sagetasks: a Python package for data and workflow orchestration in the cloud

Bruno M. Grande, Tess M. Thyer, James A. Eddy, Thomas V. Yu, Brian D. O'Connor

SageBionetworks
We're hiring!

Motivation

Launching workflows often involves multiple steps, such as:

1. Preparing workflow parameters from other (meta)data
2. Transferring input data files and workflow output files
3. Chaining multiple community-curated workflows

Including these additional tasks in the workflows would limit their portability and usefulness for the community.

Hence, there is a need for tooling that can orchestrate these high-level extract-transform-load (ETL) pipelines.

Proposed Solution

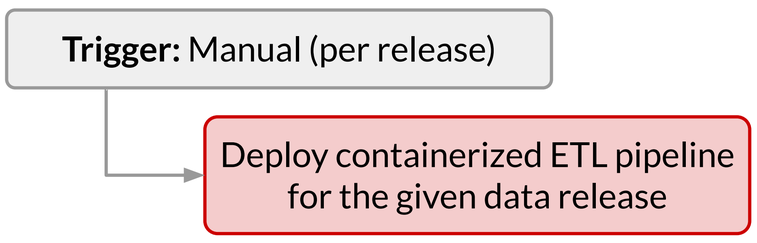
The proliferation of workflow execution platforms and APIs has opened up the opportunity for remote orchestration.

We are developing the sagetasks Python package as a collection of reusable functions for moving and processing data on platforms such as Nextflow Tower and Cavatica.

These functions could be leveraged by general-purpose workflow management systems, providing out-of-the-box user interfaces, monitoring, scheduling, and logging.

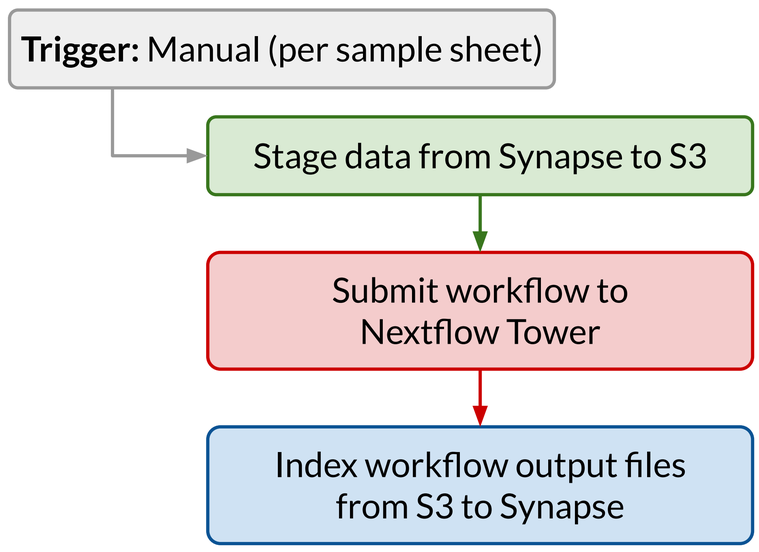
Example ETL #1: Low complexity

Fully containerized Python ETL pipeline



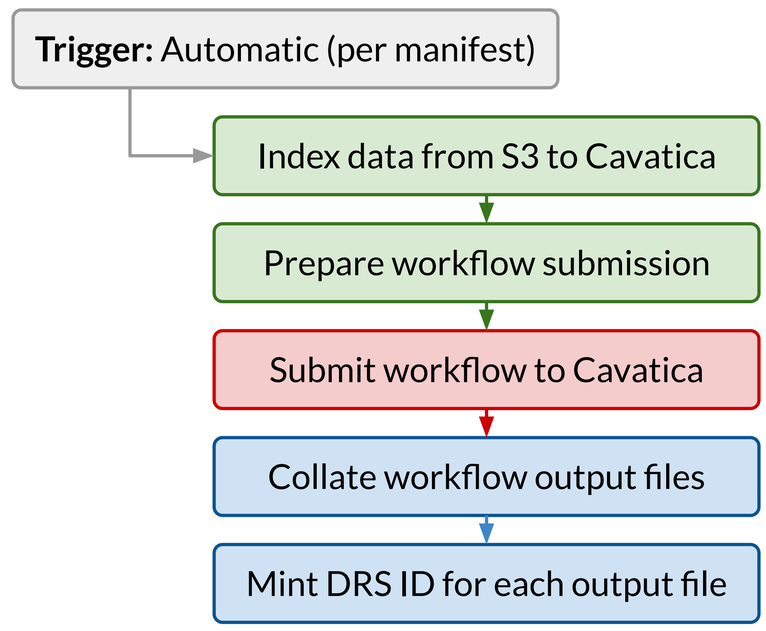
Example ETL #2: Medium complexity

Round trip from Synapse to Nextflow Tower



Example ETL #3: High complexity

Data processing and DRS ID minting on Cavatica



Color Legend

- Extract step
- Transform step
- Load step

Current Challenges

- **Selecting a workflow management system:** This process proved to be more challenging than anticipated. While most options meet our requirements on paper, they each have limitations that are not obvious at first glance.
- **Avoiding duplicate workflow runs:** Given that HTTP POST requests are not idempotent, special care is needed when submitting runs to avoid unnecessary costs. For example, the client can query existing runs before submission. Additionally, APIs need to support any caching offered by the backend execution engines.
- **Supporting custom logic for preparing API requests:** While it's possible to standardize workflow execution APIs, the client must allow users to specify how to generate the workflow inputs since organizations and projects organize their metadata in distinct ways.

Workflow Management Systems – Feature Comparison

Features (requirements in bold)	Airflow	Prefect	Dagster	Flyte
Scheduled pipelines	Yes	Yes	Yes	Yes
Manually triggered pipelines	Yes	Yes	Yes	Yes
Python support	Yes	Yes	Yes	Yes
Self-hosted option	Yes	Yes	Yes	Yes
Runtime parameters	Yes	Yes	Yes	Yes
Dynamic workflows	Yes	Yes	Partly	Yes
Secrets	Yes	Yes	Yes	Partly
Caching/memoization	No	Yes	Yes	Yes
Web user interface	Yes	Yes	Yes	Yes
Command-line interface	Yes	Yes	Yes	Yes
Nested pipelines	Yes	Yes	Yes	Yes
Container orchestration	Yes	No	Yes	Yes
AWS connectors	Yes	Partly	Partly	Partly
Static DAG visualization	Yes	No	Yes	Yes
Product maturity	Yes	Partly	Partly	Partly
Cross-pipeline triggers	Yes	Partly	Yes	Partly

References and Links

This poster has a FigShare DOI, which includes references and links. Access it using the QR code beside the poster title.

Acknowledgments

The HTAN (U24-CA233243) and INCLUDE (U2C-HL156291) NIH programs fund this work and associated travel.