MATILLION

Matillion ETL:
Best Practices To Get Started

Installation

- Immediately after launch, make sure you can SSH on to your Matillion instance
- To update Matillion ETL, create a new Matillion instance and migrate your work from the old instance to the new one. Further details available in How to Update Matillion – Best Practices
- Take regular, automated backups of your metadata via Automated EBS snapshots from the Admin menu, an export through either the GUI or our API, using our Enterprise Git integration feature, or all of the above
- If using a High Availability cluster, make sure your entire workload can be serviced by just one of the two nodes on its own
- If you are relying on any OS customization (e.g. you have added libraries), ensure that you can automate the customizations
- Have a procedure to follow in the event of operational problems (e.g. if you can't access your Matillion server, know who to contact for server administration and networking support)

Design

- Do ELT, not ETL. Load data as-is into the cloud data warehouse (CDW), then transform it within the CDW afterwards
- Don't use Python scripts for data transformations (that's usually a sign that you are doing ETL, not ELT)
- Avoid using iterators over data – Only iterate over metadata
- Avoid iterators when the CDW can do large operations all at once, such as loading multiple files simultaneously
- Follow the best practices of your target cloud data warehouse when building transformations. See our product specific ebooks below:
  - Matillion ETL for Amazon Redshift
  - Matillion ETL for Google BigQuery
  - Matillion ETL for Snowflake
- Avoid manual transactions if you can create idempotent logic. If you must use manual transactions, make them BIG
- Don't hand-write DML statements: use a transformation Job
- Have a data model (i.e. don't create new tables with no organization or standards)
- Plan for a dedicated staging layer in your data warehouse (such as a schema of objects that are always safe to drop or replace)
- Use Transformation Jobs to insert/merge data from the stage into analytics/reporting layers such as ODS, Star, or Cube layers (can be in separate schemas as well)
- Take advantage of Shared Jobs in order to create reusable components
- Within a Transformation Job, only have one write component per target table

- If you need to guarantee the order of write operations, then only have one write component per job and manage the sequence of Transformation execution from the Orchestration that calls those Transformations
- Within a Transformation Job, don't read from the same objects you're writing to (use an intermediary table instead)
- In an Orchestration Job, have exactly one start component
- Handle PII data with great care. If possible, avoid loading PII unless absolutely necessary. If you must load PII (e.g. to perform joins), then hash the PII columns in a consistent way. Consider a quarantine layer of the data warehouse to store PII, one that has extremely limited access privileges.

Implementation

- If you find you are running Python scripts which are large/complex/require non-standard libraries, then run off-instance (discussed in Improving Python Efficiency in Matillion: Offload Large Python Scripts)
- If you want to parameterize a job, first get it working using hardcoded values, then replace those values with variables once it's working
- Always set a default value for any variables you declare. Default values are used during job validation and empty defaults often prevent successful validation
- Use notes to describe your jobs and name components meaningfully
- Access your Matillion instance using https (not http). Note, Matillion comes with a self-signed certificate. Most browsers will raise a warning that the connection isn't private because of this certificate; it is safe to click through this warning since the Matillion instance is a known server that belongs to your company.
- Switch off browser plugins (or use incognito mode) when accessing Matillion
- Use Instance credentials and apply the Principle of Least Privilege to those credentials
- Ensure that development activities don't interfere with production workloads

Converting from Legacy ETL:

- Redesign your ETL jobs so they rely on the CDW's MPP architecture to process very large operations in parallel, rather than on Matillion to do a very large number of small operations. For example:
  - Load multiple files at once by using a prefix/regex load pattern to identify all the files of interest and load them simultaneously
  - Transform data in large operations inside the CDW rather than iterating over multiple, smaller transformations
- Design workflows to leverage concurrent processes and CDW connections
- Don't migrate all of your ETL jobs all at once – Start small. Take an agile approach and redevelop one process, report, or mart at a time. Use this approach as an opportunity to refactor and also demonstrate early wins.
- If you have existing SQL that does the transformation, and it is compatible with the target cloud data warehouse, putting it into SQL Script or SQL components is a quick migration path. Step 2 would then be to decompose the SQL into the related Matillion Transformation components.
- For initial loads on very large tables, pull the data in batches instead of all at once. Design the initial load so the job is restartable and can pick up where it left off.
- Start with a small amounts of cloud computing resources, keeping in mind that you can easily scale up later