(https://colab.research.google.com/github/sergejhorvat/MLEP-public/blob/main/course4/week2-ungraded-labs/C4_W2_Lab_4_ETL_Beam/C4_W2_Lab_4_Apache_Beam_and_Tensorflow.ipynb)

Ungraded Lab (Optional): ETL Pipelines and Batch Predictions with Apache Beam and Tensorflow

In this lab, you will create, train, evaluate, and make predictions on a model using Apache Beam (https://beam.apache.org/) and TensorFlow (https://www.tensorflow.org/). In particular, you will train a model to predict the molecular energy based on the number of carbon, hydrogen, oxygen, and nitrogen atoms.

This lab is marked as optional because you will not be interacting with Beam-based systems directly in future exercises. Other courses of this specialization also use tools that abstract this layer. Nonetheless, it would be good to be familiar with it since it is used under the hood by TFX which is the main ML pipelines framework that you will use in other labs. Seeing how these systems work will let you explore other codebases that use this tool more freely and even make contributions or bug fixes as you see fit. If you don't know the basics of Beam yet, we encourage you to look at the Minimal Word Count example here (https://beam.apache.org/get-started/wordcount-example/) for a quick start and use the Beam Programming Guide (https://beam.apache.org/documentation/programming-guide) to look up concepts if needed.

The entire pipeline can be divided into four phases:

1. Data extraction
2. Preprocessing the data
3. Training the model
4. Doing predictions

You will focus particularly on Phase 2 (Preprocessing) and a bit of Phase 4 (Predictions) because these use Beam in its implementation.

Let's begin!

Note: This tutorial uses code, images, and discussion from this article (https://cloud.google.com/dataflow/examples/molecules-walkthrough). We highlighted a few key parts and updated some of the code to use more recent versions. Also, we focused on making the lab running locally. The original article linked above contain instructions on running it in GCP. Just take note that it will have associated costs depending on the resources you use.

Initial setup

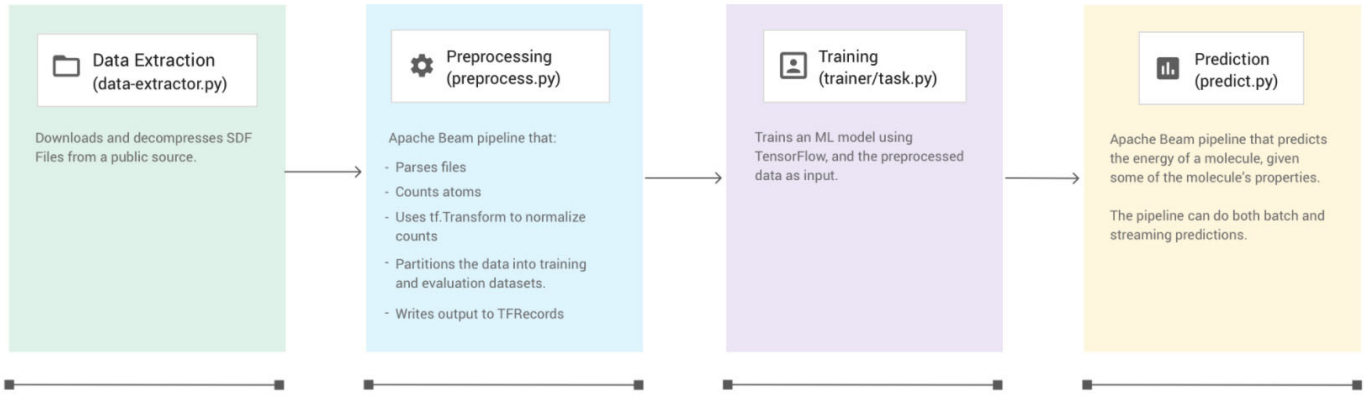
You will first download the scripts that you will use in the lab.

In [1]:
```
# Download the scripts
!wget https://github.com/https-deeplearning-ai/machine-learning-engineering-for-production-public/raw/main/course4/week2-ungraded-labs/C4_W2_Lab_4_ETL_Beam/data/molecules.tar.gz

# Unzip the archive
!tar -xvzf molecules.tar.gz
```

```
--2021-09-24 13:35:37--  https://github.com/https-deeplearning-ai/machine-learning-engineering-for-production-public/raw/main/course4/week2-ungraded-labs/C4_W2_Lab_4_ETL_Beam/data/molecules.tar.gz
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/https-deeplearning-ai/machine-learning-engineering-for-production-public/main/course4/week2-ungraded-labs/C4_W2_Lab_4_ETL_Beam/data/molecules.tar.gz [following]
--2021-09-24 13:35:38--  https://raw.githubusercontent.com/https-deeplearning-ai/machine-learning-engineering-for-production-public/main/course4/week2-ungraded-labs/C4_W2_Lab_4_ETL_Beam/data/molecules.tar.gz
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10622 (10K) [application/octet-stream]
Saving to: 'molecules.tar.gz'

molecules.tar.gz    100%[===================>]  10.37K  --.-KB/s    in 0s

2021-09-24 13:35:38 (77.0 MB/s) - 'molecules.tar.gz' saved [10622/10622]

molecules/
molecules/pubchem/
molecules/pubchem/pipeline.py
molecules/pubchem/sdf.py
molecules/pubchem/__init__.py
molecules/data-extractor.py
molecules/predict.py
molecules/requirements.txt
molecules/trainer/
molecules/trainer/task.py
```

```
molecules/trainer/__init__.py
molecules/preprocess.py
```

The `molecules` directory you downloaded mainly contain 4 scripts that encapsulate all phases of the workflow you will execute in this lab. It is summarized by the figure below:



In addition, it also contains these additional files and directories:

- `pubchem` - subdirectory which contains common modules (i.e. `pipeline.py` and `sdf.py`) shared by the preprocessing and predicition phases. If you look at `preprocess.py` and `predict.py`, you can see the line `import as pubchem` at the top.
- `requirements.txt` - contains packages to install in this Colab. These are Apache Beam and Tensorflow Transform. These allows you to create Extract-Transform-Load (ETL) pipelines and preprocess data. Let's install them in the next cell.

In [ ]:
```
# Install required packages
!pip install -r ./molecules/requirements.txt
```

Note: In Google Colab, you need to restart the runtime at this point to finalize updating the packages you just installed. You can do so by clicking the `Restart Runtime` button at the end of the output cell above (after installation), or by selecting `Runtime > Restart Runtime` in the Menu bar. ***Please do not proceed to the next section without restarting.***

Next, you'll define the working directory to contain all the results you will generate in this exercise. After each phase, you can open the Colab file explorer on the left and look under the `results` directory to see the new files and directories generated.

In [1]:
```
# Define working directory
WORK_DIR = "results"
```

With that, you are now ready to execute the pipeline. As shown in the figure earlier, the logic is already implemented in the four main scripts. You will run them one by one and the next sections will discuss relevant detail and the outputs generated.

Phase 1: Data extraction

The first step is to extract the input data. The dataset is stored as `SDF` (https://en.wikipedia.org/wiki/Chemical_table_file#SDF) files and is extracted from the National Center for Biotechnology Information (https://www.ncbi.nlm.nih.gov/) (FTP source). Chapter 6 of this document (http://c4.cabrillo.edu/404/ctfile.pdf) shows a more detailed description of the SDF file format.

The `data-extractor.py` file extracts and decompresses the specified SDF files. In later steps, the example preprocesses these files and uses the data to train and evaluate the machine learning model. The file extracts the SDF files from the public source and stores them in a subdirectory inside the specified working directory.

As you can see here (https://ftp.ncbi.nlm.nih.gov/pubchem/Compound_3D/01_conf_per_cmpd/SDF/), the complete set of files is huge and can easily exceed storage limits in Colab. For this exercise, you will just download one file. You can use the script as shown in the cells below:

```
                        d_SD/01_conf_per_cmpd/SDF ])
  --filter-regex FILTER_REGEX
                        Regular expression to filter which files to use. The
                        regular expression will be searched on the full
                        absolute path. Every match will be kept. (default:
                        \.sdf)
  --max-data-files MAX_DATA_FILES
                        Maximum number of data files for every file pattern
                        expansion. Set to -1 to use all files. (default: None)
```

In [3]:
```
# Run the data extractor
!python ./molecules/data-extractor.py --max-data-files 1 --work-dir={WORK_DIR}
```

```
Found 6227 files, using 1
Extracting data files...
Extracted results/data/00000001_00025000.sdf
```

You should now have a new folder in your work directory called `data`. This will contain the SDF file you downloaded.

In [4]:
```
# List working directory
!ls {WORK_DIR}
```

```
data
```

In the SDF Documentation linked earlier, it shows that one record is terminated by `$$$$`. You can use the command below to print the first one in the file. As you'll see, just one record is already pretty long. In the next phase, you'll feed these records in a pipeline that will transform these into a form that can be consumed by our model.

In [5]:
```
# Print one record
!sed '/$$$$/q' {WORK_DIR}/data/00000001_00025000.sdf
```

```
1
  -OEChem-09192103043D

 31 30  0     1  0  0  0  0  0999 V2000
    0.3387    0.9262    0.4600 O   0  0  0  0  0  0  0  0  0  0  0  0
    3.4786   -1.7069   -0.3119 O   0  5  0  0  0  0  0  0  0  0  0  0
    1.8428   -1.4073    1.2523 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.4166    2.5213   -1.2091 O   0  0  0  0  0  0  0  0  0  0  0  0
   -2.2359   -0.7251    0.0270 N   0  3  0  0  0  0  0  0  0  0  0  0
   -0.7783   -1.1579    0.0914 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.1368   -0.0961   -0.5161 C   0  0  2  0  0  0  0  0  0  0  0  0
   -3.1119   -1.7972    0.6590 C   0  0  0  0  0  0  0  0  0  0  0  0
   -2.4103    0.5837    0.7840 C   0  0  0  0  0  0  0  0  0  0  0  0
   -2.6433   -0.5289   -1.4260 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.4879   -0.6438   -0.9795 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.3478   -1.3163    0.1002 C   0  0  0  0  0  0  0  0  0  0  0  0
```

```
 10 24  1  0  0  0  0
 10 25  1  0  0  0  0
 10 26  1  0  0  0  0
 11 12  1  0  0  0  0
 11 27  1  0  0  0  0
 11 28  1  0  0  0  0
 13 14  1  0  0  0  0
 14 29  1  0  0  0  0
 14 30  1  0  0  0  0
 14 31  1  0  0  0  0
M  CHG  2   2  -1   5   1
M  END
> <PUBCHEM_COMPOUND_CID>
1

> <PUBCHEM_CONFORMER_RMSD>
0.6

> <PUBCHEM_CONFORMER_DIVERSEORDER>
2
43
65
46
25
35
57
19
53
42
```

> <PUBCHEM_MMFF94_PARTIAL_CHARGES>
14
1 -0.43
10 0.5
11 -0.11
12 0.91
13 0.66
14 0.06
2 -0.9