Naya College

Apache Airflow

Agenda

Part 1

- History
- Introduction to Airflow
- Why Airflow?
- Airflow Architecture
- Airflow Components
- User Interface

Part 2

- Airflow Workflow in Details
- Creating an Airflow Workflow Methods

Part 3

- Running a Job Using Airflow CLI
- Creating & Running DAGs in Airflow Workflow
- Viewing & Troubleshooting DAGs Airflow Workflows

History

Airbnb is a fast growing, data informed company.
The data teams and data volume are growing quickly, and accordingly, so does the complexity of the challenges we take on.

Airbnb growing workforce of data engineers, data scientists and analysts are using **Airflow**, a platform we built to allow us to move fast, keep our momentum as we author, monitor and retrofit **data pipelines**.

Today, **Airflow** is **open sourcing** and **sharing Airflow**.

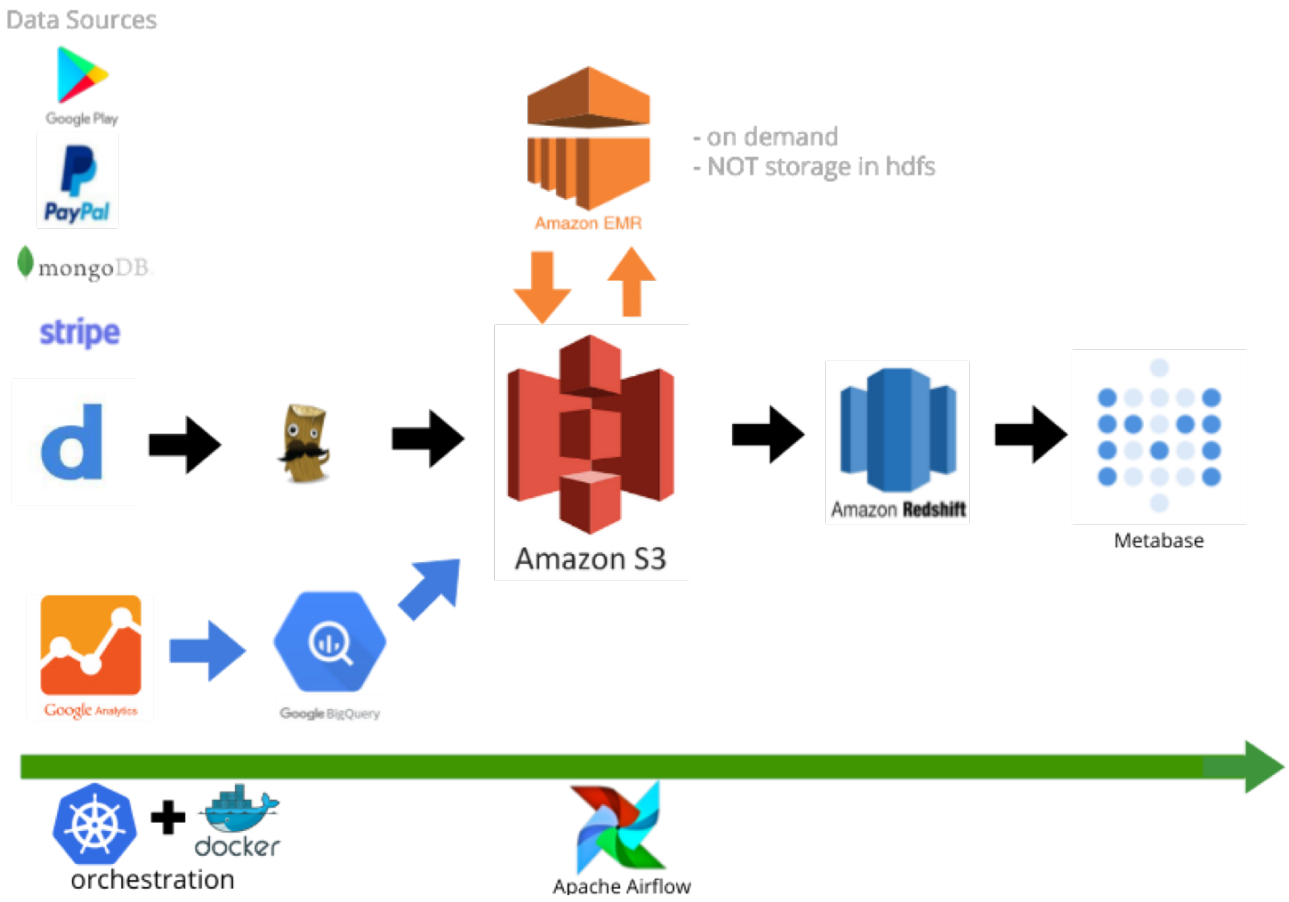
Introduction to Airflow

Airflow is a platform to programmatically author,
Schedule and monitor workflows.

Use Airflow to author workflows as Directed Acyclic Graphs (DAGs) of tasks. The Airflow scheduler executes your tasks on an array of workers while following the specified dependencies.
Rich command line utilities make performing complex surgeries on DAGs a snap.

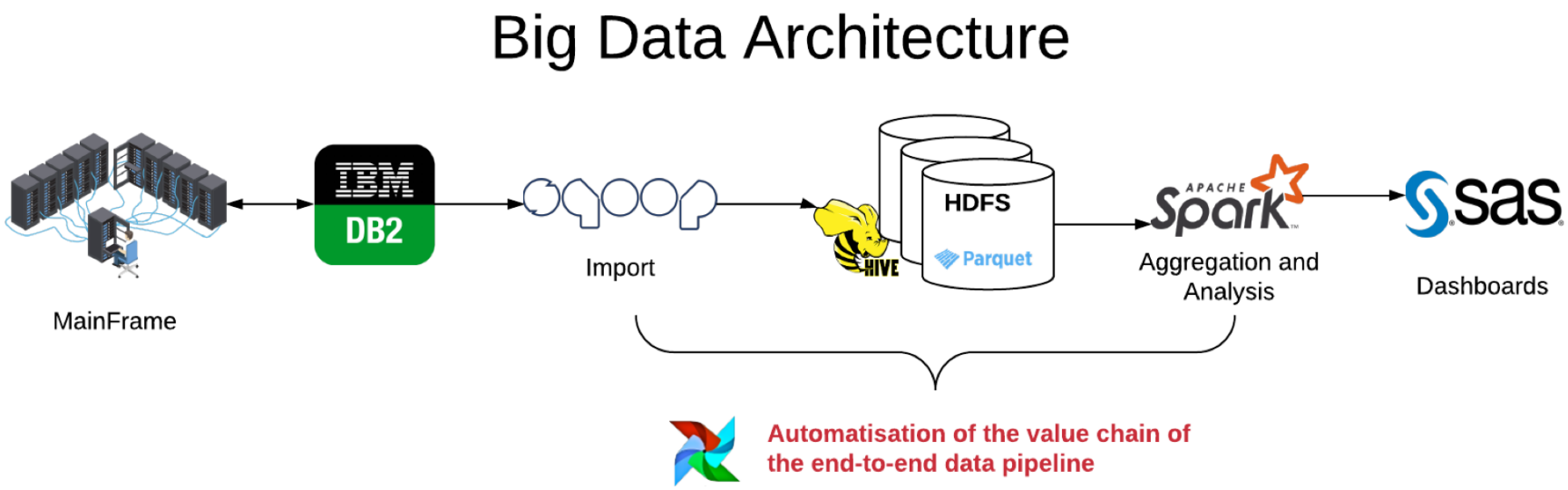
The rich user interface makes it easy to visualize pipelines running in production, monitor progress, and troubleshoot issues when needed.
When workflows are defined as code, they become more maintainable, versionable, testable, and collaborative.

Introduction to Airflow



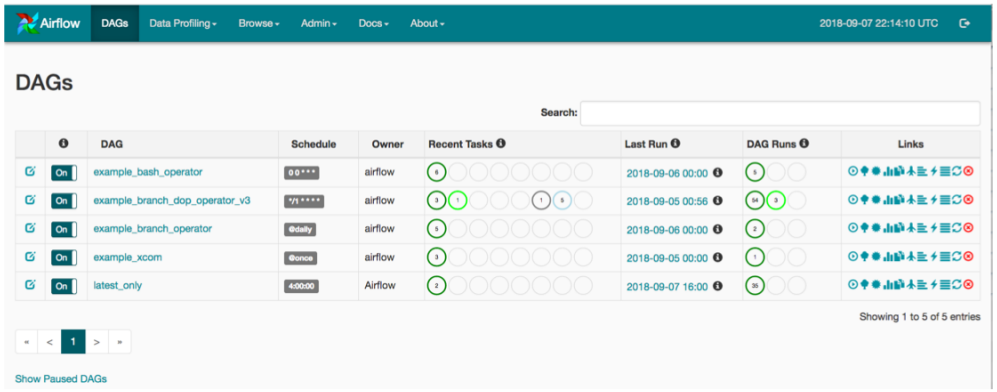
Introduction to Airflow

Airflow is a platform to programmatically author,
Schedule and monitor workflows.



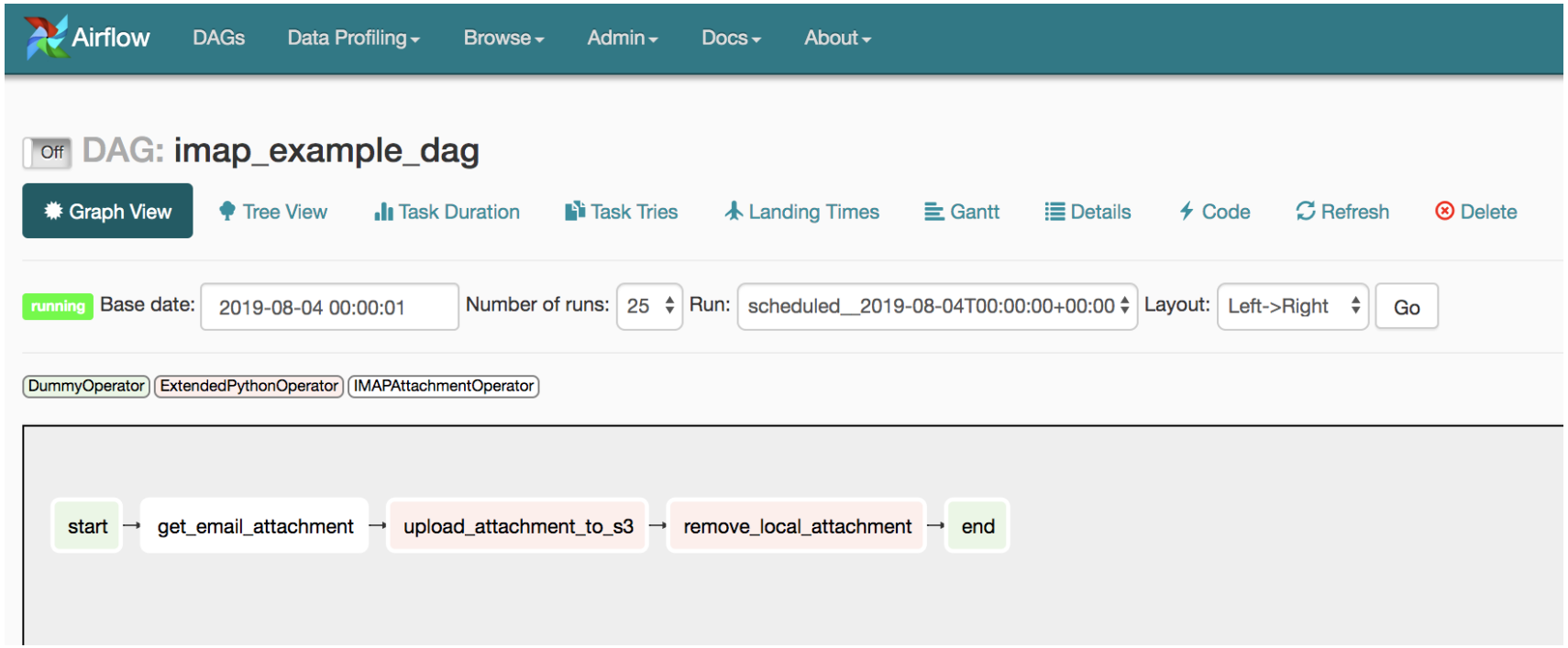
Why Airflow?

- One Place for schedule and data pipeline
- Python code base
- Callable events
- Xcoms
- Trigger rules
- Simply UI
- Powerful CLI
- Queues & Pools
- Growing community



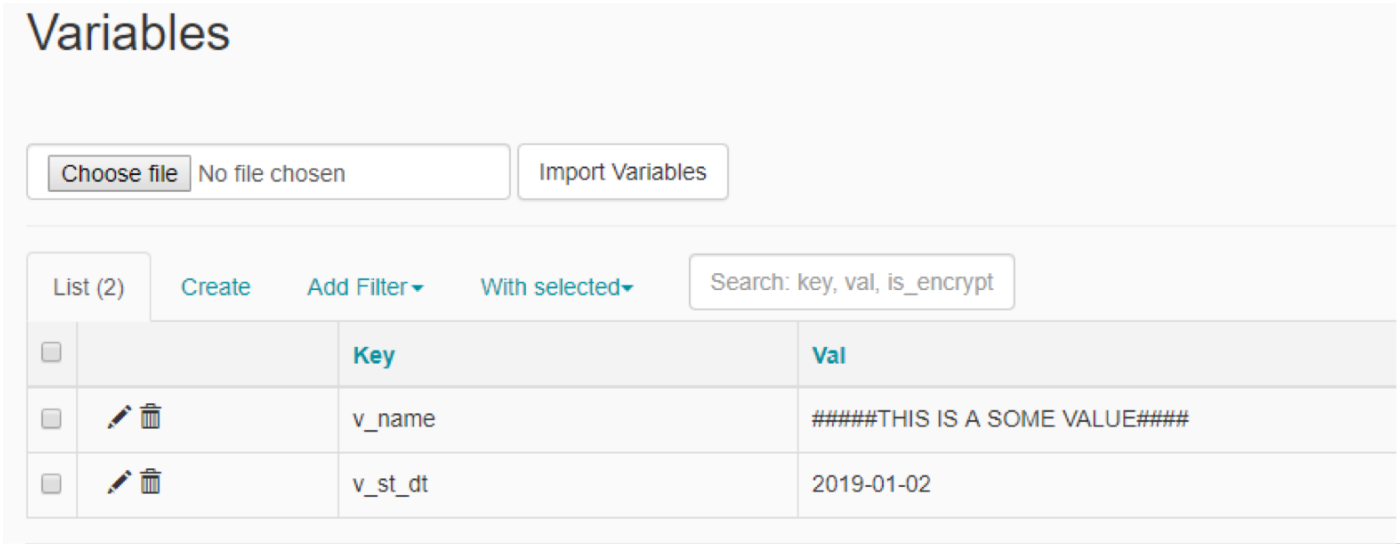
Components

- **Task –**a defined unit of work (these are called operators in Airflow)
- **DAG –**Directed acyclic graph, a set of tasks with explicit execution order, beginning, and end



Components

- **Hooks –** Interfaces to external platforms and databases like Hive, S3, MySQL, Postgres, HDFS, and Pig
- **Airflow pools –** can be used to limit the execution parallelism on arbitrary sets of tasks.
- **Variables –** Generic way to store and retrieve arbitrary content or settings as a simple key value store within Airflow



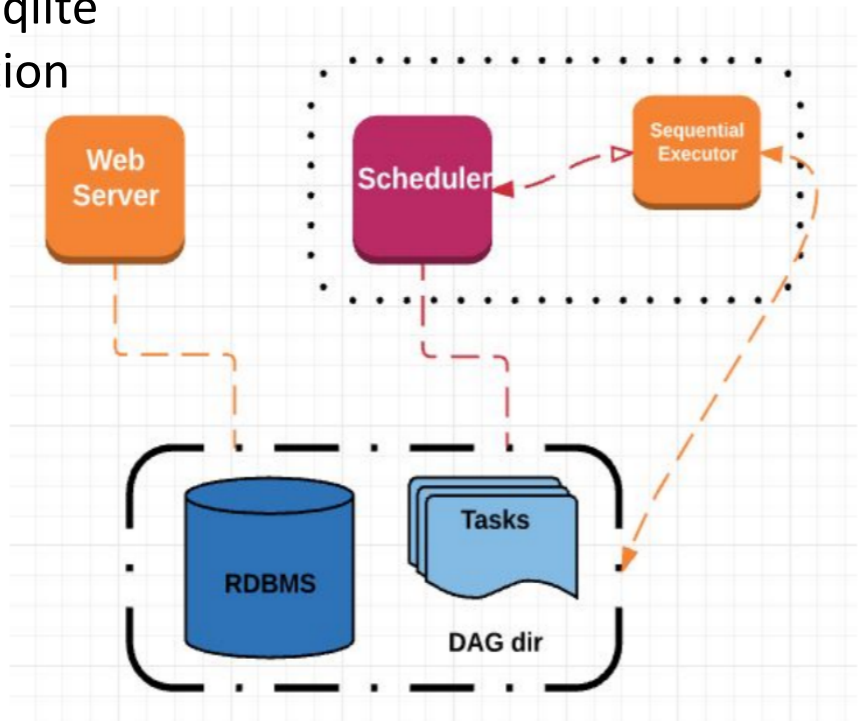
Components

- **Queues –** When using the CeleryExecutor, the celery queues that tasks are sent to can be specified. queue is an attribute of BaseOperator.
- **SubDAGs –**Defining a function that returns a DAG object is a nice design pattern when using Airflow.
- **SLAs –**Service Level Agreements, or time by which a task or DAG should have succeeded, can be set at a task level as a timedelta
- **Trigger Rules –**Though the normal workflow behavior is to trigger tasks when all their directly upstream tasks have succeeded, Airflow allows for more complex dependency settings.

Architecture

Sequential

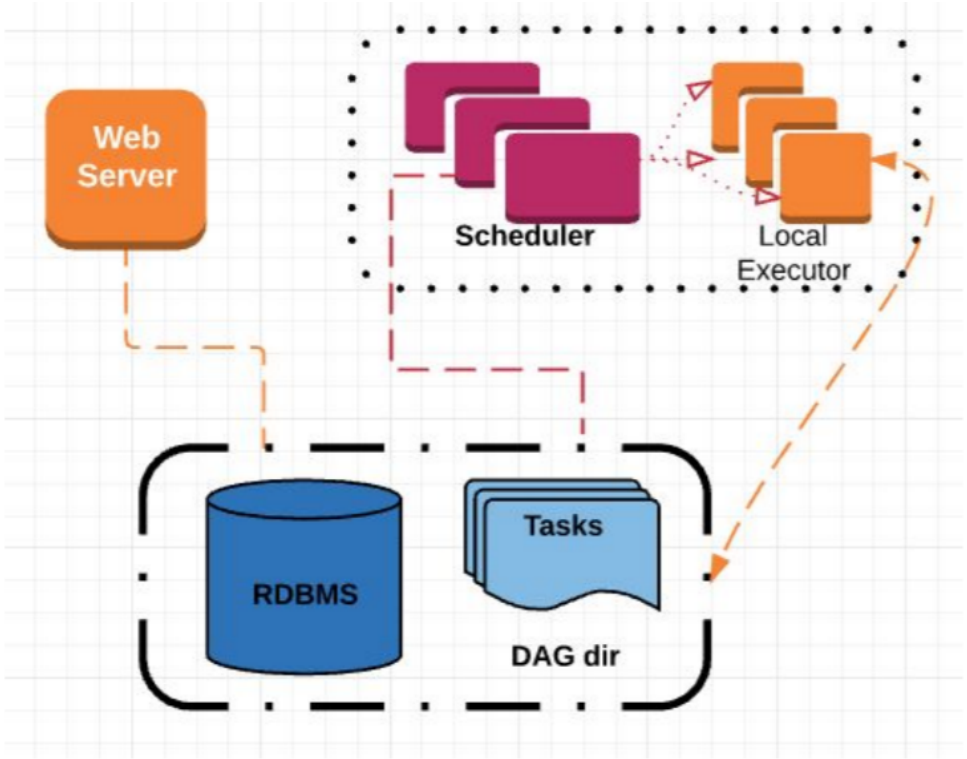
- Process one task at time (not parallelism)
- Default mode
- Minimum setup – works with sqlite
- Not recommended for production



Architecture

Local Executor

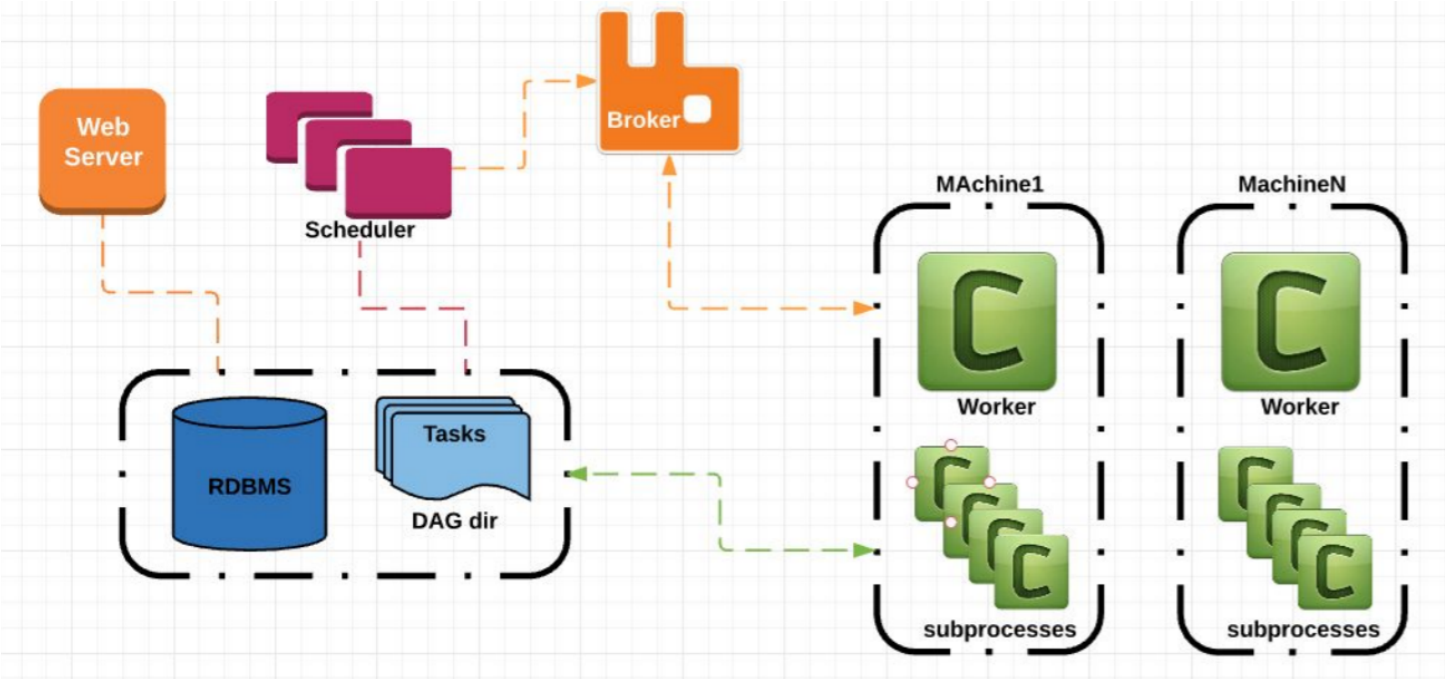
- Spawned by scheduler processes
- Scales vertical
- Production grade
- Doesn't need broker
- Suitable for production when there's not so many DAGs



Architecture

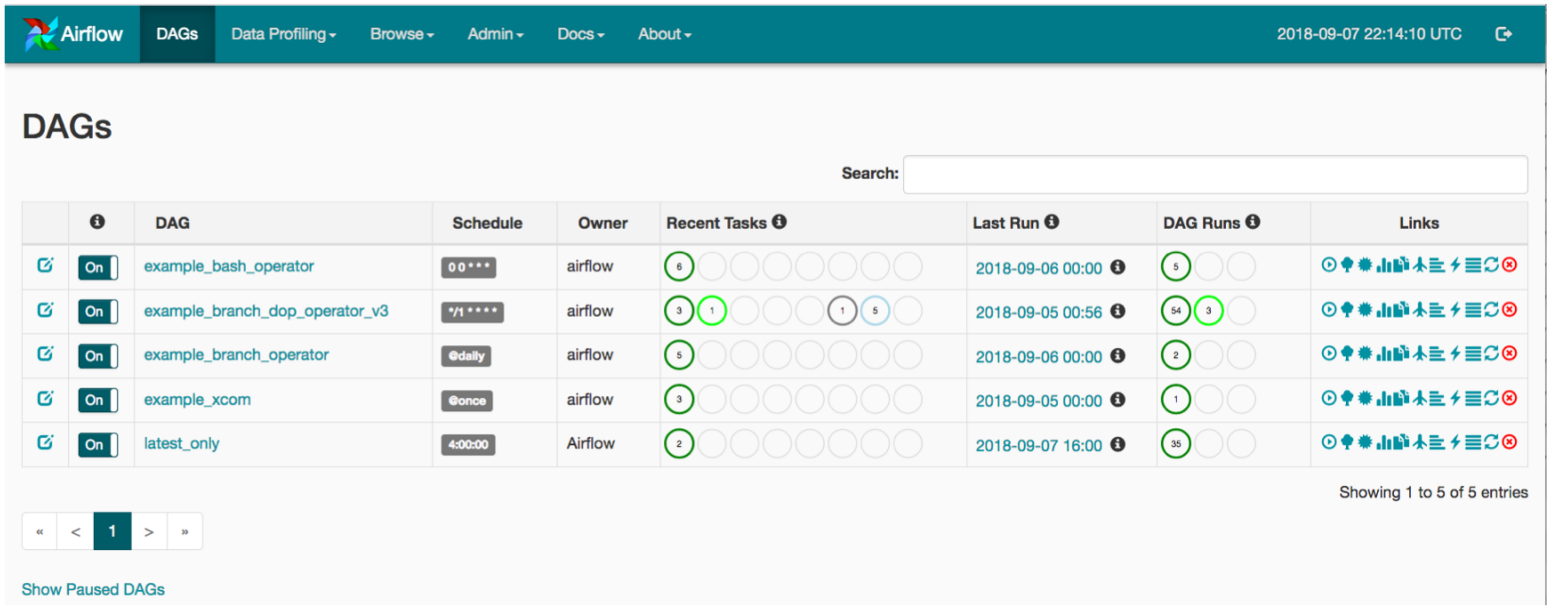
Celery Execute

- Vertical and Horizontal scalable
- Can be monitored (via Flower) Production grade
- Support Pools and Queues



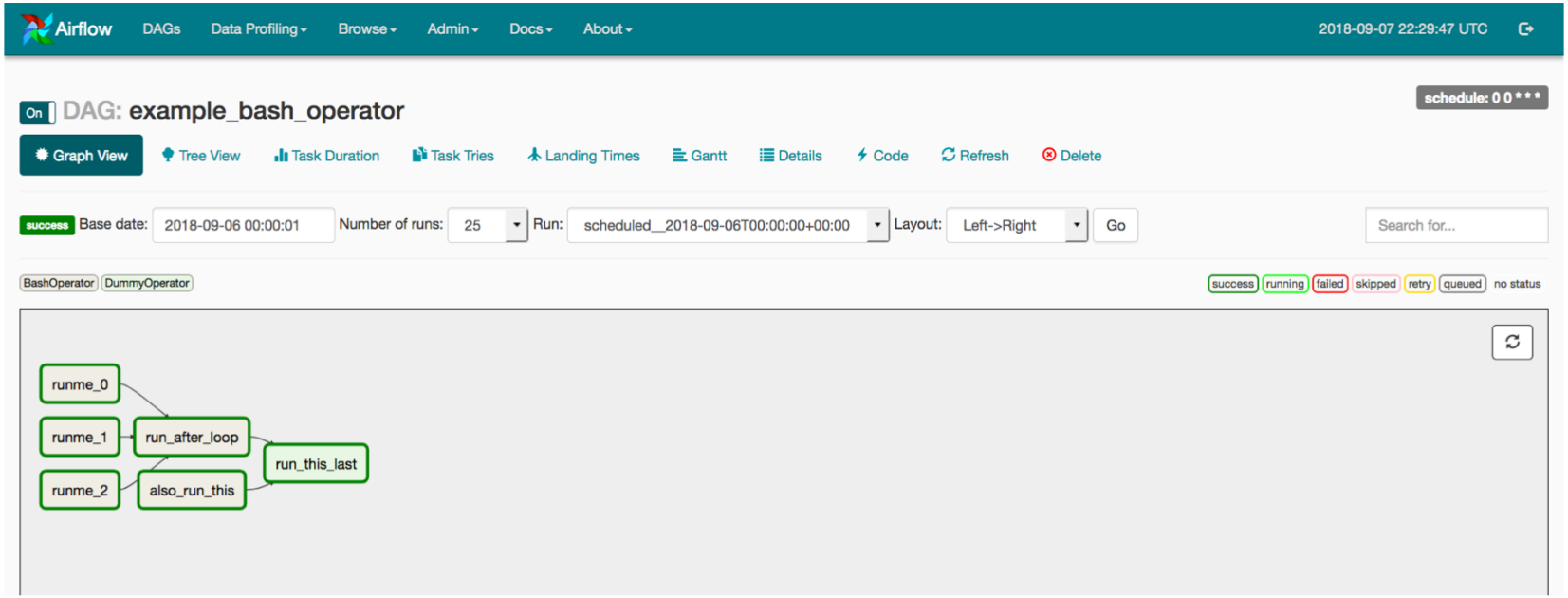
User Interface

List of DAGs



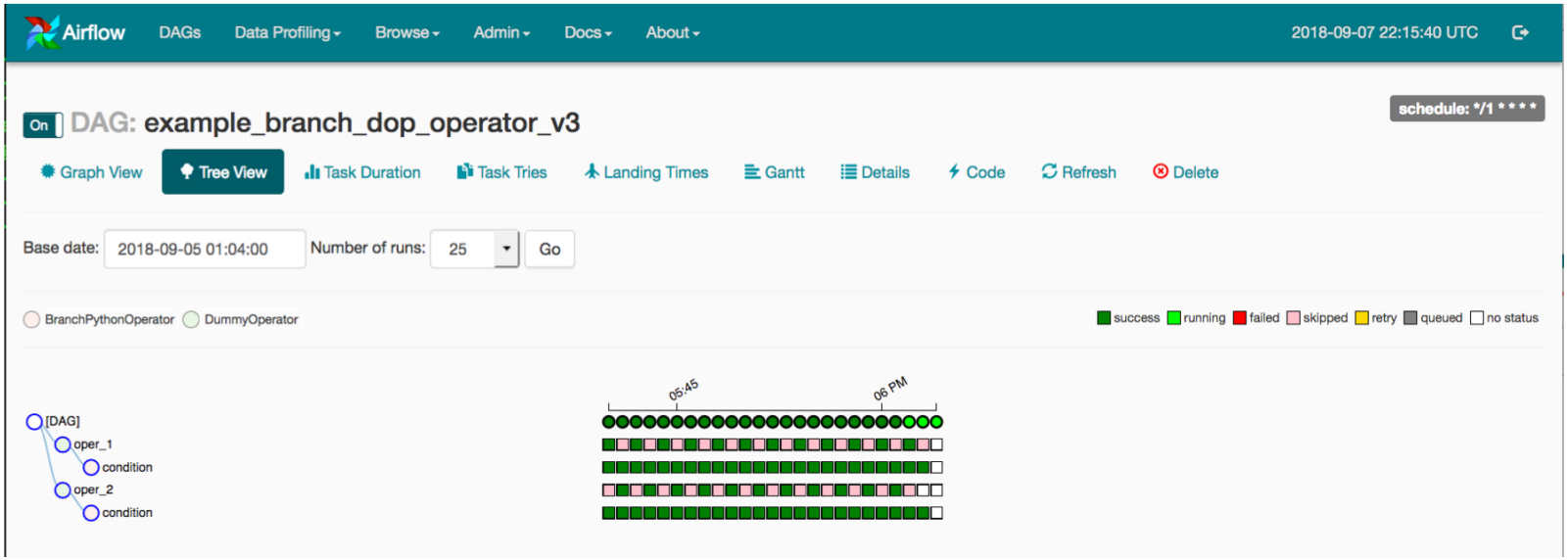
User Interface

Graph View



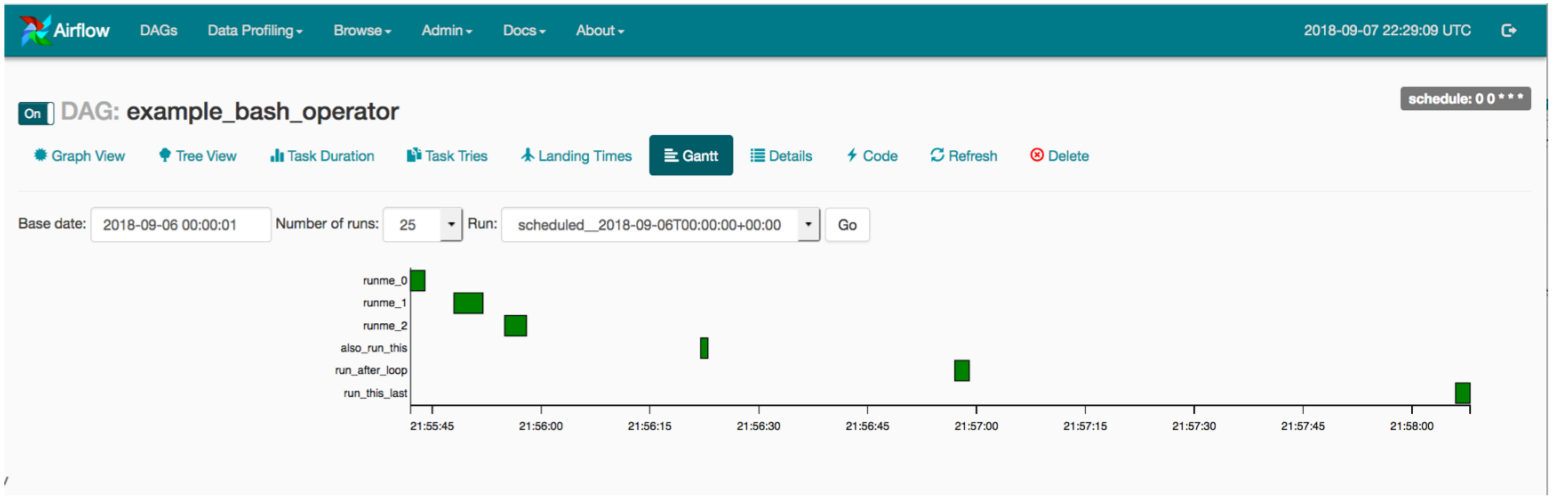
User Interface

Tree View



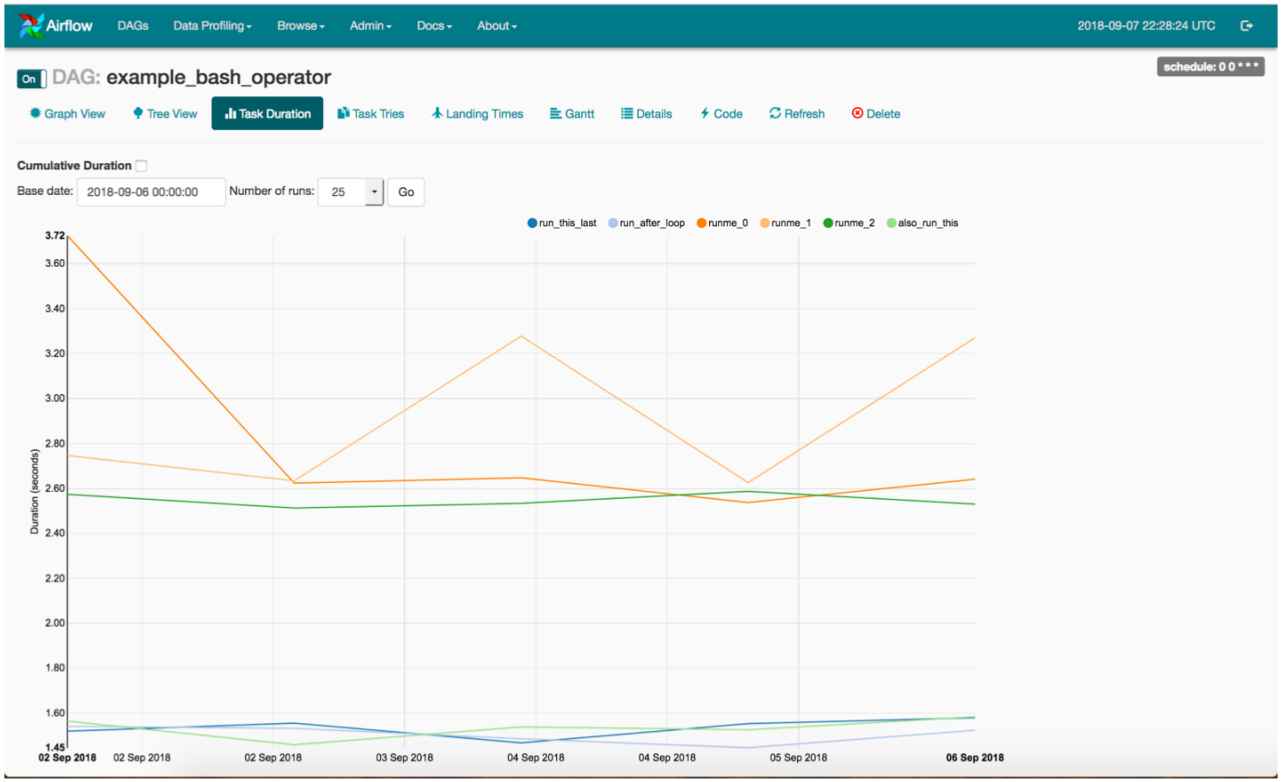
User Interface

Gantt Chart



User Interface

Task Duration



Airflow