Creating a Transfer Operator PostgresqlToElasticsearch

Add a custom operator

The Operator

In this lesson we are going to create an Operator to move data from PostgreSQL to Elasticsearch using the Hook we made previously.

Again, the goal here is not to make the perfect operator, but just to show you a good example of how to implement a useful transfer operator.

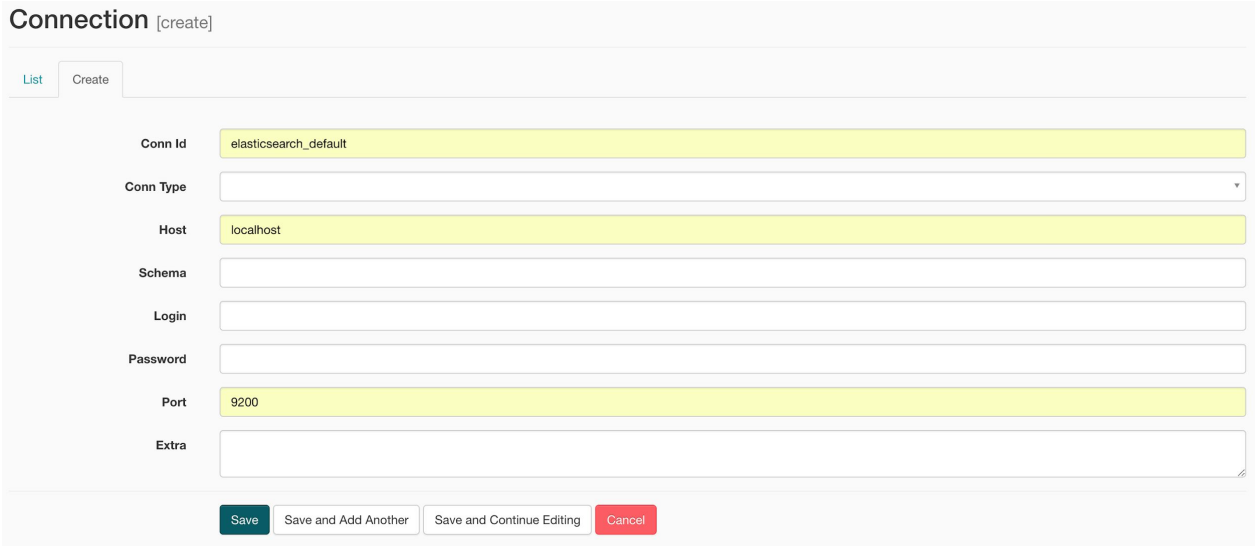
Let's do it!

Instructions

- `vim ~/airflow/plugins/elasticsearch_plugin/operators/elasticsearch_operator.py`
- `vim ~/airflow/plugins/elasticsearch_plugin/__init__.py`

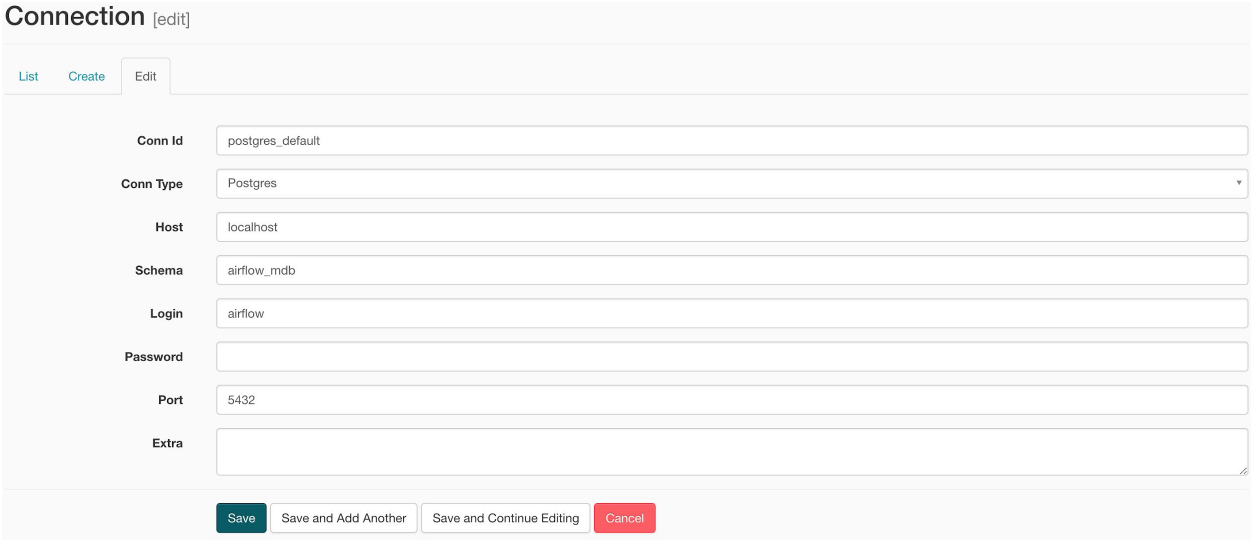
Instructions

- From the Airflow UI, click on “Admin”, “Connections” and “Create”.
- We are going to **create** the connection to Elasticsearch used by our Hook.
- Fill the fields with the same values and click on save.



Instructions

- From the Airflow UI, click on “Admin”, “Connections” and “Create”.
- We are going to **edit** the connection to PostgreSQL used by our Operator.
- Fill the fields with the same values and click on save.



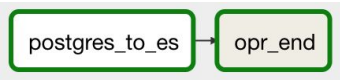
Instructions

- `cp ~/airflow_files/plugin_operator_dag.py ~/airflow/dags/`
- `vim ~/airflow/dags/plugin_operator_dag.py`
- On the Airflow UI, trigger the DAG “plugin_operator_dag” as follow:

- Then, go to the “Graph View” and wait for the DAG to finish.

Instructions

- Once the DAG has finished executing you should have the following DAG from the “Graph View”:



- If you click on the task “postgres_to_es” and “View Log”, you will notice that many POST requests have been sent to Elasticsearch corresponding to the number of records we fetched from PostgreSQL.

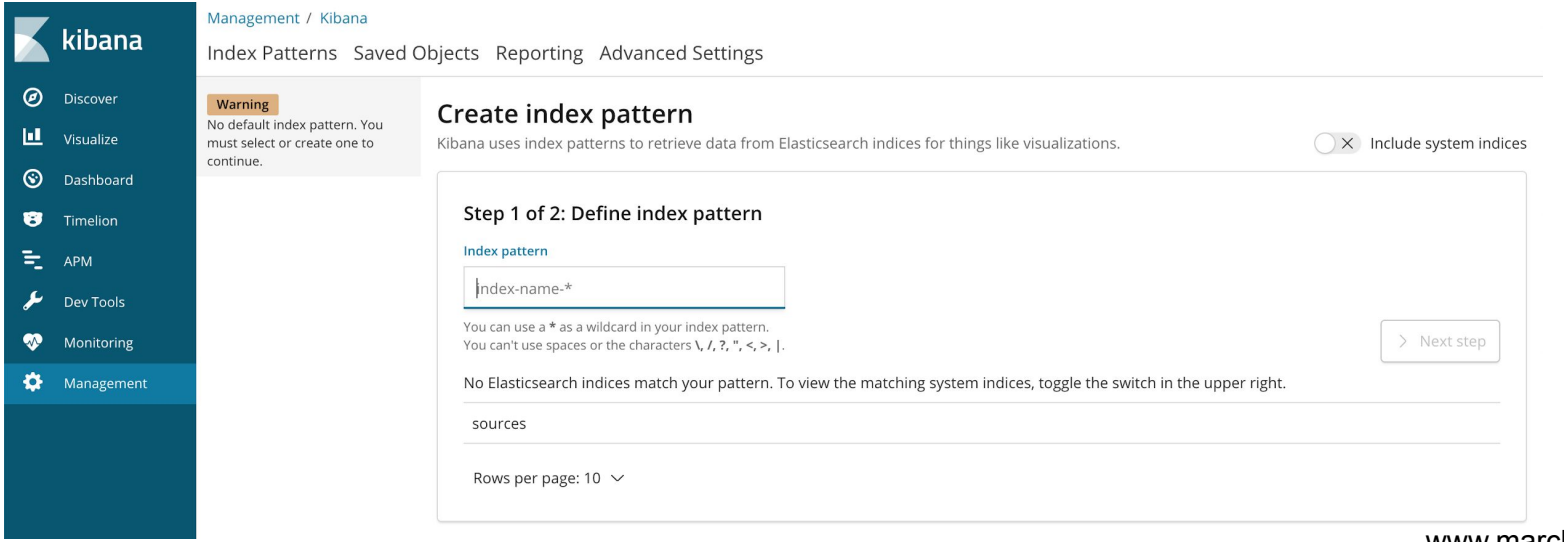
```
[2018-11-15 03:35:35,757] {elasticsearch_operator.py:41} INFO - Extracting data from PostgreSQL: SELECT * FROM course.source
[2018-11-15 03:35:37,109] {logging_mixin.py:95} INFO - [2018-11-15 03:35:37,108] {base.py:83} INFO - POST http://localhost:9200/sources/external [status:201 request:1.350s]

[2018-11-15 03:35:37,207] {logging_mixin.py:95} INFO - [2018-11-15 03:35:37,207] {base.py:83} INFO - POST http://localhost:9200/sources/external [status:201 request:0.098s]

[2018-11-15 03:35:37,232] {logging_mixin.py:95} INFO - [2018-11-15 03:35:37,232] {base.py:83} INFO - POST http://localhost:9200/sources/external [status:201 request:0.024s]

[2018-11-15 03:35:37,244] {logging_mixin.py:95} INFO - [2018-11-15 03:35:37,244] {base.py:83} INFO - POST http://localhost:9200/sources/external [status:201 request:0.011s]
```

Instructions

- Now, let’s use Kibana (the interface of Elasticsearch) to see if all the data have been saved into Elasticsearch.
- I’m not going to explain how Elasticsearch works here, but just keep in mind that data are saved into an “index” representing a collection of documents.
- In our case, each document corresponds to a record saved from PostgreSQL.

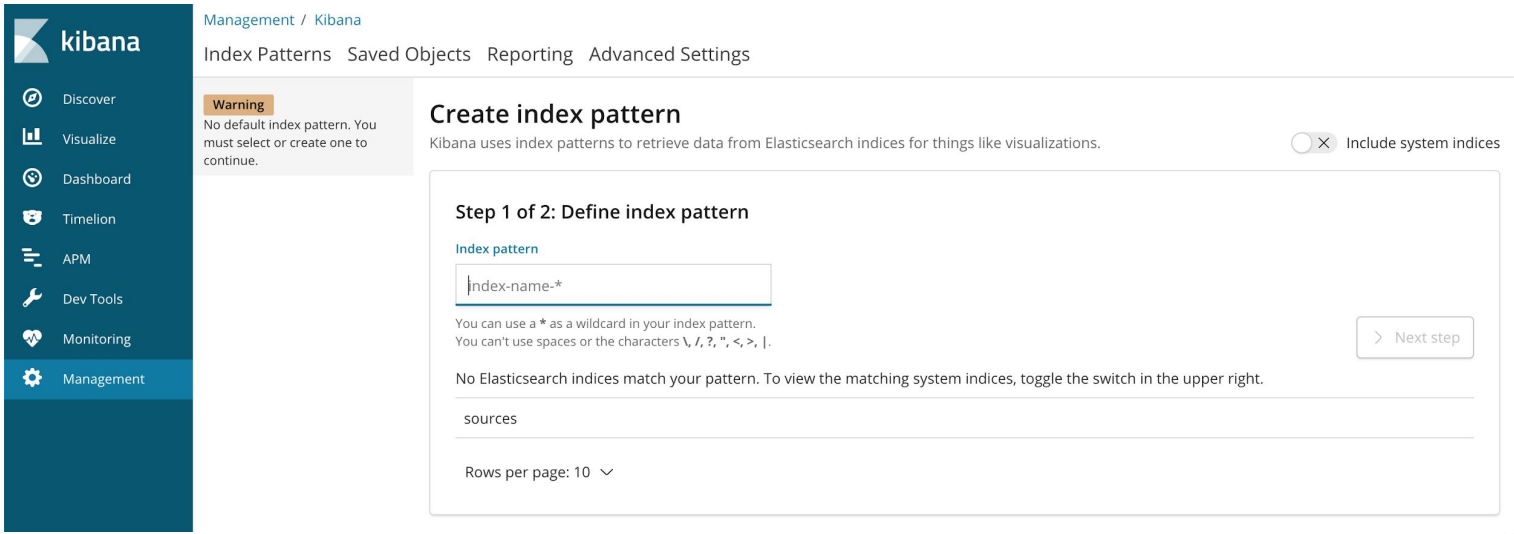
Instructions

- Go to http://localhost:5601 and click on “Management” on the left sidebar,



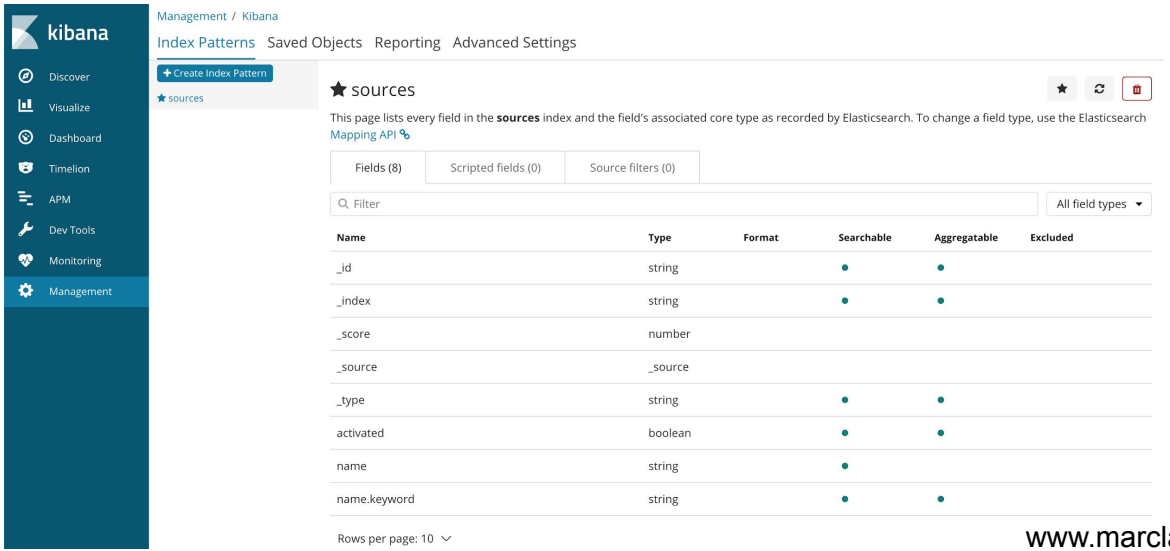
Instructions

- In the text field below “Index Pattern” type “sources”,



Instructions

- Click on “Next Step” and “Create Index Pattern”. Once you have done that you should see the following screen:



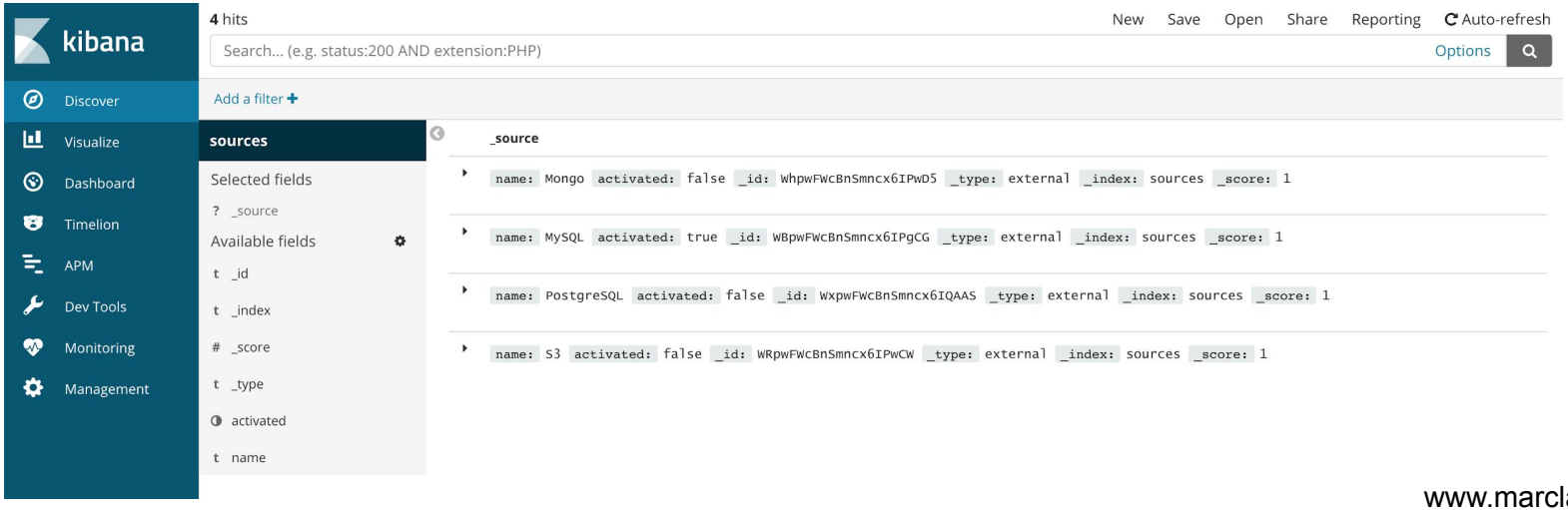
Instructions

- If you take a look at the field names, you should notice that we get the same names corresponding to the column we have from PostgreSQL as you can see below,

Name	Type	Format	Searchable	Aggregatable	Excluded
_id	string		●	●	
_index	string		●	●	
_score	number				
_source	_source				
_type	string		●	●	
activated	boolean		●	●	
name	string		●		
name.keyword	string		●	●	

Instructions

- Finally, click on “Discover” on the left sidebar to see the same data we have from PostgreSQL transferred to Elasticsearch.



Important Notes

- When you create an Operator, you must implement the method `execute(self, context)`