1. Brief
   1. Implement `get\_parsed\_job\_output(job\_name, raw\_data)` function in script-template file: [etl.py](https://drive.google.com/file/d/1lW8aLdf-3DUZKz9Kc4mjx23EABe2XGWn/view?usp=share_link)
   2. Raw\_data arg is the content of input file; input files can be one of the below files (many more in future) -
      1. [Show-inventory](https://drive.google.com/file/d/1jb0BPxZTeHHeGH5hM_OV_1cANWQaqFl-/view?usp=share_link)
      2. [show version](https://drive.google.com/file/d/1V1QzE5whnEdtBZ4BfbYScA8k2JGENT2V/view?usp=share_link)
      3. [failed-command](https://drive.google.com/file/d/1jLnTqBR4KA3aRJILYyvt6nFZu_CArQa_/view?usp=share_link)
   3. Based on the job\_name, return parsed json object (python dictionary); one of the below outputs is expected -
      1. [Out-inventory.json](https://drive.google.com/file/d/1ItcezVDXgnpeAkHKA5vt2C6fy8Jp-Tvb/view?usp=share_link)
      2. [Out-version.json](https://drive.google.com/file/d/1l2GuDWCemOGNPB-1-G2DFwjUdRdU1qL9/view?usp=share_link)
   4. Each job\_name has its own input raw\_data, parser function, output json.
2. Error handling
   1. If the input arguments (eg: job\_name, raw\_data) is/are in some unexpected format, raise an exception. Please consider raising an exception in the following way (also available in etl.py) -

sys.tracebacklimit = 0 # set sys.tracebacklimit = None to enable it back raise AirflowFailException("invalid job\_name in config.")

* 1. This ensures that there is no stack-trace in logs & Airflow DAG is marked as failed (& prevent downstream tasks from executing) upon an exception
  2. Other scenarios where you might need to raise an exception
     1. If raw\_data contains “failed=1” at the bottom (failed=<non-zero-value> in general)
     2. If raw\_data contains “fatal: [10.0.0.105]: FAILED! => {” instead of “ok: [<some-ip-address>] => {”. Reason for failure can be extracted from “msg” property in raw\_data.