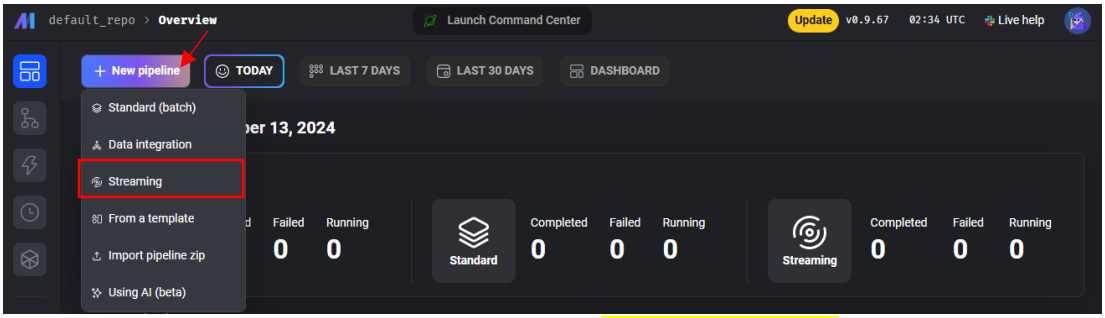
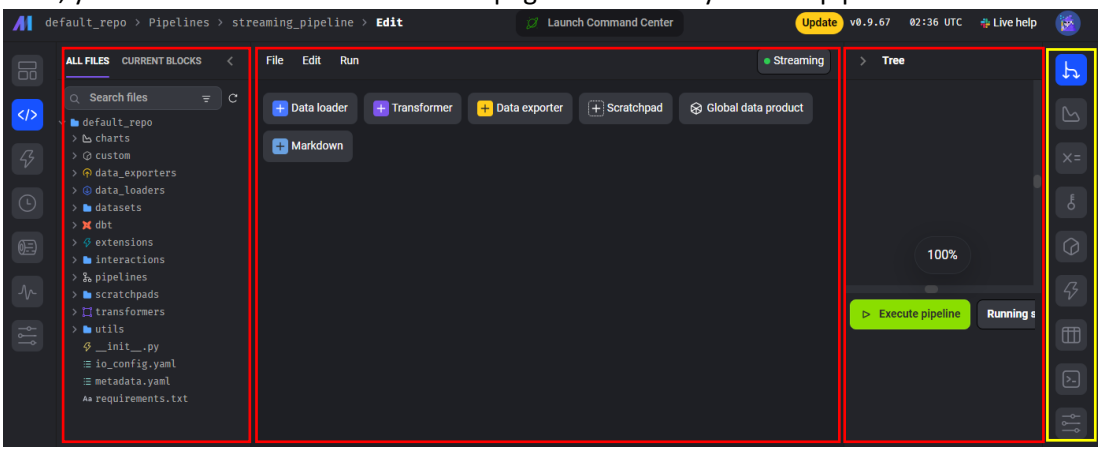
In this tutorial, we'll focus on the "**Extract**", "**Transform**", and "**Load**" steps in the ETL process using Mage.ai and Python for streaming data from Kafka. Don't worry if you're not familiar with them—this guide is designed to be easy to follow! 😊

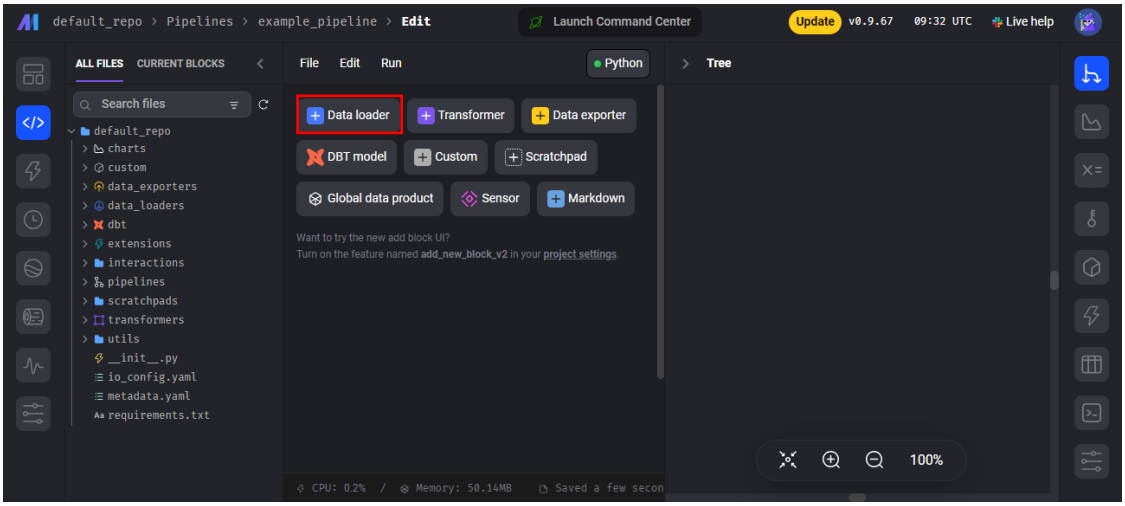
1. Click on "**+ New pipeline**" button and select "**Streaming**".



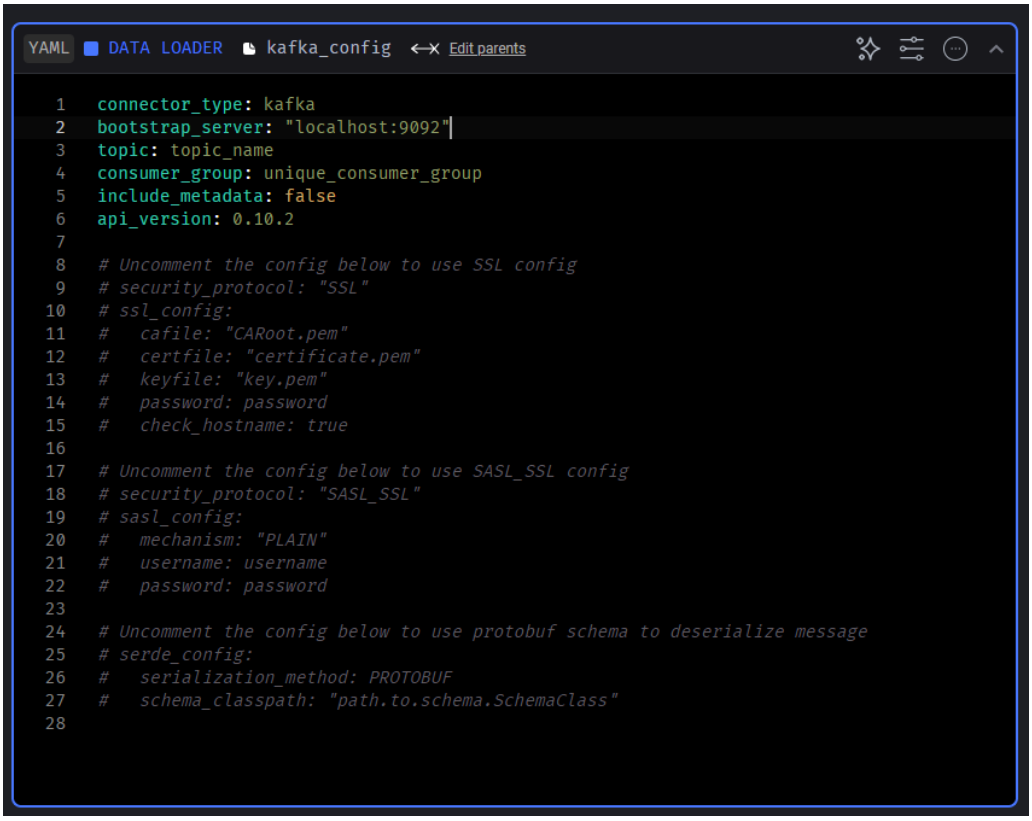
   a. Mage.ai offers different types of pipelines (TODO: Add usages):
      i. Standard (batch),
      ii. Data Integration,
      iii. Streaming,
      iv. From a template,
      v. Import pipeline zip, and
      vi. Using AI.
   b. You may change/add the pipeline's
      i. Name,
      ii. Description, and/or
      iii. Tags to this new pipeline.
2. Next, you will be redirected to the "**Edit**" page for the newly created pipeline.



   a. Left pane (Directory pane): Directory for "**/home/src/default_repo**".
   b. Middle pane: Allows you to choose any to add a block to your pipeline and serves as a code editor.
   c. Right pane: To view the items relevant to the icons on the extreme right that is boxed up in yellow.
3. Firstly, we will load the data from the file to Mage.ai:
   Select "**+ Data loader**" > "**Kafka**". Change the name of this block to "**kafka_config**".



a. A code editor will be added to the middle pane.
b. In the code editor, note that you're viewing the code for *Data Loader* that is named "**kafka_config**".
c. The following block will be created in the middle pane.



d. Replace lines 2 and 3 with the following:

```
bootstrap_server: "kafka:9093"
topic: topic
```

i. This will configure this pipeline to stream messages from hostname "**kafka**" running on port "**9093**" that has messages published on topicname "**topic**".

4. Secondly, we will **transform** the data based on the plan we ideated previously.
   a. Get the "**year**" from "**epi_week**".
      i. Scroll to the bottom of the middle pane and add a *Transformer*: Select "**Transformer**" block > "**Python**" > "**Generic (no template)**".

ii. The function "**transform**" will be executed when this block runs. The first parameter that this function takes in ("**messages**") is the data being streamed in from Kafka.
iii. To extract "**year**" from "**epi_week**", replace the code in "**transform**" function with the following. We will convert "**messages**" to Pandas' DataFrame type to transform the data in an orderly manner.

```
df = pd.DataFrame(messages)

# Extracts the first 4 characters of each value and store it under the "year" col
df['year'] = df['epi_week'].str.slice(0, 4)
# Removes "epi_week" column from the dataframe
df.drop(columns=['epi_week'], inplace=True)

# Returns this updated dataframe to pass it to the next block
return df.to_dict('records')
```

b. Get the new number of cases based on year and disease:
i. Add another *Transformer* block by going to the end of the middle pane and select "**Transformer**" block > "**Python**" > "**Generic (no template)**".
ii. Similar to the "**transform**" function mentioned in Section 4.a.ii (TODO: Add link), the first parameter taken in by the function "**transform**" in this newly created *Transformer* is the list of dictionaries returned from the previous *Transformer*.
iii. To get the new number of cases, group the data based on its disease and year values for that row and sum the no._of_cases.

```
df = pd.DataFrame(messages)

# Groups data by disease and year to get the summed no._of_cases value
df = df.groupby(['disease', 'year']).agg({'no._of_cases': 'sum'}).reset_index()
return df.to_dict('records')
```

c. Drop duplicate rows
i. Based on Section 4.b.iii (TODO: Add link), there may have many duplicated rows in the dataframe (i.e. same year, disease, and number of cases).
ii. Add a new *Transformer* from the bottom of the middle pane: select "**Transformer**" block > "**Python**" > "**Generic (no template)**" and replace the code in "**transform**" function with the following:

```
df = pd.DataFrame(messages)

df.drop_duplicates(inplace=True)
return df.to_dict('records')
```

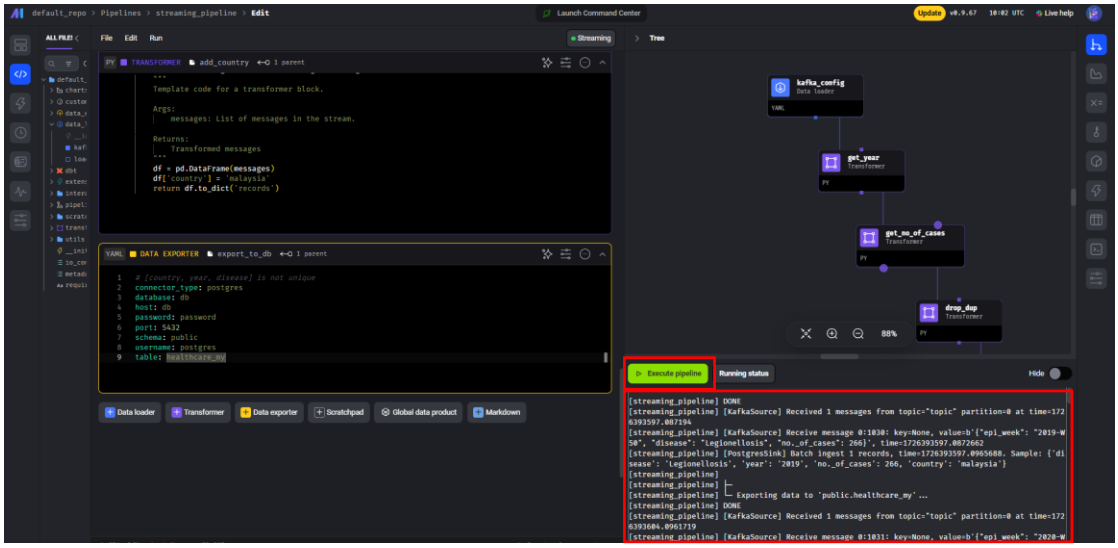
d. Add "**country**" column
i. Now that our data is ready, we want to add a new column called "**country**" and the values in this column will all be "**malaysia**". This allows us to add other countries' similar health data in the future.
ii. To do so, add a generic *Transformer* block like in Section 4.a.i (TODO: Add link) and replaced the function "**transform**" code with the following:

```
df = pd.DataFrame(messages)
df['country'] = 'malaysia'
return df.to_dict('records')
```

5. Lastly, let's **load** this data into your database.
    a. Let's add a *Data Exporter* block at the end of the middle pane: Select "**Data Exporter**" > "**PostgreSQL**".
    b. Update the code in this newly added block to the following:

```
# [country, year, disease] is not unique
connector_type: postgres
database: db
host: db
password: password
port: 5432
schema: public
username: postgres
table: healthcare_my
```

6. Now that our pipeline is ready, we can execute the pipeline on the right pane of this page:



    a. The output from the pipeline execution will be displayed in the bigger red box. You may stop your pipeline execution at any time.

For more information regarding Mage.ai's documentation, please refer to https://docs.mage.ai/.