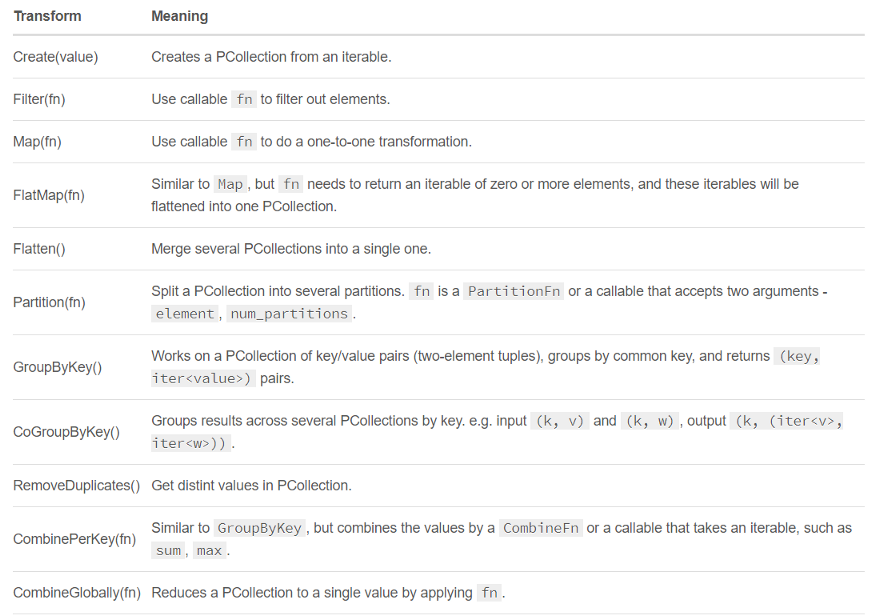
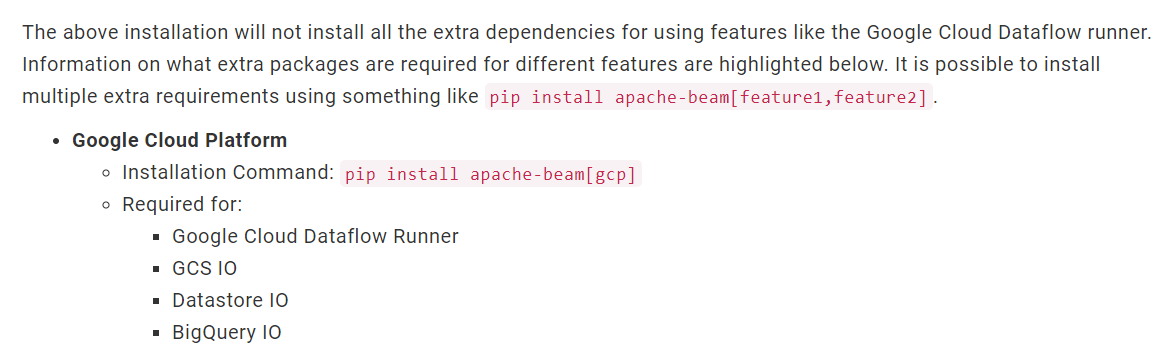
**Apache Beam:**

* Apache beam, the latest open source project of apache is a **unified** programming model for expressing efficient and **portable** big data processing pipelines.
  + **Unified** API to process both batch and streaming data.
  + **B**atch + Str**eam** -> Beam
  + **Portable,** beam pipeline once created in any language can be able to run on any execution frameworks like spark, flink, apex, cloud dataflow etc.,
  + Cloud Dataflow is fully managed service for creating and executing optimized parallel data processing pipelines.
  + Beam is a programming model whereas flink and spark are execution engines.
* **Flow of beam Programming model.**
* **Basic Terminologies in Beam.**
  + **Pipeline:** A pipeline encapsulates entire data processing task, from start to finish. Includes reading input data, transforming that data and writing output data.
  + **PCollection:** A PCollection is equivalent to RDD of spark. It represents a distributed data set that our beam pipeline operates on.
    - **Immutability:** Pcollections are immutable in nature. Applying a transformations on a pcollection results in creation of new pcollection.
    - **Element type:** The elements in pcollection may be of any type, but all must be of same type.
    - **Operation type:** Pcollection does not support grained operations. We cannot apply transformations on specific elements in pcollection.
    - **Timestamps:** Each element in pcollection has an associated timestamp with it.
    - **Unbounded pcollections:** Source assign the timestamps.
    - **Bounded pcollections:** Every element is set to same timestamp.
    - **Ptransform:** Ptransform represent a data processing operation, or a step in our pipeline. Ex., ParDo, filter, flatten, combine etc.
    - **Pipe** ‘|’ is the operator to apply transforms, and each transform can be optionally supplied with a unique label.
  + **Spark: Bounded (Dataframes**) Unbounded (Dstreams)
  + **Flink: Bounded (Dataset) and Unbounded (Datastream)**

**Transformation functions:**



**Installation:**



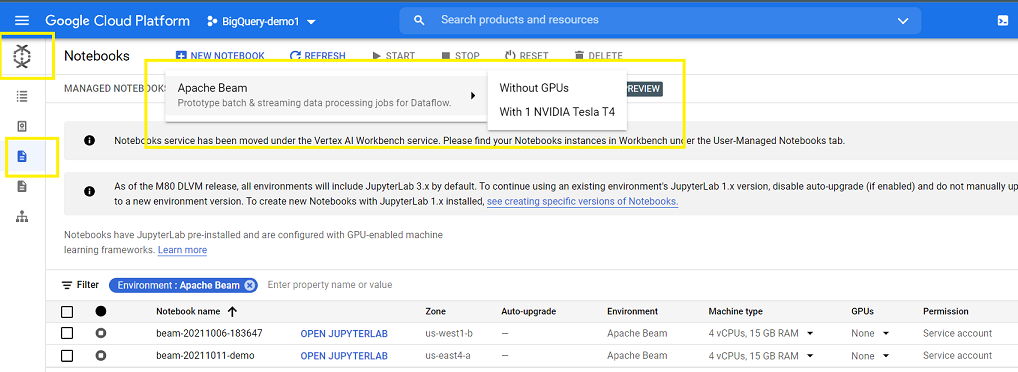
* Using **Cloud** shell (python -m pip install apache-beam)

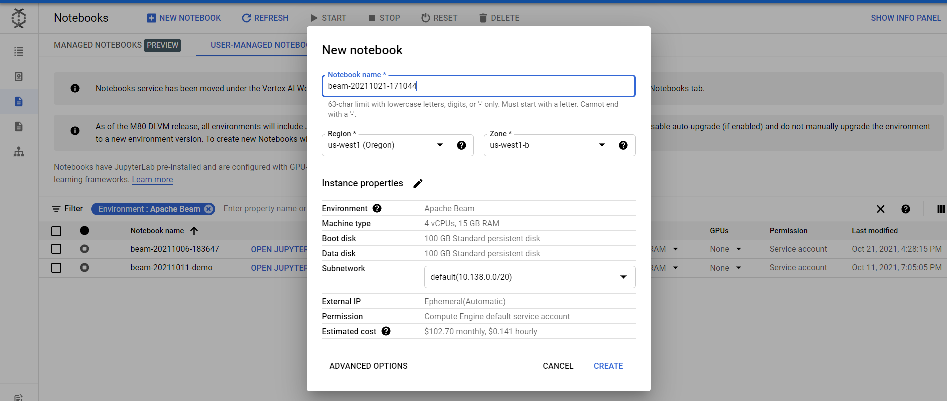
**Services used in GCP:**

* Cloud dataflow
* Big Query
* Google Cloud Storage

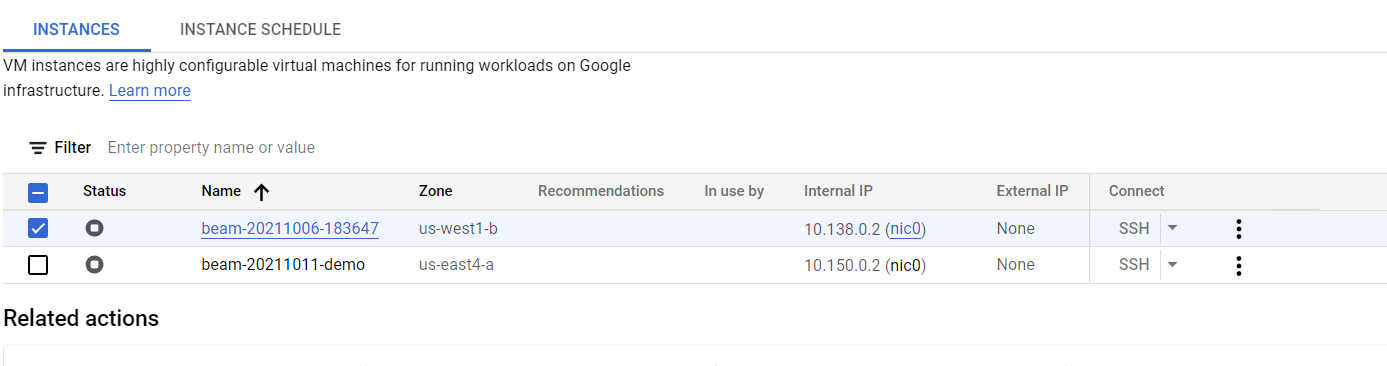
**Steps:**

* Enable API’s for GCS, Cloud data flow and Google big query.
* Open Dataflow. Go to workbench and click on it.
* Create new notebook , from drop down click on Apache beam and select Without GPUs as shown below .

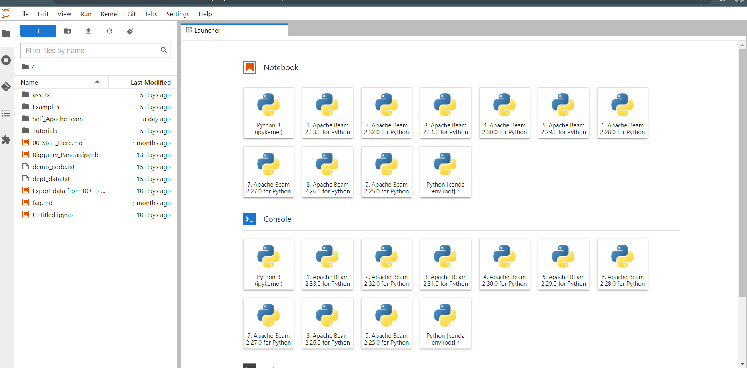




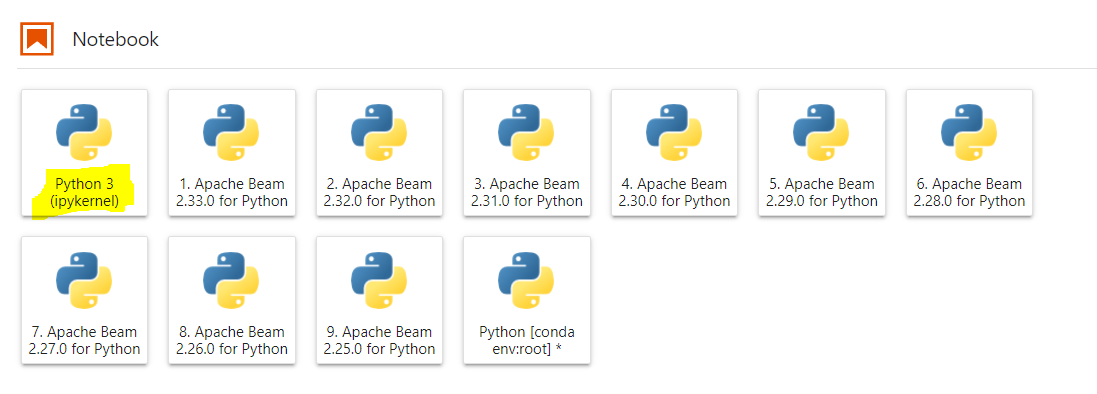
* Click on create and new notebook will be launched along with VM with the same jupyter notebook name running in Compute engine.



* Open Jupyter lab as shown below.



* Select kernel runtime as Python3 ipykernel as shown below.



* Read data from GCS (Google cloud storage) or GBQ (Google Big Query) into apache beam notebook or cloud shell editor for script files.
* Once data is inside apache beam notebooks make transformations to data and export it to either GCS or Big query.

**Technical Representation:**

**Objective:**

1. Reading data from several API’s make some transformation using Apache beam and write it back to BigQuery.
2. Reading data from several API’s make some transformation using Apache beam and write it back to GCS buckets.

For API Ill convert JSOn it to dataframe and save it to GCS bucket or BQ. From there I can read it and perform the transformation.

<https://beam.apache.org/documentation/programming-guide/#pipeline-io>

1. Reading data from Bigquery table make some transformation using Apache beam and write it back to another bigquery table.
2. Reading data from GCS location make some transformation using Apache beam and write it back to bigquery table.
3. Reading data from bigquery table make some transformation using Apache beam and write it back to GCS location.
4. Reading data from GCS make some transformation using Apache beam and write it back to another GCS location.

**GCP Services Involved:**

1. Big Query
2. GCS storage buckets
3. Cloud Dataflow (Apache beam)
4. IAM services
5. Cloud shell & Editor
6. Cloud functions