

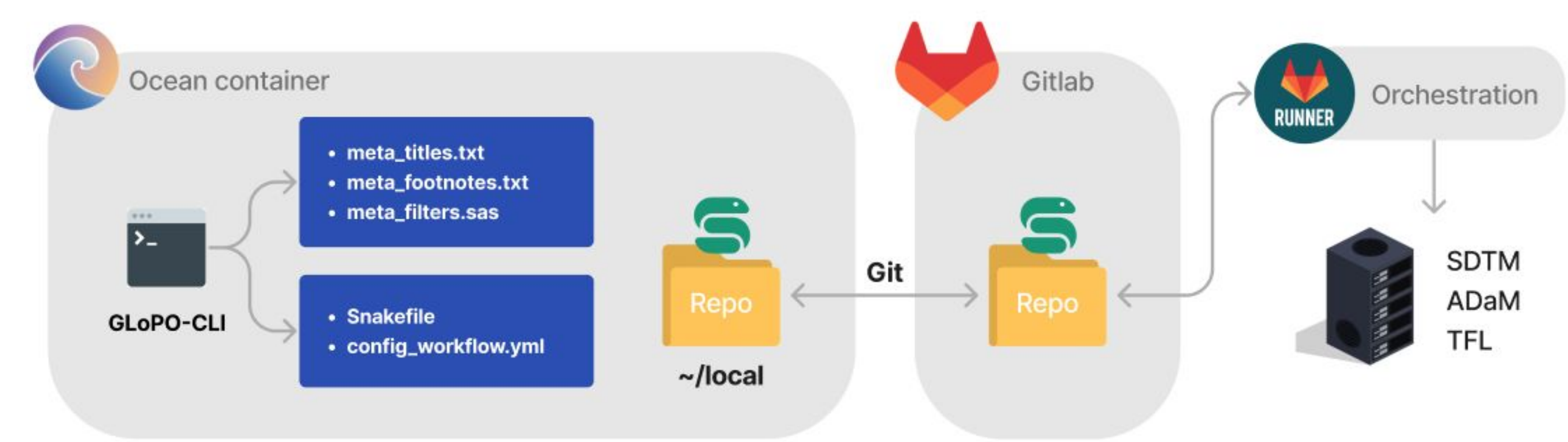
PP30 - Snakemake for Statistical Programmers

Automation and Reproducibility in Clinical Studies



Mathieu Cayssol (mathieu.cayssol@roche.com) • Nassim Sleiman (nassim.sleiman@roche.com)
Hoffmann-La Roche, Basel, Switzerland

Clinical Study Workflow



GLoPO-CLI: command line interface that extracts List of Planned Outputs content to generate required metafiles and Snakemake files (*Snakefile*, *config_workflow.yml*).
Container: files updates within a container (eg: RStudio, VS Code). Local and remote development and execution.
Gitlab: stores code and reporting activity metadata files.
Orchestration: execution of the Snakemake workflow. Produces the SDTM, ADaM, and TFL outputs in a **consistent, reproducible, and auditable** manner.

Snakemake

Snakemake is a workflow management system for creating reproducible, scalable data analyses, described in a readable, Python-based language. A single workflow definition can run on a laptop and scale to servers, clusters, grids, or cloud environments without changing the workflow code. Analyses are specified as rules in a *Snakefile* that map outputs to inputs. A shell command is specific to generate the specified outputs based on the specified inputs. From these rules, Snakemake infers a directed acyclic graph (DAG) of jobs and executes it with automatic parallelization.

```
# SDTM rules
rule dm:
    output: "data/sdtm/dm.parquet"
    shell: "Rscript programs/sdtm/create_dm.R"

rule ex:
    output: "data/sdtm/ex.parquet"
    shell: "Rscript programs/sdtm/create_ex.R"

rule ds:
    output: "data/sdtm/ds.parquet"
    shell: "Rscript programs/sdtm/create_ds.R"

rule ae:
    output: "data/sdtm/ae.parquet"
    shell: "Rscript programs/sdtm/create_ae.R"

# ADaM rules
rule adsl:
    input:
        dm="data/sdtm/dm.parquet",
        ex="data/sdtm/ex.parquet",
        ds="data/sdtm/ds.parquet",
        ae="data/sdtm/ae.parquet"
    output: "data/adam/adsl.parquet"
    shell: "Rscript programs/adam/ad_adsl.R"

rule adae:
    input:
        ae="data/sdtm/ae.parquet",
        ex="data/sdtm/ex.parquet",
        adsl="data/adam/adsl.parquet"
    output: "data/adam/adae.parquet"
    shell: "Rscript programs/adam/ad_adae.R"

# TLG rules
rule t_dm_SE:
    input:
        adsl="data/adam/adsl.parquet"
    output: "data/tlg/t_dm_SE.out"
    shell: "Rscript programs/sdtm/t_dm.R --filter SE"

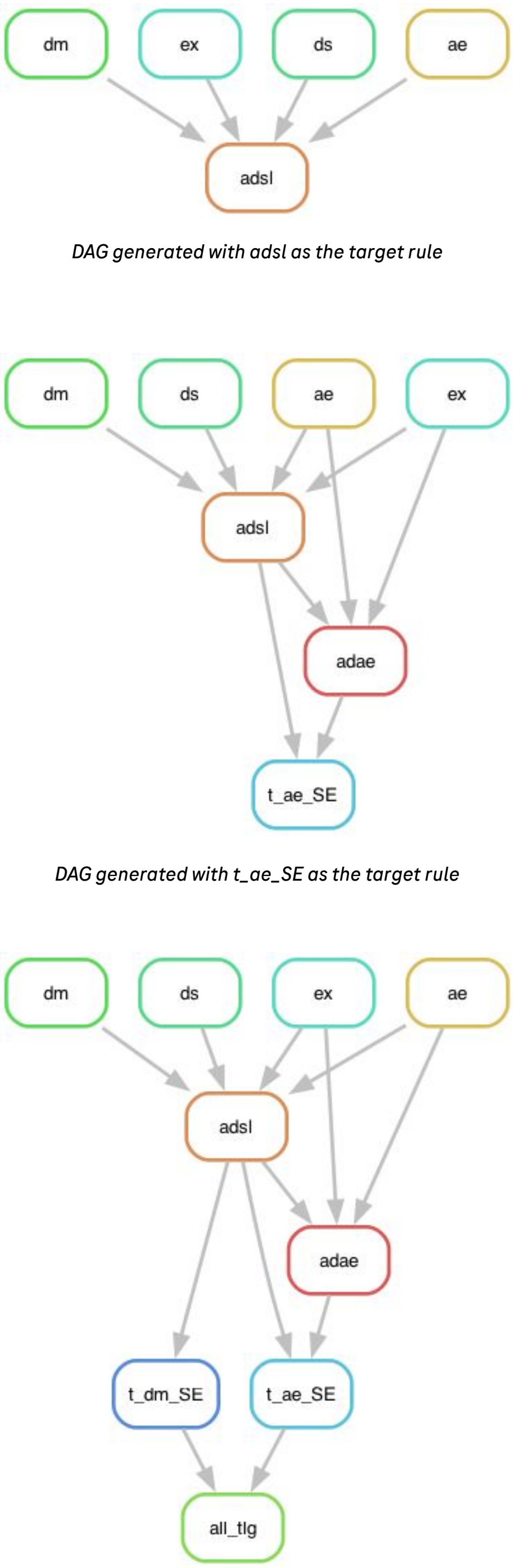
rule t_ae_SE:
    input:
        adae="data/adam/adae.parquet",
        ae="data/sdtm/ae.parquet"
    output: "data/tlg/t_ae_SE.out"
    shell: "Rscript programs/sdtm/t_ae.R --filter SE"

# All SDTM, ADaM and TLG rules
rule all_sdtm:
    input:
        "data/sdtm/dm.parquet",
        "data/sdtm/ex.parquet",
        "data/sdtm/ds.parquet",
        "data/sdtm/ae.parquet"

rule all_adam:
    input:
        "data/adam/adsl.parquet",
        "data/adam/adae.parquet"

rule all_tlg:
    input:
        "data/tlg/t_dm_SE.out",
        "data/tlg/t_ae_SE.out"
```

Snakefile

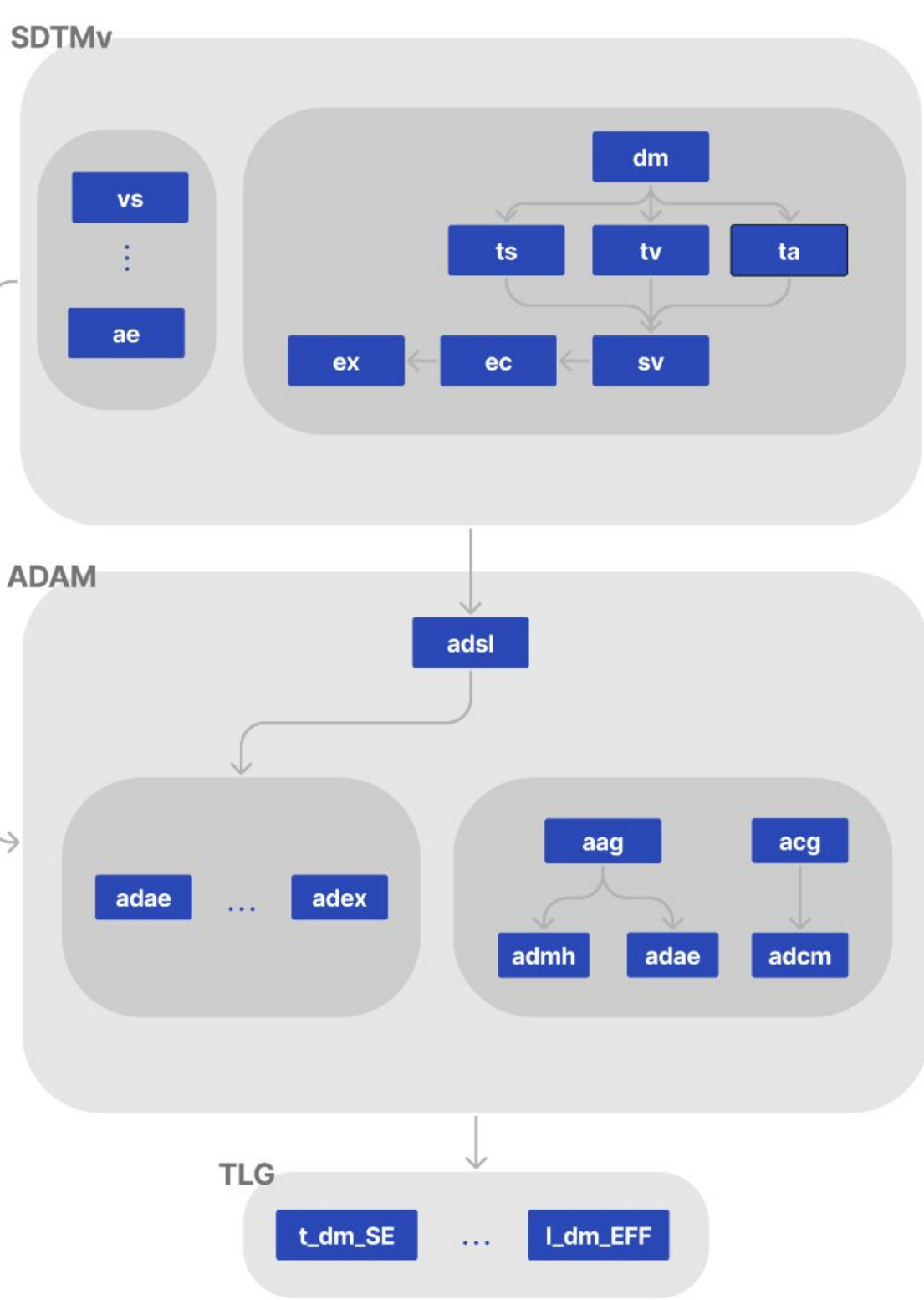


Example of Snakefile and its corresponding DAGs

Snakemake automatically tracks which files depend on which inputs, ensuring that only affected steps are re-executed when data or code change. This selective re-run capability saves time and reduces errors. It also logs every command, input, and output, providing a complete audit trail.

Implementation at Roche

At Roche, the workflow is driven by two key components: the Snakefile and the *config_workflow.yml*. The Snakefile defines one rule per SDTM domain, ADaM dataset, and TLG, specifying how outputs are generated from their inputs. The *config_workflow.yml* holds all study-specific parameters (paths, dependencies) so that programmers can adapt the workflow without touching the core code. This structure separates logic from configuration: the Snakefile manages process flow, while the YAML file customizes it for each study. Dependencies are automatically inferred by Snakemake but can be overridden through the configuration.



```
# config_workflow.yml

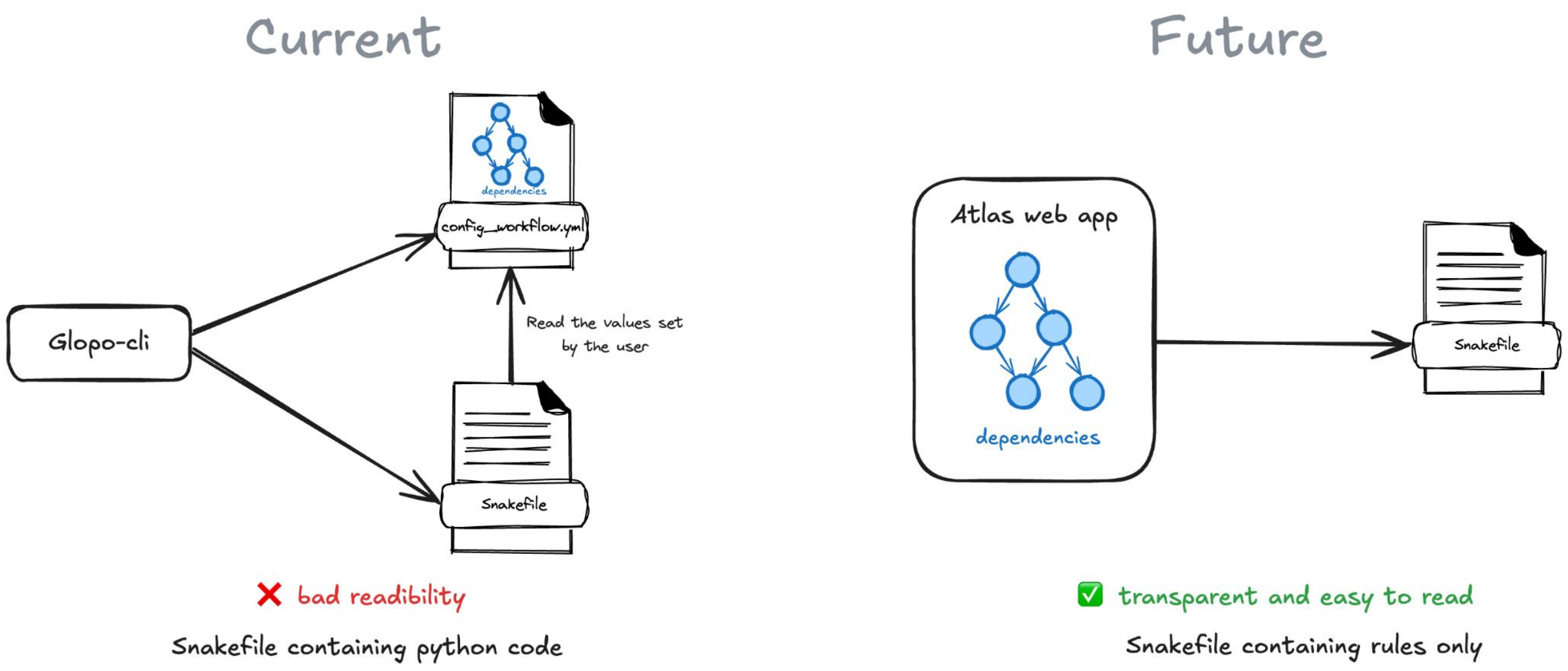
SDTMv:
  ts:
    inputs:
      - dm
  tv:
    inputs:
      - dm
  ta:
    inputs:
      - dm
  sv:
    inputs:
      - dm
      - ts
      - tv
      - ta

ADaM:
  aag:
    inputs:
      - None
  acg:
    inputs:
      - None
  adae:
    inputs:
      - aag
      - adsl
  admh:
    inputs:
      - aag
      - adsl
  adom:
    inputs:
      - acg
      - adsl
```

Default set of dependencies defined in config_workflow.yml

Advantages		Challenges
Reproducibility & Traceability	Guarantees a fully transparent audit trail for every output, ensuring results are verifiable and compliant with regulatory standards.	The strict enforcement of traceability can be rigid; even a tiny input change forces a full re-run of all downstream tasks, impacting efficiency.
Automation & Dependency Management	Automatically manages complex dependencies and runs tasks in parallel, which minimizes manual errors and dramatically reduces overall processing time.	A dependency on a single centralized file (e.g., footnotes) creates a bottleneck; updating it can trigger a massive and unnecessary re-run of all outputs.
Standardization via Templates	Provides a consistent, reusable framework that accelerates study setup, simplifies programmer onboarding, and ensures uniformity across projects.	Standard templates struggle to accommodate unique study "edge cases," often requiring complex manual overrides that add complexity and maintenance burden.
Workflow Abstraction	Separates core logic from study-specific configuration, empowering statistical programmers to easily adapt the pipeline without needing deep coding expertise.	This separation can make the workflow opaque; when errors occur, it can be difficult to trace their root cause through the layers of abstraction.
Scalability & Portability	The same workflow definition can be executed seamlessly on a local machine, a high-performance cluster, or a cloud environment without modification.	While the workflow can run on any scale, its efficiency can be poor if the logic constantly re-runs large, unaffected portions of the pipeline.

Future implementation



Our future implementation will address manual dependency management and workflow abstraction by replacing the current interdependent structure. Instead of dynamically linking the Snakefile to the configuration file, dependencies and study settings will be managed through a web app (Atlas). Statistical programmers will define study metadata, dataset dependencies, and analysis options there, and the system will generate a fully explicit Snakefile with clear, non-abstrated rules.

Contributions & thanks

GLoPO-CLI team: Daphne Grasselly, Craig Gower-Page, Mahdi About, and Helene Royo. Atlas team, including Daphne Grasselly, Haoyang Ju, Franciszek Walkowiak, Dinakar Kulkarni, Eddy Foster, and Rohan Parmar, for their dedicated efforts. Onboarding team on OCEAN: Helene Royo, Christof Binder, Valeria Visona, and Ivan Robinson.