Beyond the bundle
AIP-85+

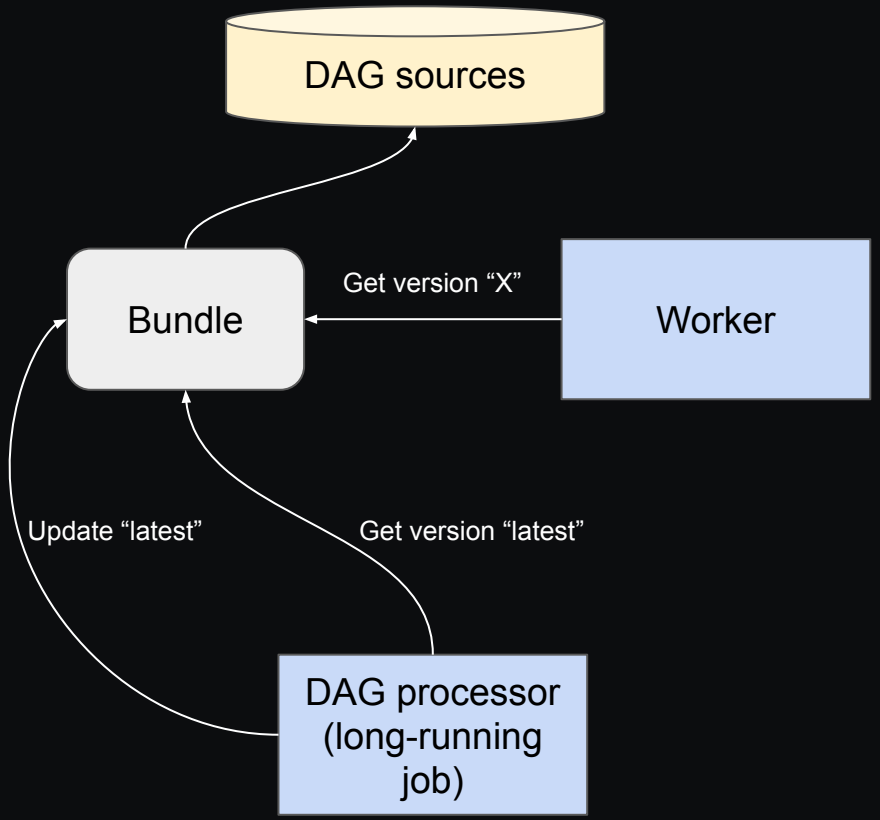
3.0

Igor Kholopov
Cloud Composer, Google

Bundles in Airflow 3

Bundles (introduced in AIP-66 by @jedcunningham):

- Consistency of version within a DAG run
- Customizable refresh of the latest version
- Many-to-one bundles to DAG processor support
- Operates within existing Airflow “eventual consistency” paradigm with source of truth being DAG sources storage (Git repo, FS, S3, etc.)

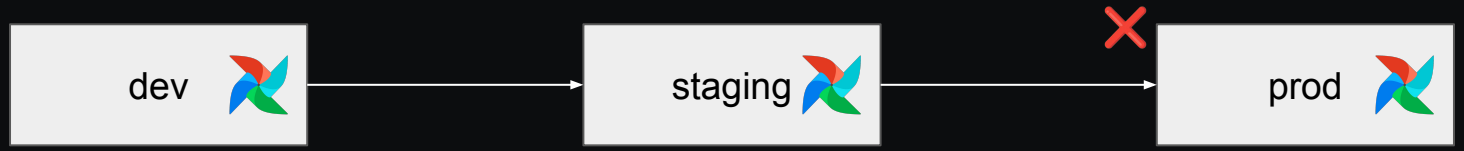


A case for “strong consistency” in DAG processing

- Feature of “continuous processing” can easily become a bug:
  - Transient DAG processing errors can affect effective DAG availability
  - AIP-92 introduces additional hops in DAG processing path => higher availability requirements for hops to maintain the same DAG availability

```
1.  @dag(start_date=datetime.datetime(2021, 1, 1), schedule="@hourly")
2.  def generate_dag():
3.      users_list = external_99_availability_service.fetch_users()
4.      for u in users_list:
5.          PythonOperator(task_id=f'per_user_task_{u}',
6.              python_callable=do_something_for(u))
```

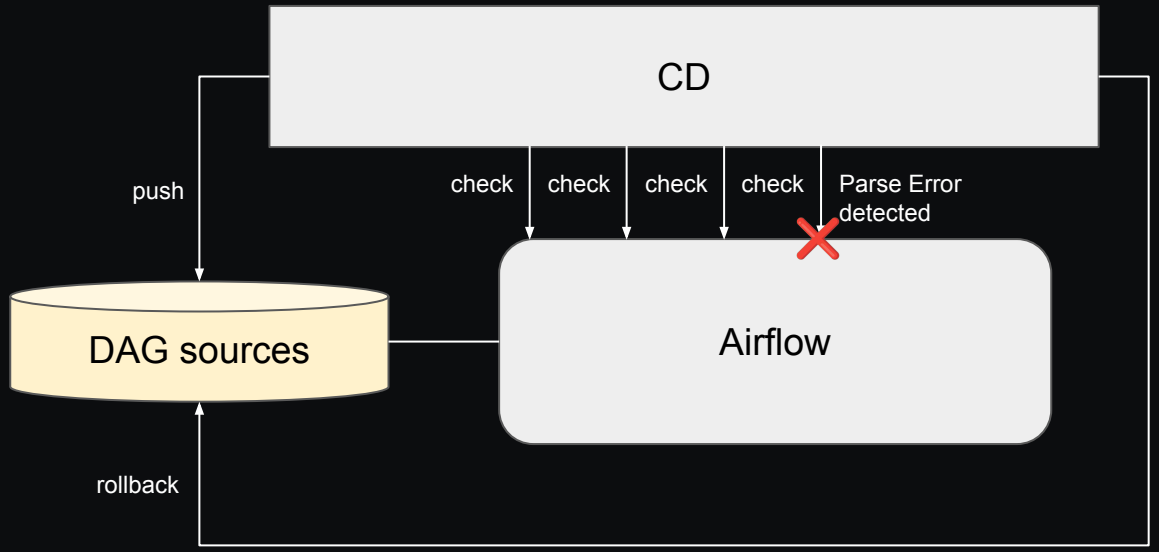
A case for “strong consistency” in DAG processing



- A classic deployment between different “environments” works well up to a certain DAG codebase size
- A more reliable approach is introduction of canarying

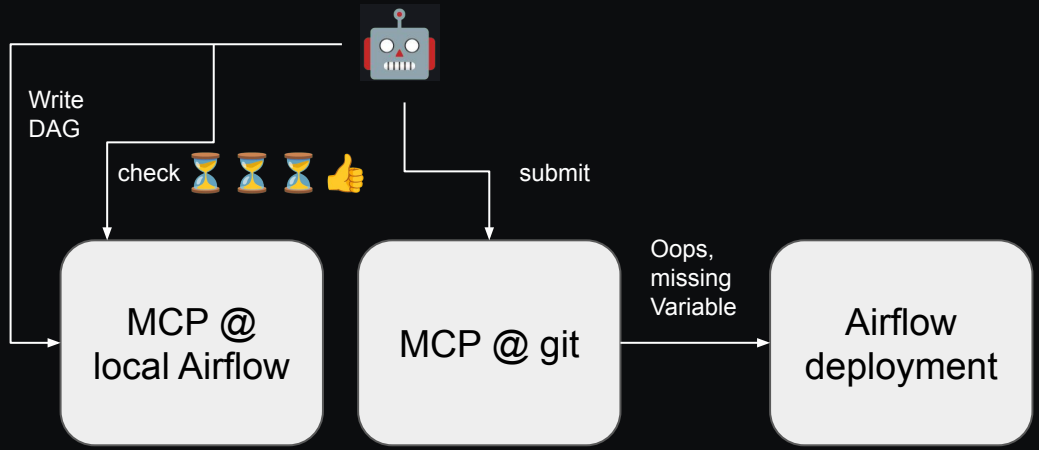
A case for “strong consistency” in DAG processing

- Safe continuous delivery in Airflow requires offloading a lot of complexity to CI/CD implementation
- Multiple asynchronous checks for all DAGs to be re-parsed

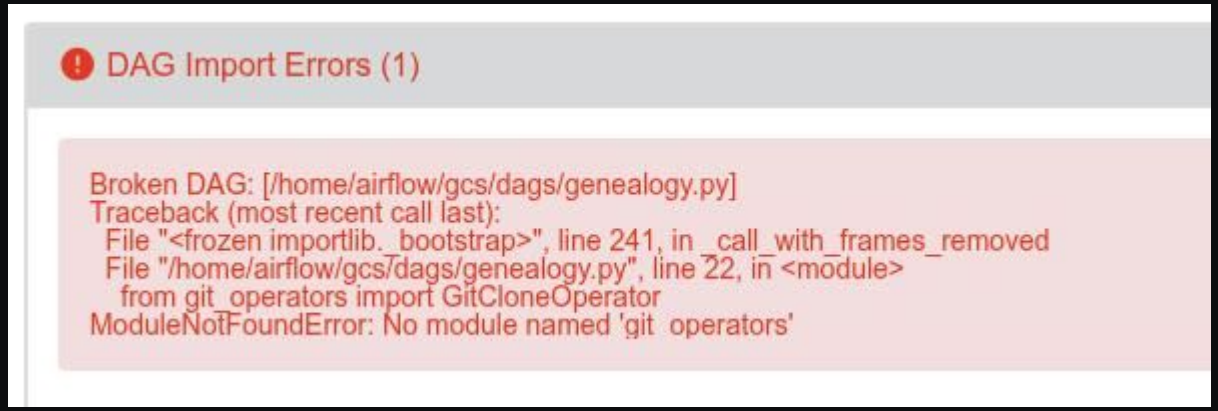


A case for “strong consistency” in DAG processing

- AI agents need a reliable and a fast way to evaluate authored DAGs
- Preferably in the environment with the final target context

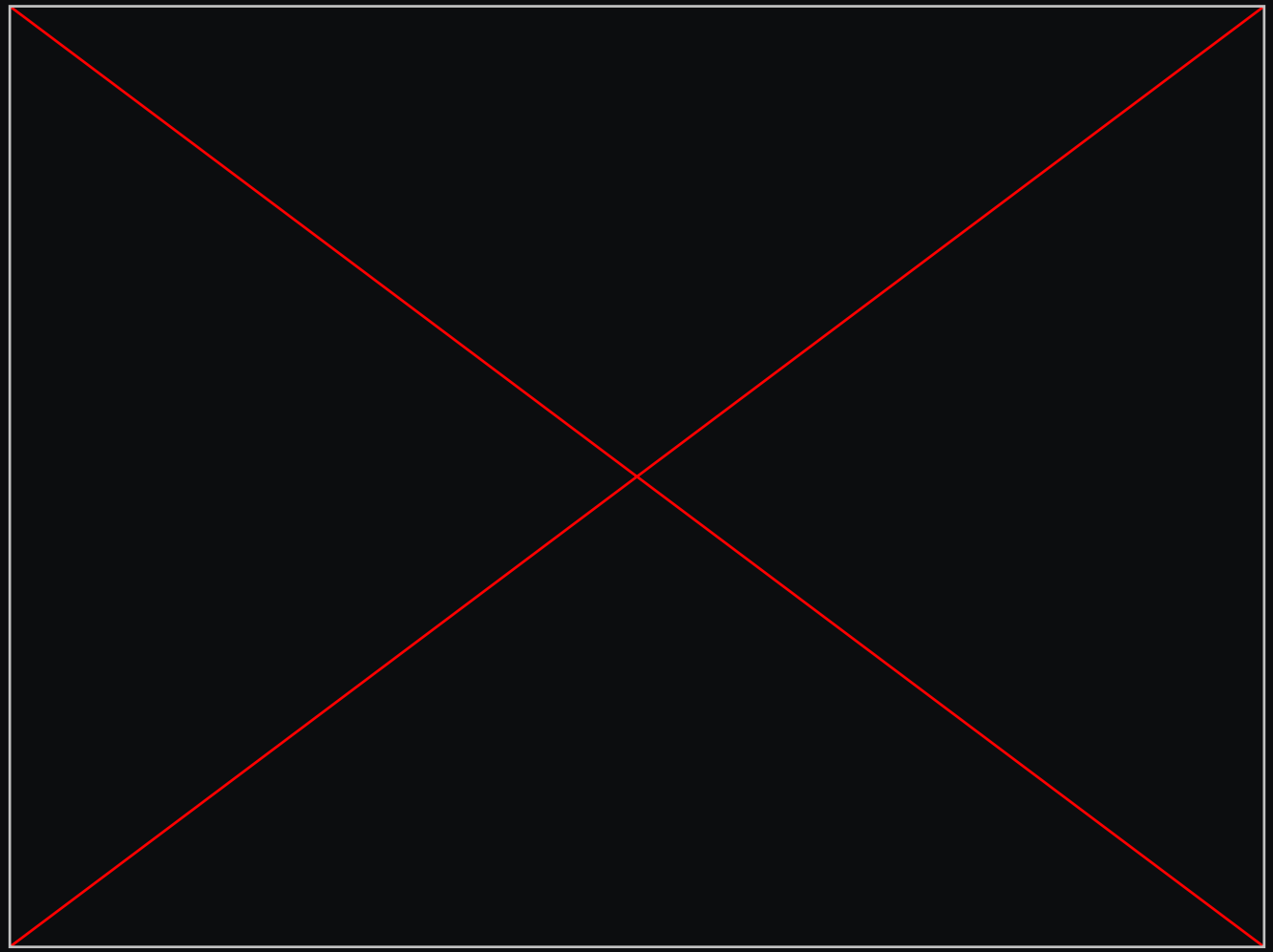


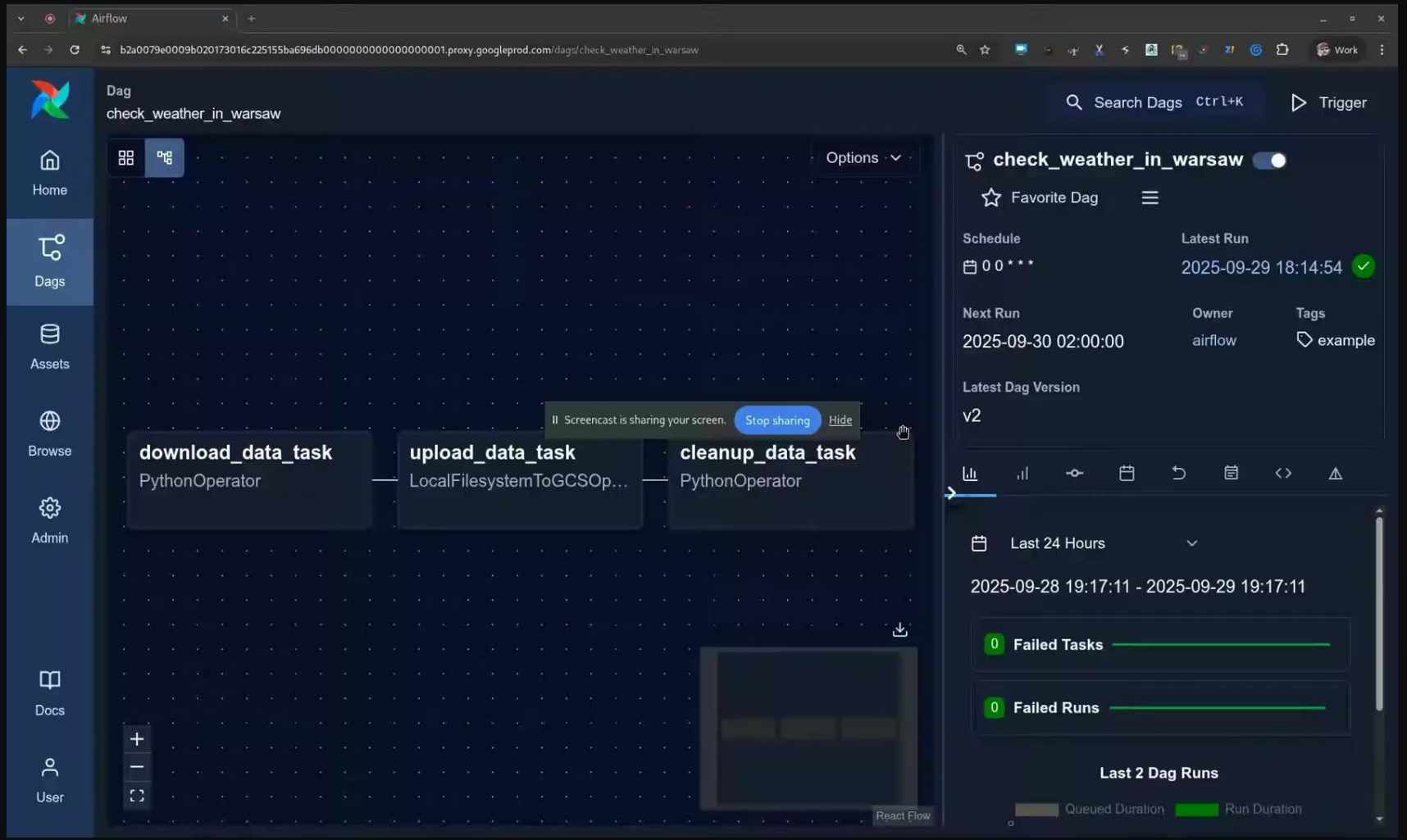
A case for “strong consistency” in DAG processing



Phase 1 Demo

3.0

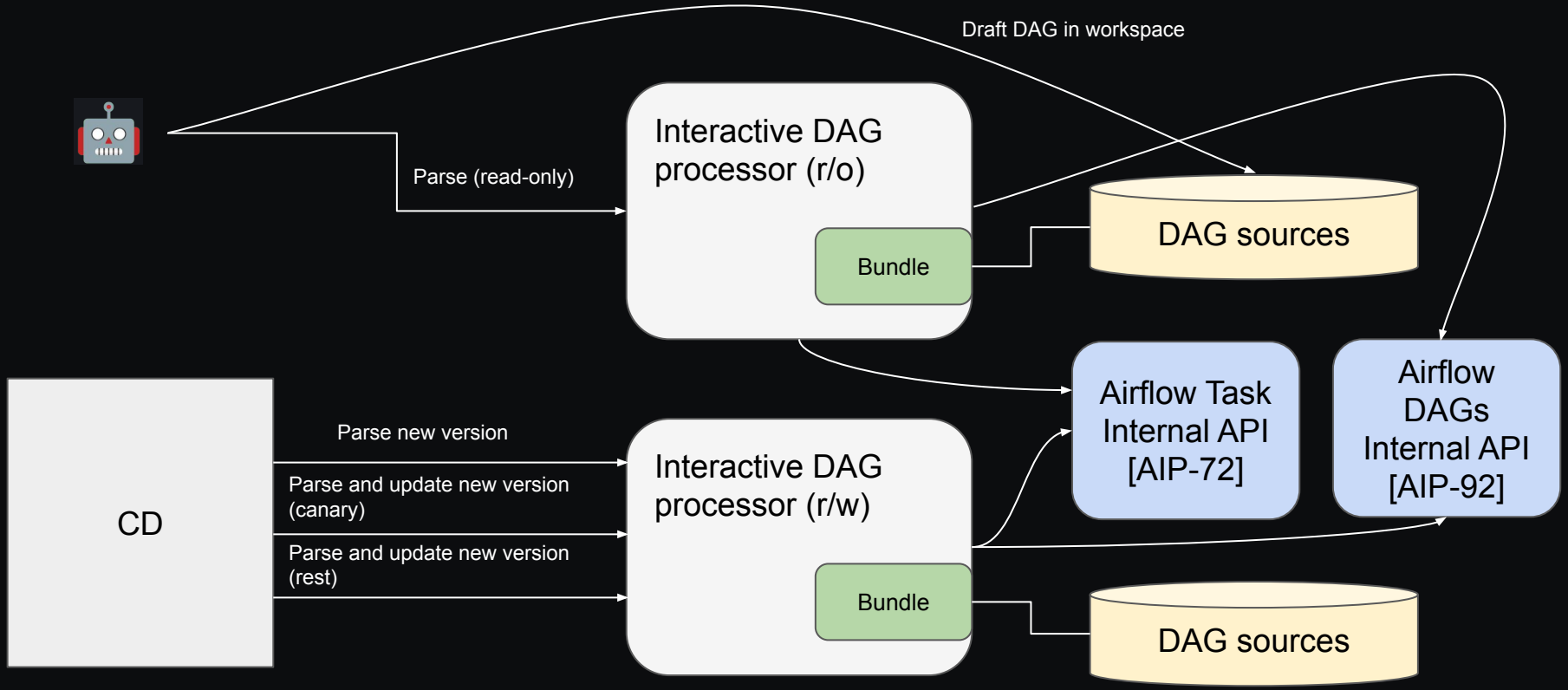



Airflow
b2a0079e0009b020173016c225155ba696db000000000000000001.proxy.googleprod.com/dags/check_weather_in_warsaw
Work
Dag
check_weather_in_warsaw
Search Dags Ctrl+K
Trigger
Home
Dags
Assets
Browse
Admin
Docs
User
Options
Screencast is sharing your screen.
Stop sharing
Hide
download_data_task
PythonOperator
upload_data_task
LocalFilesystemToGCSOp...
cleanup_data_task
PythonOperator
React Flow
check_weather_in_warsaw
Favorite Dag
Schedule
0 0 * * *
Latest Run
2025-09-29 18:14:54
Next Run
2025-09-30 02:00:00
Owner
airflow
Tags
example
Latest Dag Version
v2
Last 24 Hours
2025-09-28 19:17:11 - 2025-09-29 19:17:11
0 Failed Tasks
0 Failed Runs
Last 2 Dag Runs
Queued Duration
Run Duration

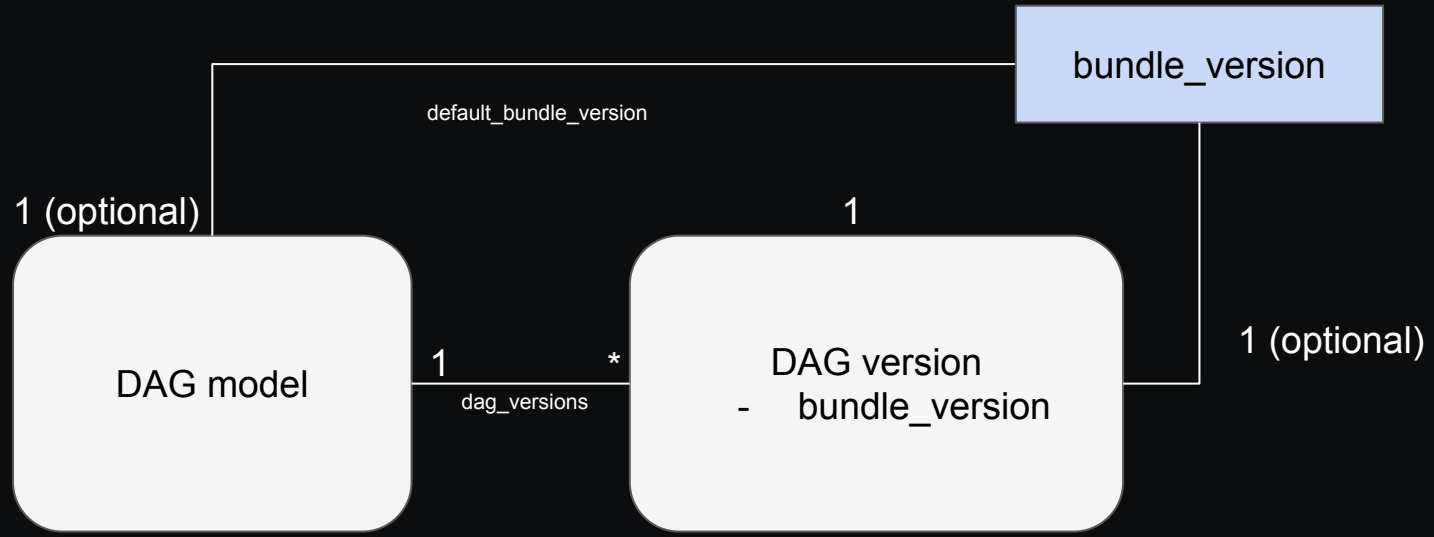
Phase 1: Interactive DAG processor

- POST `/bundle/parse` { “bundle”, “path”, “version” } -> { “import_errors”, “dags” }
- POST `/bundle/parse_update` { “bundle”, “path”, “version” } -> { “import_errors”, “dags”, “version” }
- DELETE `/bundle` {“bundle”, “path”, “dag_ids” }
- GET `/bundle` -> { “bundle”, “version” }
- POST `/bundle/set_version` { “bundle”, “dag_ids”, “version” }

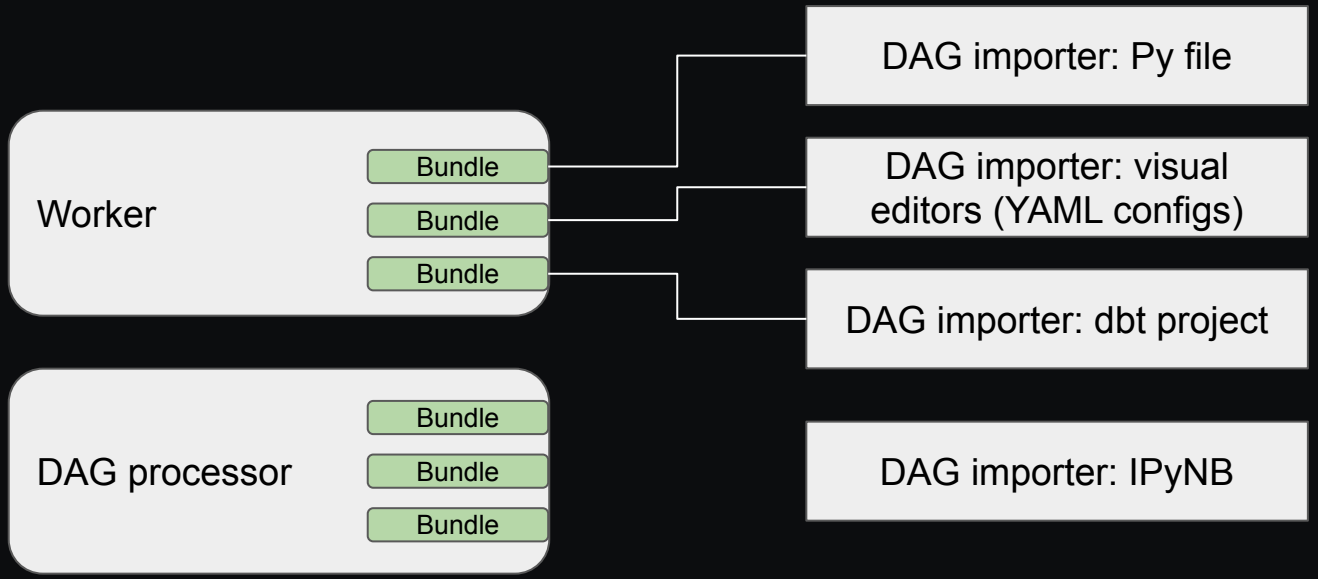
Phase 1: Interactive DAG processor



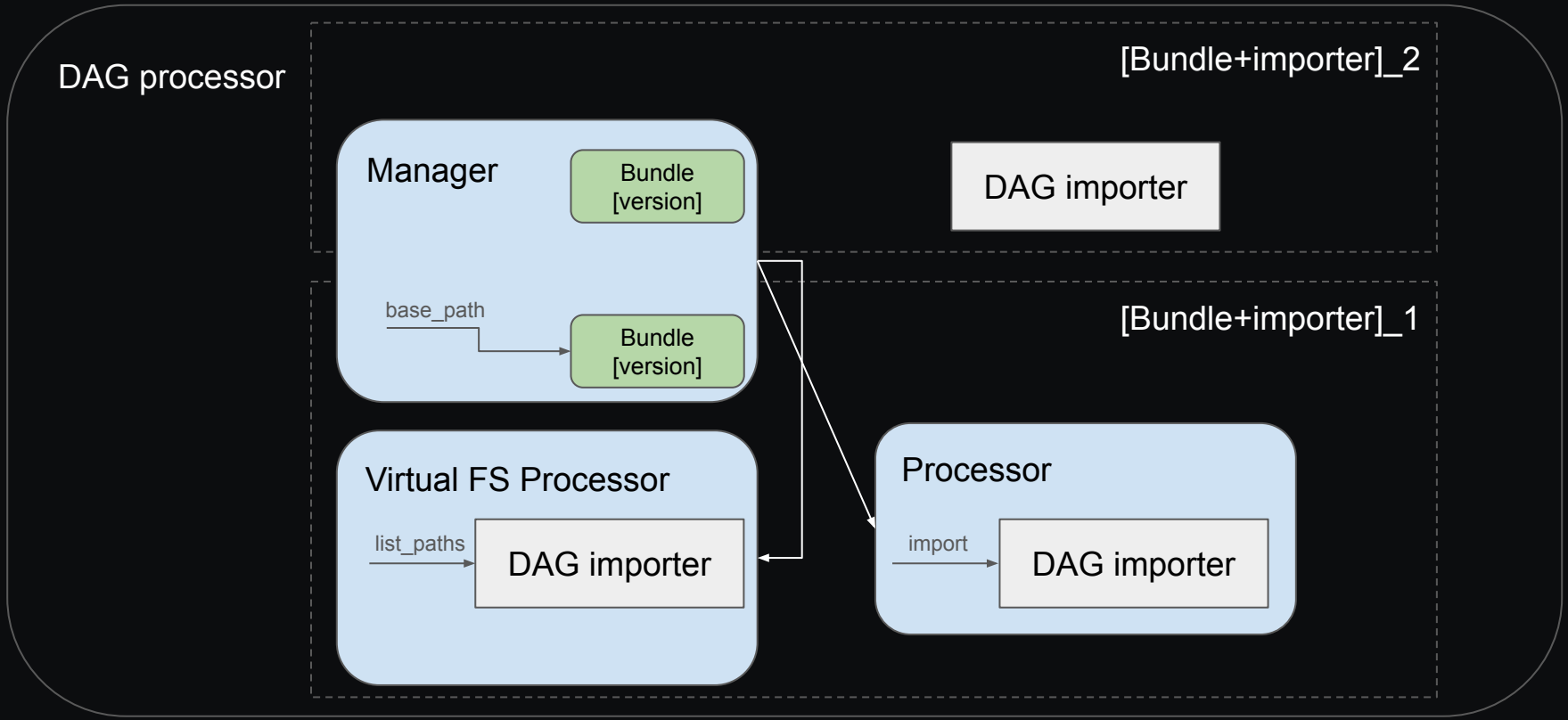
Model extension



Phase 2: Abstract DAG importer



DAG importer - bundle interaction



DAG importer interface

- `import_path`(path, options) -> Iterable[DagsImportResult]
- `list_paths`(subpath) -> Iterable[Path]
- `dag_path_exists`(dagpath) -> bool
- `modified_time`(dagpath) -> timestamp

Split of responsibility

DAG processor	DAG bundle	DAG importer
- What DAGs are ingested - When DAGs are ingested - Airflow metadata updates (through API after AIP-92)	- DAG sources storage management (as a “bundle” unit) - DAG sources version control (as a “bundle” unit)	- How the DAG is constructed from sources - What DAG paths exist in the sources within a bundle

Phase 2 Demo

3.0

Google Cloud | ikholopov-dag-api-plg | Search (/) for resources, docs, products, and more | Search

Explorer | + Add data

Search for resources

- Home
- Starred
- Job history
- ikholopov-dag-api-plg
  - Datasets
  - External connections
  - Queries
  - (Classic) Queries
  - Notebooks
  - Data canvases
  - Data preparations
  - Pipelines
  - Repositories
- bigquery-public-data
- concord-prod

demo-... ne1

demo-pipeline1 | Run | Schedule | Share | Ask Agent | Compiled

Pipeline | Executions | Settings

Add task | 35%

Query updated

Show debug panel

DAG bundle x DAG importer compatibility

FS bundles - git - LocalFS - S3	File importers - Local FS Python - Local FS Notebook - BQ pipeline YAML-format
Synthetic bundles (virtual path, pass version to importer)	External importers (no files to import, produce “paths” and DAGs directly) - BQ Pipelines importer

Example: BQ pipeline (unversioned-online)

Synthetic FS Bundle

path = `/fixed/mnt-point`
dag_importer=`BQPipelinesImporter`

BQ Pipelines Importer

- list_paths ->
  - `/fixed/mnt-point/pipeline_a`
  - `/fixed/mnt-point/pipeline_b`
- import_path ->
  - rpc://Dataform.CreateCompilationResult
  - rpc://Dataform.QueryCompilationResultAction
  - Translate BQ Action -> Airflow task

Example: BQ pipeline (versioned-local)

git Bundle

path = `/checkedout/version`
dag_importer=`YAMLPipelinesImporter`

Local BQ Pipelines Importer

- list_paths -> os.listdir()
- import_path ->
  - Run local compilation of YAML
  - Translate BQ Action -> Airflow task

Questions?

Email: ikholopov@google.com
Airflow Slack: Igor Kholopov
AIP-85: wiki
Dev list thread:
tinyurl.com/aip-85-dev-thread