

## Data Architecture, Integration and Ingestion

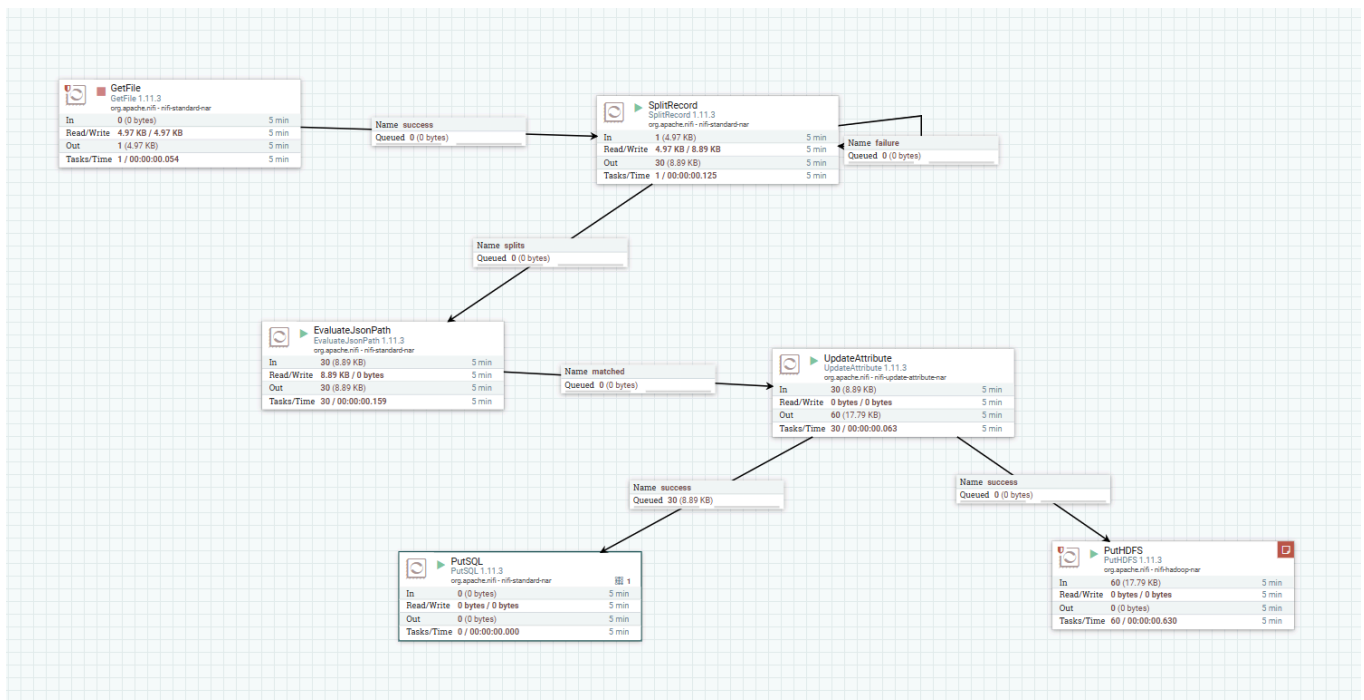
356520 NATANAEL DE OLIVEIRA CASTRO

355729 ALYSSON MARQUEZELLI

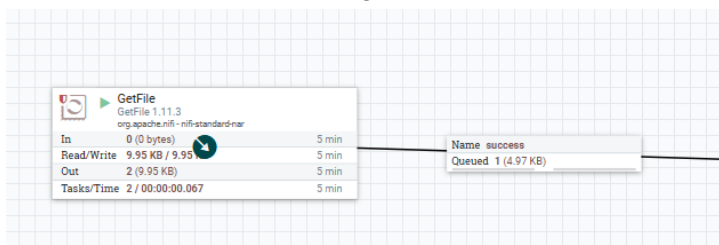
356427 MATEUS DA ROS SCHUMACHER

355741 VICTOR FERNANDES CARDOSO

- O fluxo abaixo consiste em adquirir uma carga de dados gerada no formato .csv e integrá-la ao HDFS no formato .json.
- Para esse trabalho, estamos utilizando o docker-compose para manipular os containers criados.
- O arquivo CSV foi copiado para o container do nifi e as configurações necessárias de para acesso do nifi e hdfs foram feitas baseadas no git [https://github.com/fabiogjardim/bigdata\\_docker.git](https://github.com/fabiogjardim/bigdata_docker.git)



O processor abaixo faz a ingestão do .csv que fica enfileirado em um único arquivo:

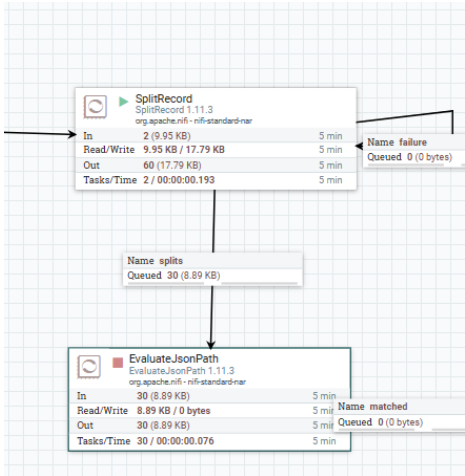


SUCCESS

Displaying 1 of 1 (4.97 KB)

	Position	UUID	Filename	File Size
1	1	805d8cc0-b482-4848-bf32-b4feaf9a203e	carga-clientes.csv	4.97 KB

Utilizamos então o SplitRecord para fazer a divisão do mesmo. Observe que ele gerou 30 itens na fila após o split ser executado com sucesso.



splits

Displaying 30 of 30 (8.89 KB)

	Position	UUID	Filename
1	1	d07d95bb-31df-4526-9fb8-e81f14c51179	carga-clientes.csv
2	2	6bc22f3c-2e78-44e5-8443-3cbbb8e8c7f	carga-clientes.csv
3	3	c6c42430-28c4-49ec-a294-ee6020835841	carga-clientes.csv
4	4	af02ab34-0d2f-4993-a0e3-fe1928ea8e06	carga-clientes.csv
5	5	4a2ea64e-25a3-4842-b36d-1cc5386ba12f	carga-clientes.csv
6	6	2e6f5ed2-a040-4a36-8701-e58a3ce65cef	carga-clientes.csv

Mas ainda temos um problema, pois o nome do arquivo (coluna filename) ainda se repete n-vezes. Para resolver isso, conectamos a um UpdateAttribute e configuramos da maneira abaixo:

Running

SETTINGS SCHEDULING PROPERTIES 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	\$(filename).json

Após a execução temos esse resultado:

success

Displaying 30 of 30 (8.89 KB)

	Position	UUID	Filename
1	1	d07d95bb-31df-4526-9fb8-e81f14c51179	Vanessa Cunha.json
2	2	6bc22f3c-2e78-44e5-8443-3cbbb8e8c7f	Vinicius Ferreira.json
3	3	c6c42430-28c4-49ec-a294-ee6020835841	Marcelo Carvalho.json
4	4	af02ab34-0d2f-4993-a0e3-fe1928ea8e06	Sabrina Silva.json
5	5	4a2ea64e-25a3-4842-b36d-1cc5386ba12f	Mariana Oliveira.json
6	6	2e6f5ed2-a040-4a36-8701-e58a3ce65cef	Paulo Lima.json
7	7	9b044973-2a57-4a6c-8d38-3a5e9ee0ff4	Isabela Mendes.json

E seu conteúdo:

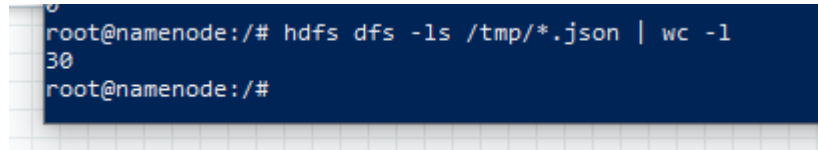
View as: original

Filename: Vanessa Cunha.json  
Content Type: application/json

```
1 [{"id":26,"nome":"Vanessa Cunha","email":"vanessa.cunha@example.com","telefone":"+5511754321098","endereco":"Rua do Lago, 1111","cidade":"São Luis","estado":"MA","pais":"Brasil","datanascimento":"1992-09-18","foto":"imagem26.jpg","assin
```

Por fim, adicionamos um PutHDFS e podemos ver o resultado da ingestão dos arquivos de csv para json para o container HDFS. Os dados são extraídos pelo nifi, transformados e imputados no Hadoop

Efetuada a instrução abaixo, podemos confirmar os 30 arquivos .json corretamente copiados para o hdfs



```
root@namenode:/# hdfs dfs -ls /tmp/*.json | wc -l
30
root@namenode:/#
```

Com esses arquivos .json disponibilizados no hdfs, temos algumas ações extras a esse trabalho, das quais podemos listar:

- \* Processamento em Lote com MapReduce;
- \* Consulta de Dados com Apache Hive;
- \* Efetuar análises com Spark e etc;