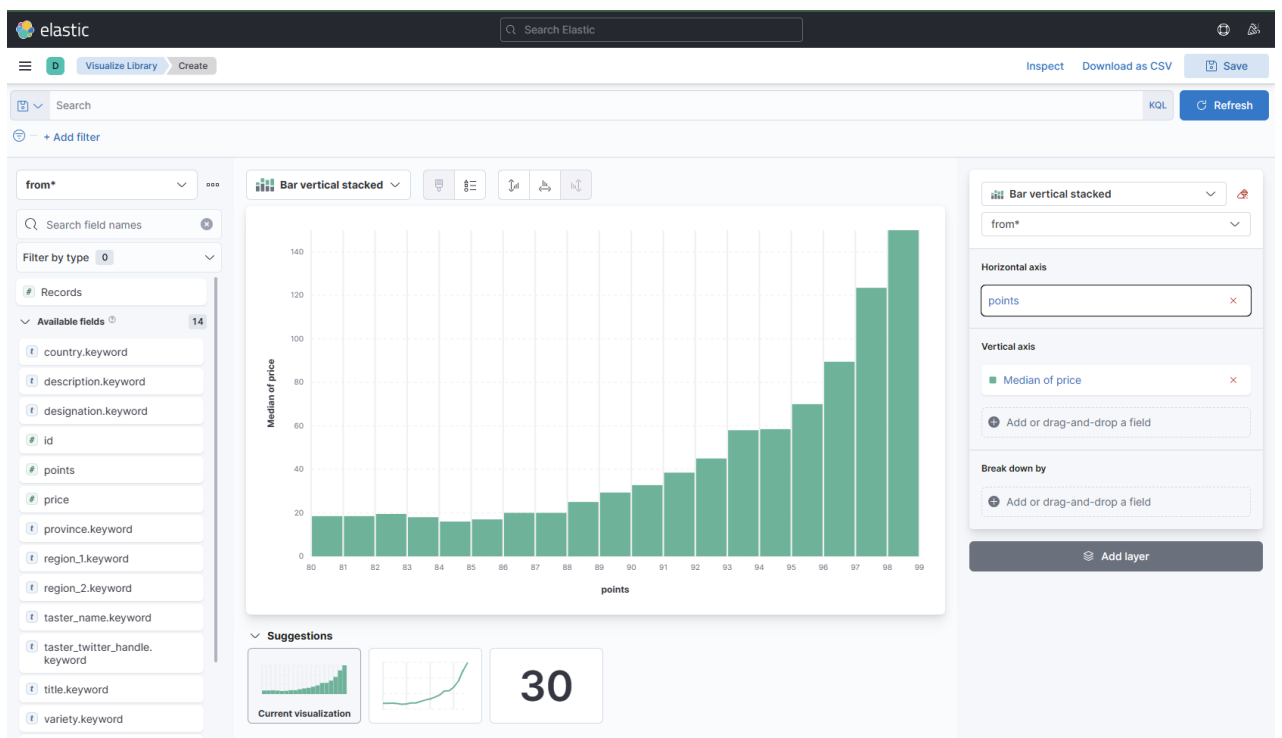


Рисунок 9 – Гистограмма зависимости цены от очков

Apache Airflow

Код компонента DAG:

```
1. @dag(
2.     start_date=datetime(2021, 1, 1),
3.     schedule_interval=None,
4.     catchup=False,
5.     doc_md=__doc__,
```

```

6. )
7. def data_clean_pipeline():
8.     filename_path_save = '/opt/airflow/data/lab1_output/lab1_output.csv'
9.     load_data_folder = '/opt/airflow/data/lab1'
10.
11.     @task
12.     def load_data(path_folder: str):
13.         csv_files = glob.glob(os.path.join(path_folder, '*.csv'))
14.         print(f'Found csv files: {len(csv_files)}')
15.
16.         return pd.concat(
17.             (
18.                 pd.read_csv(single_file)
19.                 for single_file in csv_files
20.             ),
21.             ignore_index=True
22.         )
23.
24.     @task
25.     def preprocessing(df: pd.DataFrame):
26.         df = df[df['designation'].notnull()]
27.         df = df[df['region_1'].notnull()]
28.         df.loc[df["price"].isnull(), "price"] = 0.0
29.
30.         return df
31.
32.     @task
33.     def save_data(df: pd.DataFrame, filename_path_save: str):
34.         df.to_csv(filename_path_save)
35.
36.     @task
37.     def save_data_to_elastic(df: pd.DataFrame):
38.         from elasticsearch import Elasticsearch
39.         from elasticsearch.helpers import bulk
40.         es = Elasticsearch(['elasticsearch-kibana'], port=9200)
41.
42.         data = df.to_json(orient='records')
43.         actions = [
44.             {
45.                 '_index': 'my_index',
46.                 '_type': 'my_type',
47.                 '_id': i,
48.                 '_source': doc
49.             }
50.             for i, doc in enumerate(data)
51.         ]
52.         # bulk(es, actions)
53.         es.bulk(actions)
54.
55.     df = load_data(load_data_folder)
56.     pd_preprocessed = preprocessing(df)
57.     save_data(pd_preprocessed, filename_path_save)
58.     save_data_to_elastic(pd_preprocessed)

```