What's new in Airflow 2.3?

Kaxil Naik
Who am I?

- Committer & PMC Member of Apache Airflow
- Director of Airflow Engineering @ Astronomer

@kaxil
Please fill our Airflow Survey

Biggest Airflow Release since 2.0

700+ commits! with 50 new features
Dynamic Task Mapping
Highlight feature of 2.3

First-class support for common ETL pattern around dynamic tasks

Run same set of tasks for N number of files in a bucket, DB records, ML models where N is unpredictable.
Dynamic Task Mapping

```python
from datetime import datetime
from airflow import DAG
from airflow.decorators import task

with DAG(dag_id="simple_mapping", start_date=datetime(2022, 3, 4)) as dag:
    @task
def add_one(x: int):
        return x + 1

    @task
def sum_it(values):
        total = sum(values)
        print(f"Total was {total}"

    added_values = add_one.expand(x=[1, 2, 3])
    sum_it(added_values)
```
Grid View replaces Tree View!!
Grid View replaces Tree View!!

Better support for Task Groups & Task Mapping

Grid lines and hover effects to see which task you are inspecting

Show durations of dag runs to quickly see performance changes

Paves way for DAG Versioning
Create Connection in native JSON format

```bash
export AIRFLOW_CONN_MY_PROD_DATABASE='my-conn-type://login:password@host:port/schema?param1=val1&param2=val2'
```
Create Connection in native JSON format

```json
export AIRFLOW_CONN_MY_PROD_DATABASE='{
  "conn_type": "my-conn-type",
  "login": "my-login",
  "password": "my-password",
  "host": "my-host",
  "port": 1234,
  "schema": "my-schema",
  "extra": {
    "param1": "val1",
    "param2": "val2"
  }
}
'
Create Connection in native JSON format

```
export AIRFLOW_CONN_MY_PROD_DATABASE='{
    "conn_type": "my-conn-type",
    "login": "my-login",
    "password": "my-password",
    "host": "my-host",
    "port": 1234,
    "schema": "my-schema",
    "extra": {
        "param1": "val1",
        "param2": "val2"
    }
}'
```
Create Connection in native JSON format

```
airflow connections add 'my_prod_db' \n   --conn-json '{
   "conn_type": "my-conn-type",
   "login": "my-login",
   "password": "my-password",
   "host": "my-host",
   "port": 1234,
   "schema": "my-schema",
   "extra": {
      "param1": "val1",
      "param2": "val2"
   }
}';
```
DB downgrades

First class support

Downgrades to a
- Airflow version
- or to a specific Alembic revision id

```
airflow@11a9898163e:/opt/airflow$ airflow db downgrade --help
usage: airflow db downgrade [-h] [--from-revision FROM_REVISION] [--from-version FROM_VERSION] [\--show-sql-only] [-s] [-r TO_REVISION] [-n TO_VERSION] [-y]

Downgrade the schema of the metadata database. You must provide either '--to-revision' or '--to-version'. To print but not execute commands, use option '--show-sql-only'. If using options '--from-revision' or '--from-version', you must also use '--show-sql-only', because if actually *running* migrations, we should only migrate from the *current* Alembic revision.

optional arguments:
  -h, --help                show this help message and exit
  --from-revision FROM_REVISION
                             (Optional) If generating sql, may supply a *from* Alembic revision
  --from-version FROM_VERSION
                             (Optional) If generating sql, may supply a *from* version
  -s, --show-sql-only       Don't actually run migrations; just print out sql scripts for offline migration. Required if using either '--from-version' or '--from-revision'.
  -r TO_REVISION, --to-revision TO_REVISION
                             The Alembic revision to downgrade to. Note: must provide either
                             '--to-revision' or '--to-version'.
  -n TO_VERSION, --to-version TO_VERSION
                             (Optional) If provided, only run migrations up to this version.
  -y, --yes                 Do not prompt to confirm. Use with care!
```
DB downgrades
First class support
Generate SQL for DB upgrade & downgrade
Allows DBA to run the DB Migrations ("--show-sql-only" flag)

```bash
astro@f5647e47e9c5:/usr/local/airflow$ airflow db upgrade --from-version 2.3.0 --to-version 2.3.1 --show-sql-only
DB: postgresql://postgres:**@postgres:5432
Generating sql for upgrade -- upgrade commands will *not* be submitted.
BEGIN;

-- Running upgrade b1b348e02d07 -> 1de7bc13c950

CREATE INDEX idx_log_event ON log (event);

UPDATE alembic_version SET version_num='1de7bc13c950' WHERE alembic_version.version_num = 'b1b348e02d07';

COMMIT;
```
Purge DB history

First class support

Helps reduce time when running DB Migrations when updating Airflow version

Removes need of Maintenance DAGs!

`--dry-run` option to print the row counts in the tables to be cleaned

Backup your DB before running this!

```
> airflow db clean --help


Purge old records in metastore tables

optional arguments:
  -h, --help                show this help message and exit
  --clean-before-timestamp  CLEAN_BEFORE_TIMESTAMP
                            The date or timestamp before which data should be purged.
                            If no timezone info is supplied then dates are assumed to be in airflow default timezone.
                            Example: '2022-01-01 00:00:00+01:00'
  --dry-run                 Perform a dry run
  -t TABLES, --tables TABLES
                            Table names to perform maintenance on (use comma-separated list).
                            Options: ['callback_request', 'dag', 'dag_run', 'import_error', 'job', 'log',
                                        'rendered_task_instance_fields', 'sensor_instance', 'sla_miss', 'task_fail',
                                        'task_instance', 'task_reschedule', 'xcom']
  -v, --verbose             Make logging output more verbose
  -y, --yes                 Do not prompt to confirm. Use with care!
```
LocalKubernetesExecutor
Speed, Isolation & Simplicity packed in one!

Allows users to simultaneously run a LocalExecutor and KubernetesExecutor.

An executor is chosen to run a task based on the task's queue

Tasks just calling APIs + Tasks requiring isolation due to dependencies or computation-heavy

Slide from Jed’s Airflow’s Summit talk: https://www.crowdcast.io/e/airflowsummit2022/35
DAG Processor separation

Standalone process for DAG parsing

“airflow dag-processor” CLI Command

Code Parsing and Callbacks (Sla + DAG’s on_{success,failure}_callbacks)

Makes scheduler not run any user code*

First step towards multi-tenancy

Disabled by default, can be enabled by

AIRFLOW__SCHEDULER__STANDALONE_DAG_PROCESSOR=True

Images from AIP-43
Events Timetable

Run DAGs at arbitrary dates

Built-in Timetable

Useful for events which can’t be expressed by Cron or Timedelta

```python
import pendulum

from airflow import DAG
from airflow.timetables.events import EventsTimetable

with DAG(
    dag_id="example_after_workday_timetable_dag",
    start_date=pendulum.datetime(2022, 5, 20, tz="UTC"),
    timetable=EventsTimetable(
        [pendulum.datetime(2022, 5, 22, tz="UTC"),
         pendulum.datetime(2022, 5, 25, tz="UTC"),
         pendulum.datetime(2022, 7, 8, tz="UTC"),
        ],
    tags=\["example", "timetable"\],
) as dag:
    ...
```
Smooth Operator

```python
from airflow.operators.smooth import SmoothOperator
SmoothOperator(task_id="power", ...)
```
Other Minor features

Minor but very handy!

- A new REST API endpoint (`/dags`) that lets you bulk-pause/resume DAGs
- `airflow dags reserialize` command to delete serialized dags & reparse them
- A new listener plugin API that tracks TaskInstance state changes (used by OpenLineage)
- New Trigger Rule: `all_skipped`
- (Experimental) Support for ARM Docker Images
Upgrade Now to Airflow 2.3!

WHAT ARE YOU WAITING FOR?
#MISSIONIMPOSSIBLE
Thank You

@kaxil