ETL Features

This is a list of features of an ETL product that can be adjusted by engineers for any specific use.

Pluggable Protocols

Ability to implement and plug an input/output connector for any data source/target.

Pluggable Creds Store

A Creds Store stores connection credential and other secrets that for security reasons should not be in the source and configuration of any specific pipeline to keep it away from hackers, users, devops and most engineers.

Pluggable Conf Store

Pipeline configurations should live separately from a specific deployment of a pipeline in order to be able to tear down, restore, redeploy etc. pipelines.

Connector Features

This assumes that data can be pulled in batches because some data sources are big. It should be possible to configure:

- Max data size to pull/push in each batch
- Interval between batches
- Retry policy

Pipeline Features

- Ability to pull data in batches
- Each step in a pipeline should have its own executor for faster flow of data
- Scheduling
- Dependencies among pipelines
- Ability to both stream data infinitely and to reload it fully

Monitoring

- Pluggable storage and logging for all steps of a pipeline for centralized monitoring and management

- Each step in a pipeline should persist data (assets) it outputs for the next steps
- Ability to apply logic on top of assets and logging in order to handle errors and notifications

Permissions

- Ability to create users and define permissions in order to
    - allow users access to various groups of pipelines
    - limit users' ability to see data that passes through and control pipelines

Programmatic Access

Ability to manage pipelines programmatically and remotely

Deployments

Provide multiple deployment targets probably through Kubernetes

Evaluated Products

dagster; pulsar; spark, flink; stitch, meltano, airbyte; prefect, airflow; aws batch + step functions; fivetran, domo, hevo, trifacta; streamsets;

[recently discovered Nexla, need to evaluate]

Winner: Dagster

- The goal of Dagster is DAG
- It is possible to build ETL on top of Dagster
- Dagster is complex

It is possible to use Dagster to build a better ETL tool but one must take into account that Dagster is a complex product.

Prefect is close to Dagster but in comparison lacks features that make coding in Dagster more flexible.

Regarding **Airflow**, Prefect was created to address Airflow deficiencies.

Proposal

- Build an ETL product on top of Dagster with simple deployment, configuration, programmatic access, UI
- Provide it as a SAS and as an in-house product
- Create a repo of connectors and other reusable components so that all companies that use it would share the benefits

Comparisons

List of core issues that cannot be remedies due to the architecture of the products.
This list does not mention all deficiencies of each product but only the most outstanding ones.

Fivetran

- SAS, therefore this is a security concern for credentials and data.
- No way to create dependencies among pipelines.

Domo

- SAS, therefore this is a security concern for credentials and data.
- Custom connectors can be built to HTTPs only.

Hevo

- SAS, therefore this is a security concern for credentials and data.
- No way to create dependencies among pipelines.

Trifacta

- Does not support custom connectors

Airbyte

- SQL transformations only
- Needs an external product such as Dagster to create dependencies among pipelines

Meltano

- SQL transformations only
- Needs an external product such as Airflow to create dependencies among pipelines

Stitch

- SAS, therefore this is a security concern for credentials and data.
- It only allows to specify a source + frequency out of a list of predefined values + multiple outputs. There is no scheduling, no transformations, no dependencies.
- No way to add a custom connector

StreamSets

- When StreamSets was evaluated, it was not reliable

Spark

- Requires an external scheduler such as Dagster.
- Designed to work with its in-memory data whereas an ETL tool needs to be able to run pieces of code and queries without loading data into its memory.
- No branching and dependencies

Flink is essentially the same as Spark

Pulsar

- Requires external scheduler and external management of pipelines

AWS Step Functions

Step Functions must be combined with other services such as Cloud Watch for scheduling, SQS for creating dependencies etc. This logic can be expressed in code, for example CDK + Python or Python with boto3. Overall this is no different than using multiple AWS services as building blocks. On the one hand this is modular and scalable, on the other hand this is complex. Better to use a single framework like Dagster.
It can only run in AWS.