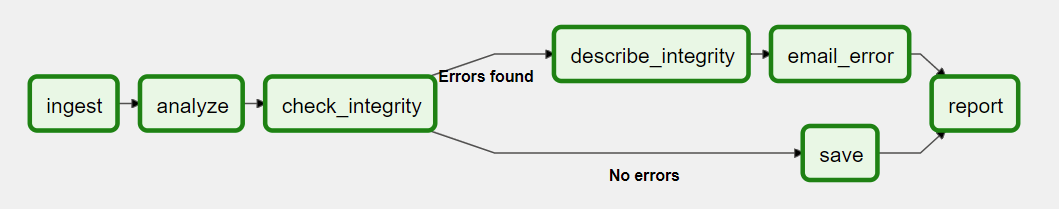
Architecture Overview

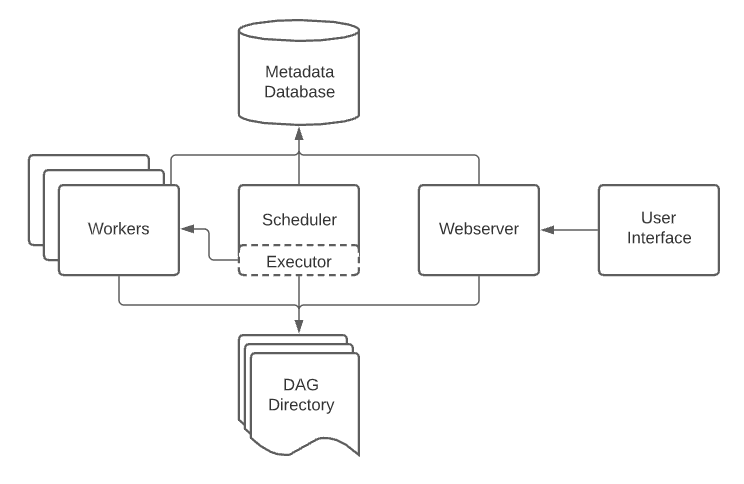
Airflow is a platform that lets you build and run *workflows*. A workflow is represented as a [DAG](https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html) (a Directed Acyclic Graph), and contains individual pieces of work called [Tasks](https://airflow.apache.org/docs/apache-airflow/stable/concepts/tasks.html), arranged with dependencies and data flows taken into account.



A DAG specifies the dependencies between Tasks, and the order in which to execute them and run retries; the Tasks themselves describe what to do, be it fetching data, running analysis, triggering other systems, or more.

An Airflow installation generally consists of the following components:

* A [scheduler](https://airflow.apache.org/docs/apache-airflow/stable/concepts/scheduler.html), which handles both triggering scheduled workflows, and submitting [Tasks](https://airflow.apache.org/docs/apache-airflow/stable/concepts/tasks.html) to the executor to run.
* An [executor](https://airflow.apache.org/docs/apache-airflow/stable/executor/index.html), which handles running tasks. In the default Airflow installation, this runs everything *inside* the scheduler, but most production-suitable executors actually push task execution out to *workers*.
* A *webserver*, which presents a handy user interface to inspect, trigger and debug the behaviour of DAGs and tasks.
* A folder of *DAG files*, read by the scheduler and executor (and any workers the executor has)
* A *metadata database*, used by the scheduler, executor and webserver to store state.



Most executors will generally also introduce other components to let them talk to their workers - like a task queue - but you can still think of the executor and its workers as a single logical component in Airflow overall, handling the actual task execution.

Airflow itself is agnostic to what you’re running - it will happily orchestrate and run anything, either with high-level support from one of our providers, or directly as a command using the shell or Python [Operators](https://airflow.apache.org/docs/apache-airflow/stable/concepts/operators.html).

Workloads

A DAG runs through a series of [Tasks](https://airflow.apache.org/docs/apache-airflow/stable/concepts/tasks.html), and there are three common types of task you will see:

* [Operators](https://airflow.apache.org/docs/apache-airflow/stable/concepts/operators.html), predefined tasks that you can string together quickly to build most parts of your DAGs.
* [Sensors](https://airflow.apache.org/docs/apache-airflow/stable/concepts/sensors.html), a special subclass of Operators which are entirely about waiting for an external event to happen.
* A [TaskFlow](https://airflow.apache.org/docs/apache-airflow/stable/concepts/taskflow.html)-decorated @task, which is a custom Python function packaged up as a Task.

Internally, these are all actually subclasses of Airflow’s BaseOperator, and the concepts of Task and Operator are somewhat interchangeable, but it’s useful to think of them as separate concepts - essentially, Operators and Sensors are *templates*, and when you call one in a DAG file, you’re making a Task.

Control Flow

[DAGs](https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html) are designed to be run many times, and multiple runs of them can happen in parallel. DAGs are parameterized, always including an interval they are “running for” (the [data interval](https://airflow.apache.org/docs/apache-airflow/stable/dag-run.html#data-interval)), but with other optional parameters as well.

[Tasks](https://airflow.apache.org/docs/apache-airflow/stable/concepts/tasks.html) have dependencies declared on each other. You’ll see this in a DAG either using the >> and << operators:

first\_task >> [second\_task, third\_task]

fourth\_task << third\_task

Or, with the set\_upstream and set\_downstream methods:

first\_task.set\_downstream([second\_task, third\_task])

fourth\_task.set\_upstream(third\_task)

These dependencies are what make up the “edges” of the graph, and how Airflow works out which order to run your tasks in. By default, a task will wait for all of its upstream tasks to succeed before it runs, but this can be customized using features like [Branching](https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html#concepts-branching), [LatestOnly](https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html" \l "concepts-latest-only), and [Trigger Rules](https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html#concepts-trigger-rules).

To pass data between tasks you have two options:

* [XComs](https://airflow.apache.org/docs/apache-airflow/stable/concepts/xcoms.html) (“Cross-communications”), a system where you can have tasks push and pull small bits of metadata.
* Uploading and downloading large files from a storage service (either one you run, or part of a public cloud)

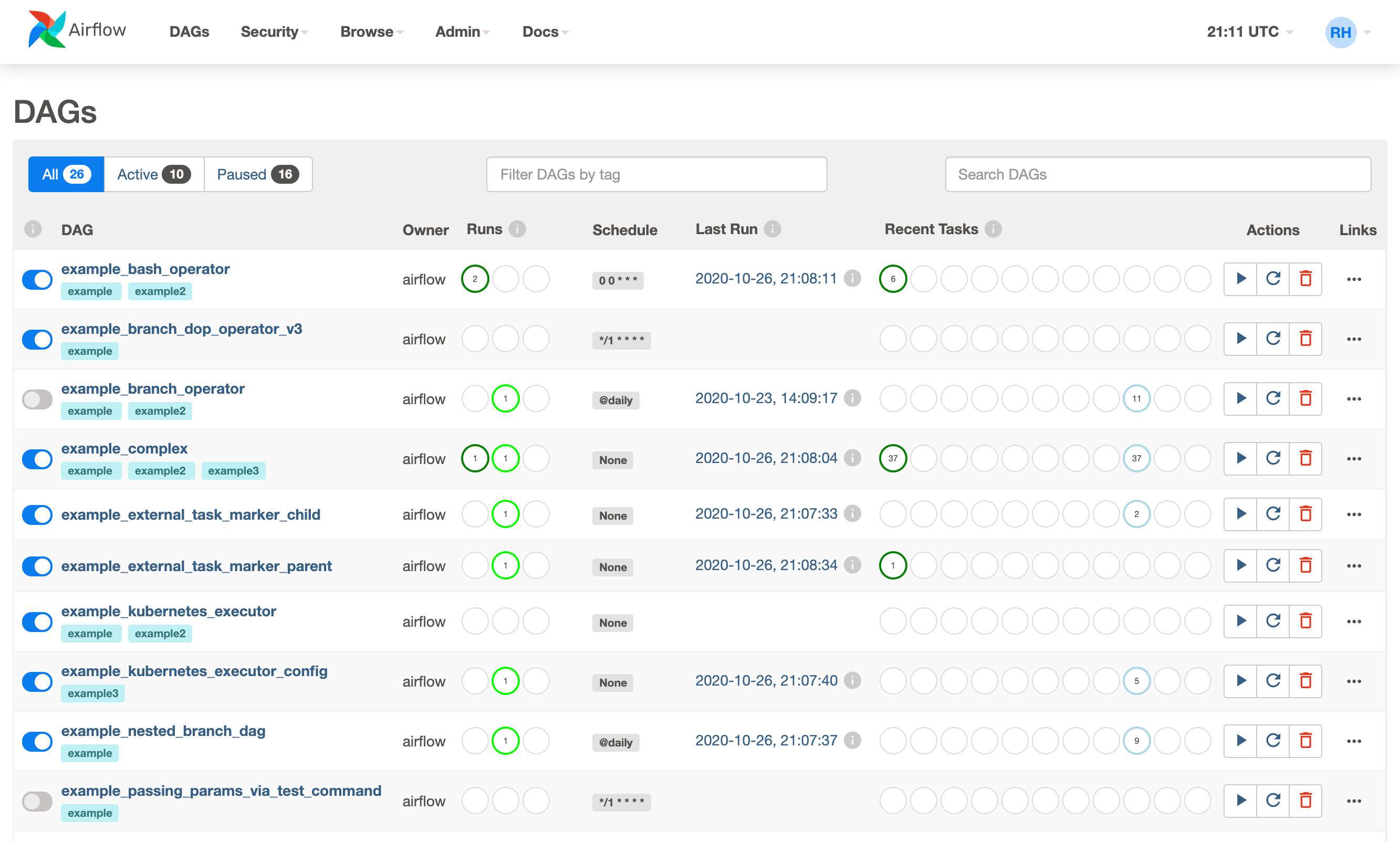
Airflow sends out Tasks to run on Workers as space becomes available, so there’s no guarantee all the tasks in your DAG will run on the same worker or the same machine.

As you build out your DAGs, they are likely to get very complex, so Airflow provides several mechanisms for making this more sustainable - [SubDAGs](https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html" \l "concepts-subdags) let you make “reusable” DAGs you can embed into other ones, and [TaskGroups](https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html" \l "concepts-taskgroups) let you visually group tasks in the UI.

There are also features for letting you easily pre-configure access to a central resource, like a datastore, in the form of [Connections & Hooks](https://airflow.apache.org/docs/apache-airflow/stable/concepts/connections.html), and for limiting concurrency, via [Pools](https://airflow.apache.org/docs/apache-airflow/stable/concepts/pools.html).

User interface

Airflow comes with a user interface that lets you see what DAGs and their tasks are doing, trigger runs of DAGs, view logs, and do some limited debugging and resolution of problems with your DAGs.



It’s generally the best way to see the status of your Airflow installation as a whole, as well as diving into individual DAGs to see their layout, the status of each task, and the logs from each task.