

What's coming in Apache Airflow 2.0

NYC Meetup
13th of May 2020



Who are we?



Tomek Urbaszek

Committer
Software Engineer @ Polidea



Jarek Potiuk

Committer, PMC member
Principal Software Engineer @ Polidea



Kamil Breguła

Committer
Software Engineer @ Polidea



Ash Berlin-Taylor

Committer, PMC member
Airflow Engineering Lead @ Astronomer



Daniel Imberman

Committer
Senior Data Engineer @ Astronomer



Kaxil Naik

Committer, PMC member
Senior Data Engineer @ Astronomer

High Availability



Scheduler High Availability

Goals:

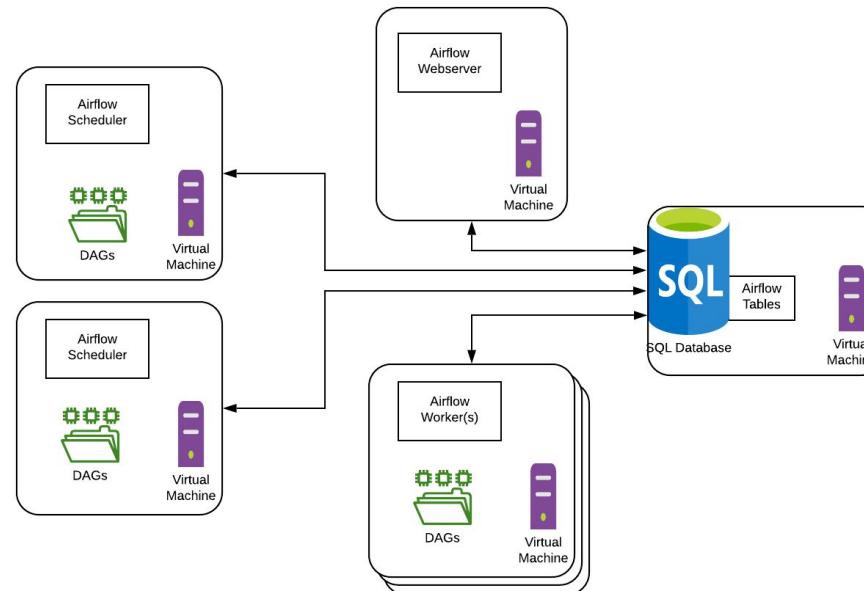
- Performance - reduce task-to-task schedule "lag"
- Scalability - increase task throughput by horizontal scaling
- Resiliency - kill a scheduler and have tasks continue to be scheduled

Scheduler High Availability: Design

- Active-active model. Each scheduler does everything
- Uses existing database - no new components needed, no extra operational burden
- Plan to use row-level-locks in the DB
- Will re-evaluate if performance/stress testing show the need

Example HA configuration

Airflow Schedulers running in High Availability
on virtual machines - example configuration



Scheduler High Availability: Tasks

- Separate DAG parsing from DAG scheduling

This removes the tie between parsing and scheduling that is still present

- Run a mini scheduler *in the worker* after each task is completed

A.K.A. "fast follow". Look at immediate down stream tasks of what just finished and see what we can schedule

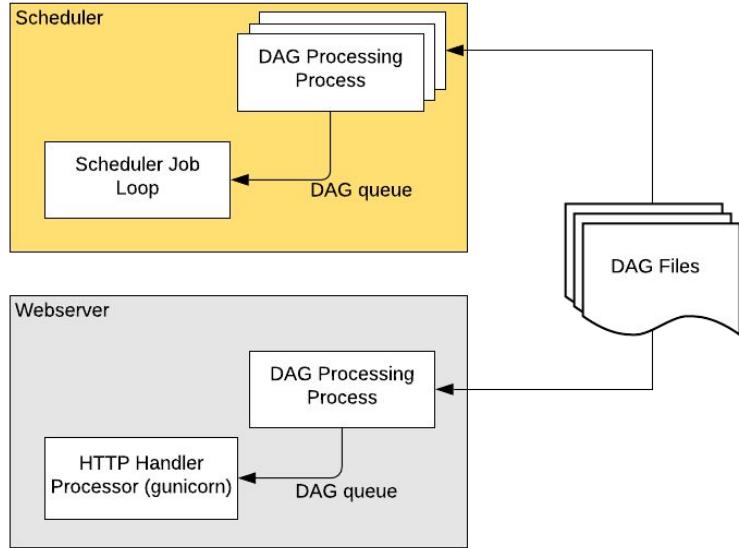
- Test it to destruction

This is a big architectural change, we need to be sure it works well.

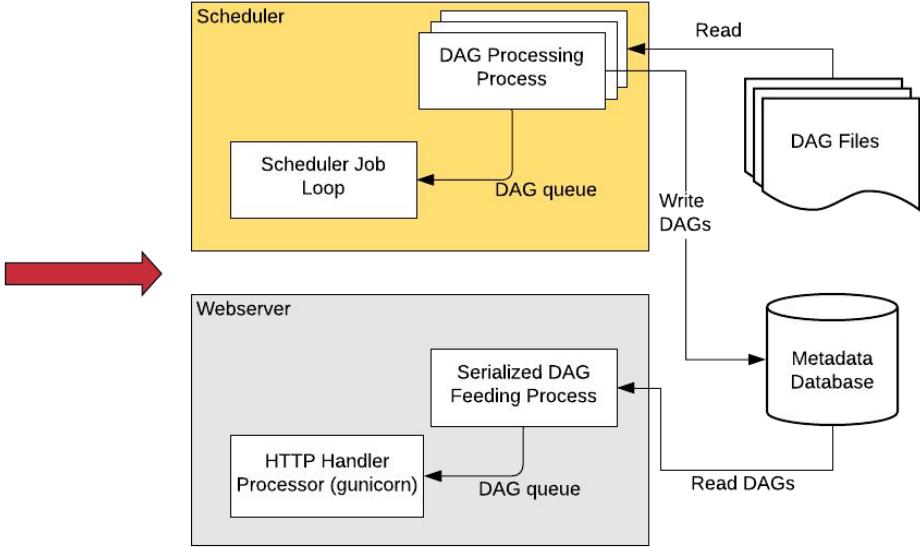
DAG Serialization



Dag Serialization



(1) Vanilla Airflow



(2) Airflow with DAG
Serialization

Dag Serialization (Tasks Completed)

- **Stateless Webserver:** Scheduler parses the DAG files, serializes them in JSON format & saves them in the Metadata DB.
- **Lazy Loading of DAGs:** Instead of loading an entire DagBag when the Webserver starts we only load each DAG on demand. This helps **reduce Webserver startup time and memory**. This reduction is notable with large number of DAGs.
- Deploying new DAGs to Airflow - no longer requires long restarts of webserver (if DAGs are baked in Docker image)
- Feature to use the “JSON” library of choice for Serialization (default is inbuilt ‘json’ library)
- Paves way for **DAG Versioning & Scheduler HA**

Dag Serialization (Tasks In-Progress for Airflow 2.0)

- Decouple DAG Parsing and Serializing from the scheduling loop.
- Scheduler will fetch DAGs from DB
- DAG will be parsed, serialized and saved to DB by a separate component “Serializer”/ “Dag Parser”
- This should reduce the delay in Scheduling tasks when the number of DAGs are large

DAG Versioning



Dag Versioning

Current Problem:

- Change in DAG structure affects viewing previous DagRuns too
- Not possible to view the code associated with a specific DagRun

Dag Versioning (Current Problem)

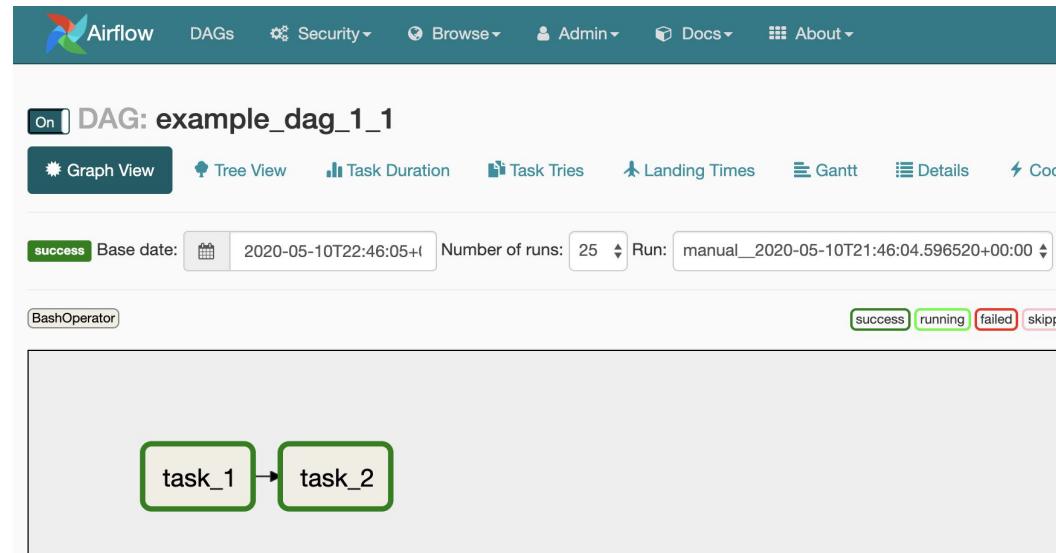
```
from airflow.models.dag import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime

with DAG('example_dag_1_1', schedule_interval=None,
         start_date=datetime(2020, 4, 25)) as example_dag_1_1:

    task_1 = BashOperator(
        task_id='task_1',
        bash_command='echo hello',
    )

    task_2 = BashOperator(
        task_id='task_2',
        bash_command='echo hello',
    )

    task_1 >> task_2
```



Dag Versioning (Current Problem)



```
from airflow.models.dag import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime

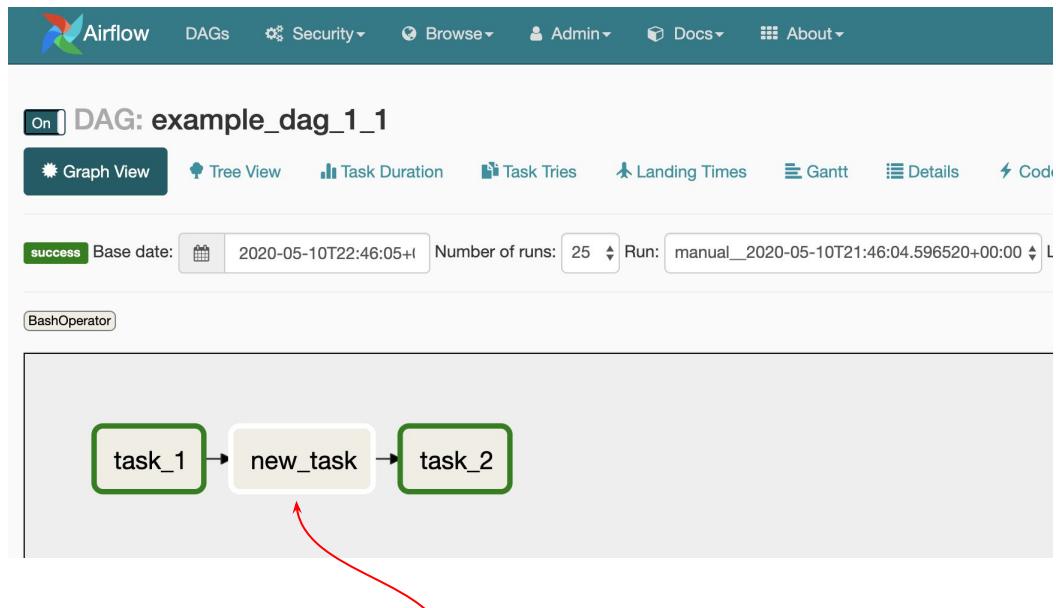
with DAG('example_dag_1_1', schedule_interval=None,
         start_date=datetime(2020, 4, 25)) as example_dag_1_1:

    task_1 = BashOperator(
        task_id='task_1',
        bash_command='echo hello',
    )

    new_task = BashOperator(
        task_id='new_task',
        bash_command='echo hello',
    )

    task_2 = BashOperator(
        task_id='task_2',
        bash_command='echo hello',
    )

    task_1 >> new_task >> task_2
```



New task is shown in Graph View for older DAG Runs too with “no status”.

Dag Versioning

Current Problem:

- Change in DAG structure affects viewing previous DagRuns too
- Not possible to view the code associated with a specific DagRun

Goal:

- Support for storing multiple versions of Serialized DAGs
- Baked-In Maintenance DAGs to cleanup old DagRuns & associated Serialized DAGs
- Graph View shows the DAG associated with that DagRun

Performance Improvements



Performance improvements

- Review each component of scheduler in turn and its optimization.
- Perf kit
 - A set of tools that allows you to quickly check the performance of a component



```
@timing()
def test_dag_sync():
    with count_queries():
        DAG.bulk_sync_to_db()
```

Do you see a performance problem?



```
paused_dag_ids = {dag.dag_id for dag in dagbag.dags.values() if dag.is_paused}
```



```
paused_dag_ids = {dag.dag_id for dag in dagbag.dags.values() if dag.is_paused}
```



```
class DAG:

    @provide_session
    def _get_is_paused(self, session=None):
        qry = session.query(DagModel).filter(
            DagModel.dag_id == self.dag_id)
        return qry.value(DagModel.is_paused)

    @property
    def is_paused(self):
        return self._get_is_paused()
```

Results for DagFileProcessor

When we have one DAG file with 200 DAGs, each DAG with 5 tasks:

	Before	After	Diff
Average time:	8080.246 ms	628.801 ms	-7452 ms (92%)
Queries count:	2692	5	-2687 (99%)

How to avoid regression?



```
with assert_queries_count(3):  
    DAG.bulk_sync_to_db(dags)
```

REST API



API: follows Open API 3.0 specification

Outreachy interns



Ephraim Anierobi



Omair Khan

Dev/CI environment



CI environment

- Moving to GitHub Actions
 - Kubernetes Tests
 - Easier way to test Kubernetes Tests locally
- Quarantined tests
 - Process of fixing the Quarantined tests
- Thinning CI image
 - Move integrations out of the image (hadoop etc)
- Automated System Tests (AIP-21)

GitHub Actions

The screenshot shows a GitHub Actions CI Build log for a repository. The build is triggered by a pull request and consists of several steps:

- Run actions/setup-python@v1**: 0s
- Free space**: 0s
- Build CI image 3.6**: 7m 19s
- Tests**: 24m 38s

The tests section lists numerous test cases across various provider modules, each with a timestamp, file path, and progress bar indicating completion status (e.g., [61%]). Some tests are marked as skipped (e.g., `.....`).

Test Case	Timestamp	File Path	Status
tests/providers/google/cloud/hooks/test_spanner.py	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_spanner.py	[61%]
.....	Tue, 12 May 2020 10:51:38 GMT	[61%]
tests/providers/google/cloud/hooks/test_speech_to_text.py ..	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_speech_to_text.py	[61%]
tests/providers/google/cloud/hooks/test_stackdriver.py	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_stackdriver.py	[62%]
tests/providers/google/cloud/hooks/test_tasks.py	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_tasks.py	[62%]
tests/providers/google/cloud/hooks/test_text_to_speech.py ..	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_text_to_speech.py	[62%]
tests/providers/google/cloud/hooks/test_translate.py ..	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_translate.py	[62%]
tests/providers/google/cloud/hooks/test_video_intelligence.py ..	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_video_intelligence.py	[62%]
tests/providers/google/cloud/hooks/test_vision.py	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/hooks/test_vision.py	[62%]
.....	Tue, 12 May 2020 10:51:38 GMT	[63%]
tests/providers/google/cloud/operators/test_adls_to_gcs.py ...	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/operators/test_adls_to_gcs.py	[63%]
tests/providers/google/cloud/operators/test_automl.py	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/operators/test_automl.py	[63%]
tests/providers/google/cloud/operators/test_automl_system.py ss	Tue, 12 May 2020 10:51:38 GMT	tests/providers/google/cloud/operators/test_automl_system.py ss	[63%]
tests/providers/google/cloud/operators/test_bigquery.py	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery.py	[63%]
.....	Tue, 12 May 2020 10:51:39 GMT	[64%]
tests/providers/google/cloud/operators/test_bigquery_dts.py ...	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_dts.py	[64%]
tests/providers/google/cloud/operators/test_bigquery_dts_system.py s	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_dts_system.py s	[64%]
tests/providers/google/cloud/operators/test_bigquery_system.py ss	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_system.py ss	[64%]
tests/providers/google/cloud/operators/test_bigquery_to_bigquery.py ..	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_to_bigquery.py	[64%]
tests/providers/google/cloud/operators/test_bigquery_to_bigquery_system.py s [64%]	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_to_bigquery_system.py s	[64%]
.....	Tue, 12 May 2020 10:51:39 GMT	[64%]
tests/providers/google/cloud/operators/test_bigquery_to_gcs.py ..	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_to_gcs.py	[64%]
tests/providers/google/cloud/operators/test_bigquery_to_gcs_system.py s	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_to_gcs_system.py s	[64%]
tests/providers/google/cloud/operators/test_bigquery_to_mysql.py ..	Tue, 12 May 2020 10:51:39 GMT	tests/providers/google/cloud/operators/test_bigquery_to_mysql.py	[64%]
tests/providers/google/cloud/operators/test_bigtable.py	Tue, 12 May 2020 10:51:40 GMT	tests/providers/google/cloud/operators/test_bigtable.py	[64%]
.....	Tue, 12 May 2020 10:51:40 GMT	[64%]
tests/providers/google/cloud/operators/test_bigtable_system.py s [64%]	Tue, 12 May 2020 10:51:40 GMT	tests/providers/google/cloud/operators/test_bigtable_system.py s	[64%]
tests/providers/google/cloud/operators/test_cassandra_to_gcs.py ..	Tue, 12 May 2020 10:51:40 GMT	tests/providers/google/cloud/operators/test_cassandra_to_gcs.py	[64%]

Final steps include **Post Run actions/checkout@master** (0s) and **Complete job** (0s).

Dev environment

- Breeze
 - unit testing
 - package building
 - release preparation
 - refreshing videos
- CodeSpaces integration

```
Usage: breeze [FLAGS] [COMMAND] -- <EXTRA_ARGS>
By default the script enters IT environment and drops you to bash shell, but you can choose one
of the commands to run specific actions instead. Add --help after each command to see details:
Commands without arguments:
shell                                     [Default] Enters interactive shell in the container
build-docs                                 Builds documentation in the container
build-image                                Builds CI or Production docker image
cleanup-image                             Cleans up the container image created
exec                                       Execs into running breeze container in new terminal
generate-requirements                     Generates pinned requirements for pip dependencies
generate-backport-readme                Generates backport packages readme files
prepare-backport-packages               Prepares backport packages
initialize-local-virtualenv             Initializes local virtualenv
setup-autocomplete                      Sets up autocomplete for breeze or from
stop                                       Stops the docker-compose environment
restart                                     Stops the docker-compose environment including DB cleanup
toggle-suppress-cheatsheet              Toggles on/off cheatsheet
toggle-suppress-asciart                  Toggles on/off asciiart
Commands with arguments:
docker-compose                            Executes specified docker-compose command
execute-command                           Executes specified command in the container
static-check                               Performs selected static check for changed files
static-check-all-files                   Performs selected static check for all files
test-target                                Runs selected test target in the container
Help commands:
flags                                     Shows all breeze's flags
help                                      Shows this help message
help-all                                   Shows detailed help for all commands and flags
Run 'breeze flags' to see all applicable flags.
```

Backport Packages

- Bring Airflow 2.0 providers to 1.10.*
- Packages per-provider
- 58 packages (!)
- Python 3.6+ only(!)
- Automatically tested on CI
- Future
 - Automated System Tests (AIP-21)
 - Split Airflow (AIP-8)?

```
    ✓ Prepare & test backport packages  
1880 -----  
1881 Prepared backporting package jdbc  
1882 ======  
1883 Preparing backporting package jenkins  
1884 -----  
1885 Prepared backporting package jenkins  
1886 ======  
1887 Preparing backporting package jira  
1888 -----  
1889 Prepared backporting package jira  
1890 ======  
1891 Preparing backporting package microsoft.azure  
1892 -----  
1893 Prepared backporting package microsoft.azure  
1894 ======  
1895 Preparing backporting package microsoft.mssql  
1896 -----  
1897 Prepared backporting package microsoft.mssql  
1898 ======  
1899 Preparing backporting package microsoft.winrm  
1900 -----  
1901 Prepared backporting package microsoft.winrm  
1902 ======  
1903 Preparing backporting package mssql
```

```
    ✓ Prepare & test backport packages  
2499 -----  
2500 Installing apache-airflow-backport-providers-microsoft-mssql  
2501 -----  
2502 Installed apache-airflow-backport-providers-microsoft-mssql  
2503 -----  
2504 Uninstalling apache-airflow-backport-providers-microsoft-mssql  
2505 -----  
2506 Uninstalled apache-airflow-backport-providers-microsoft-mssql  
2507 -----  
2508 Airflow version after installation 1.10.10  
2509 ======  
2510 Installing apache-airflow-backport-providers-microsoft-winrm  
2511 -----  
2512 Installed apache-airflow-backport-providers-microsoft-winrm  
2513 -----  
2514 Uninstalling apache-airflow-backport-providers-microsoft-winrm  
2515 -----  
2516 Uninstalled apache-airflow-backport-providers-microsoft-winrm  
2517 -----  
2518 Airflow version after installation 1.10.10  
2519 -----
```

Automated release notes for backport packages

Compatibility

For full compatibility and test status of the backport packages check [Airflow Backport Package Compatibility](#)

PIP requirements

PIP package	Version required
snowflake-connector-python	>=1.5.2
snowflake sqlalchemy	>=1.1.0

Provider class summary

All classes in Airflow 2.0 are in `airflow.providers.snowflake` package.

Operators

New operators

New Airflow 2.0 operators: `airflow.providers.snowflake` package

- `operators.s3_to_snowflake.S3ToSnowflakeTransfer`
- `operators.snowflake.SnowflakeOperator`

Hooks

New hooks

New Airflow 2.0 hooks: `airflow.providers.snowflake` package

- `hooks.snowflake.SnowflakeHook`

Releases

Release 2020.05.11

Commit	Committed	Subject
cd635dd7d	2020-05-10	[AIRFLOW-5906] Add authenticator parameter to snowflake_hook (#8642)
297ad3088	2020-04-20	Fix Snowflake hook conn id (#8423)
cf1109d66	2020-02-07	[AIRFLOW-6755] Fix snowflake hook bug and tests (#7300)

Compatibility

For full compatibility and test status of the backport packages check [Airflow Backport Package Compatibility](#)

PIP requirements

PIP package	Version required
paramiko	>=2.6.0
pysftp	>=0.2.9
sshtunnel	>=0.1.4,<0.2

Cross provider package dependencies

Those are dependencies that might be needed in order to use all the features of the package. You need to install the specified backport providers package in order to use them.

You can install such cross-provider dependencies when installing from PyPi. For example:

```
pip install apache-airflow-backport-providers-sftp[ssh]
```

Dependent package	Extra
apache-airflow-backport-providers-ssh	ssh

Provider class summary

All classes in Airflow 2.0 are in `airflow.providers.sftp` package.

Operators

Moved operators

Airflow 2.0 operators: <code>airflow.providers.sftp</code> package	Airflow 1.10.* previous location (usually <code>airflow.contrib</code>)
<code>operators.sftp.SFTPOperator</code>	<code>operators.sftp_operator.SFTPOperator</code>

Sensors

Moved sensors

Airflow 2.0 sensors: <code>airflow.providers.sftp</code> package	Airflow 1.10.* previous location (usually <code>airflow.contrib</code>)
<code>sensors.sftp.SFTPSensor</code>	<code>sensors.sftp_sensor.SFTPSensor</code>

Hooks

Support for Production Deployments



Production Image

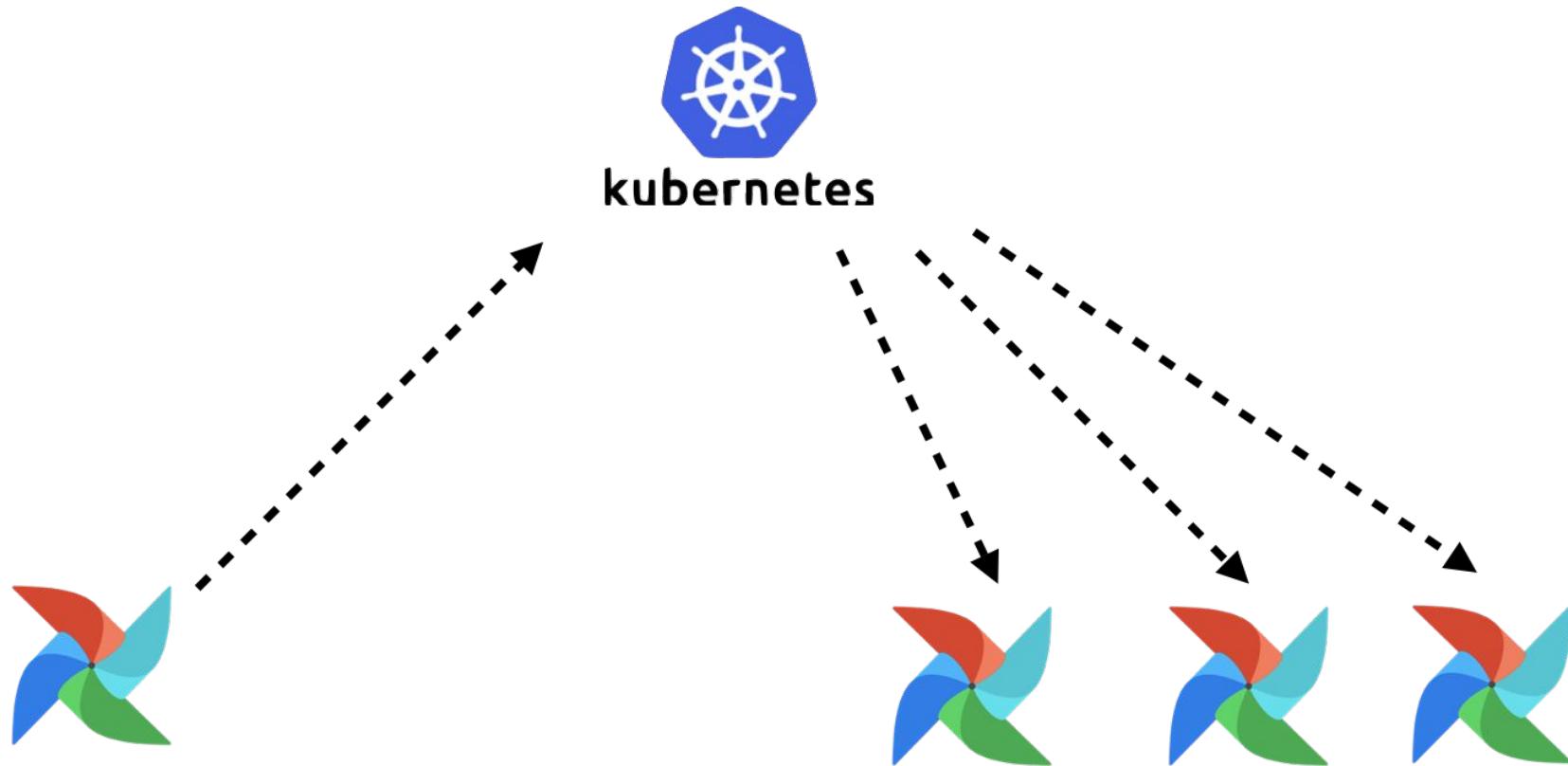
- Alpha quality image is ready
- Gathering feedback
- Started with “bare image”
- Listening to use cases from users
- Integration with Docker Compose
- Integration with Helm Chart

	Author	Label	Projects	Milestones	Assignee	Sort
12 Open ✓ 0 Closed						
<input type="checkbox"/> ⓘ docker image does not include 'pymssql' area:production-image kind:bug	#8712 opened 8 days ago by ishoni					2
<input type="checkbox"/> ⓘ User 'airflow' should be in group 'root' in the Docker image area:production-image kind:feature	#8706 opened 8 days ago by NBardelot					
<input type="checkbox"/> ⓘ Entrypoint does not take into account _CMD variables for DB and Broker configurations area:production-image kind:feature	#8705 opened 8 days ago by NBardelot					1
<input type="checkbox"/> ⓘ Production Docker Image AIRFLOW_INSTALL_VERSION does not overwrite AIRFLOW_VERSION environment variable area:production-image kind:bug	#8612 opened 14 days ago by ldacey					1
<input type="checkbox"/> ⓘ Deal with initial username/passwords in the production docker image area:production-image kind:feature	#8606 opened 14 days ago by potiuk					1
<input type="checkbox"/> ⓘ Add Production-ready docker compose for the production image area:production-image kind:feature	#8605 opened 14 days ago by potiuk					6
<input type="checkbox"/> ⓘ Add ADDITIONAL_PYTHON_DEPS to the image area:production-image kind:feature	#8604 opened 14 days ago by potiuk					
<input type="checkbox"/> ⓘ Describe contents of the production image for users area:docs area:production-image kind:documentation kind:feature	#8578 opened 15 days ago by potiuk					
<input type="checkbox"/> ⓘ Build production images from release tags area:production-image kind:feature	#8577 opened 15 days ago by potiuk					
<input type="checkbox"/> ⓘ Make production image pullable without tag (automatically) area:production-image kind:feature	#8574 opened 15 days ago by potiuk					1
<input type="checkbox"/> ⓘ [Umbrella] Add Docker compose example area:production-image kind:feature	#8548 opened 18 days ago by yordis					6
<input type="checkbox"/> ⓘ Production Docker Image - quick start area:production-image kind:documentation kind:feature	#8542 opened 18 days ago by mik-laj					2

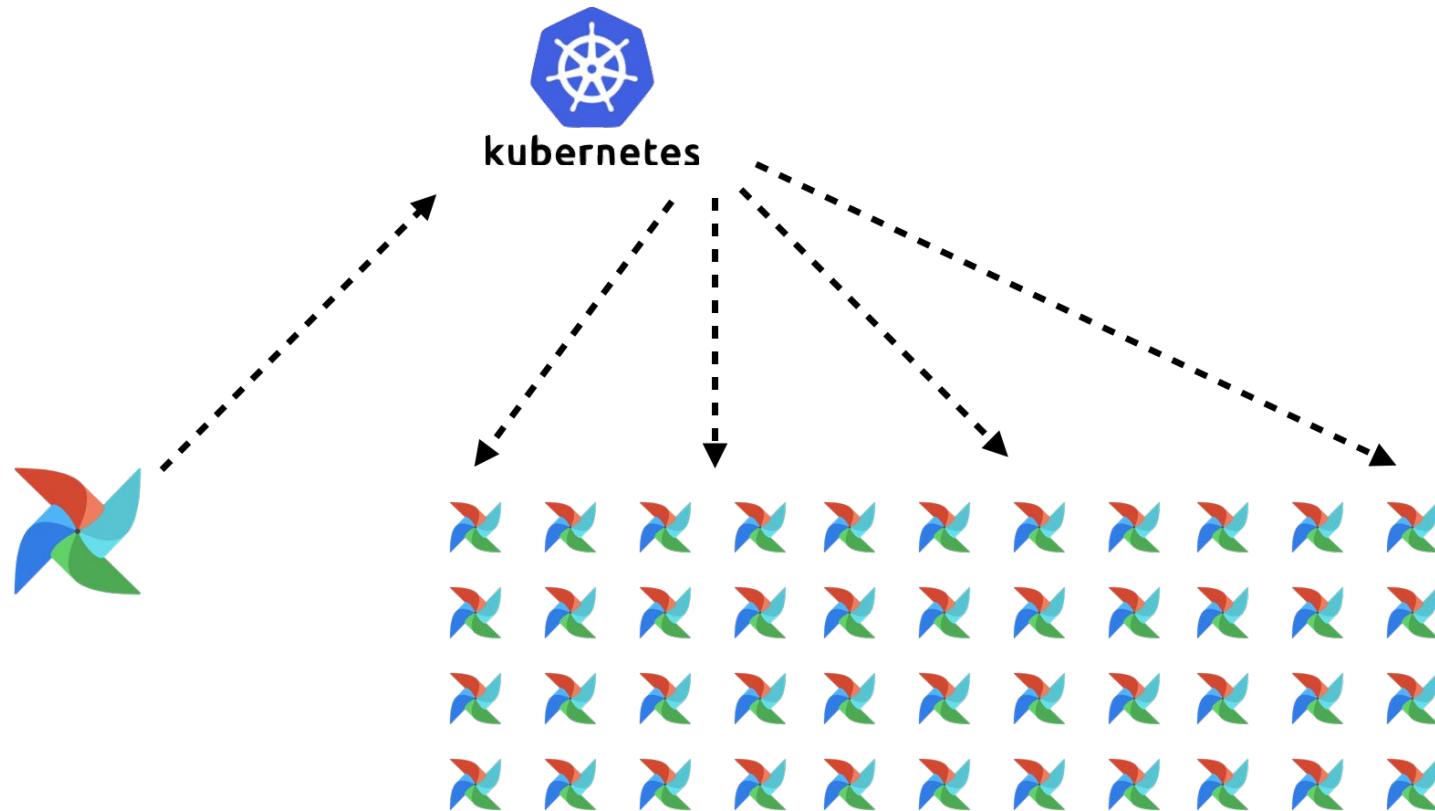
KEDA Autoscaling



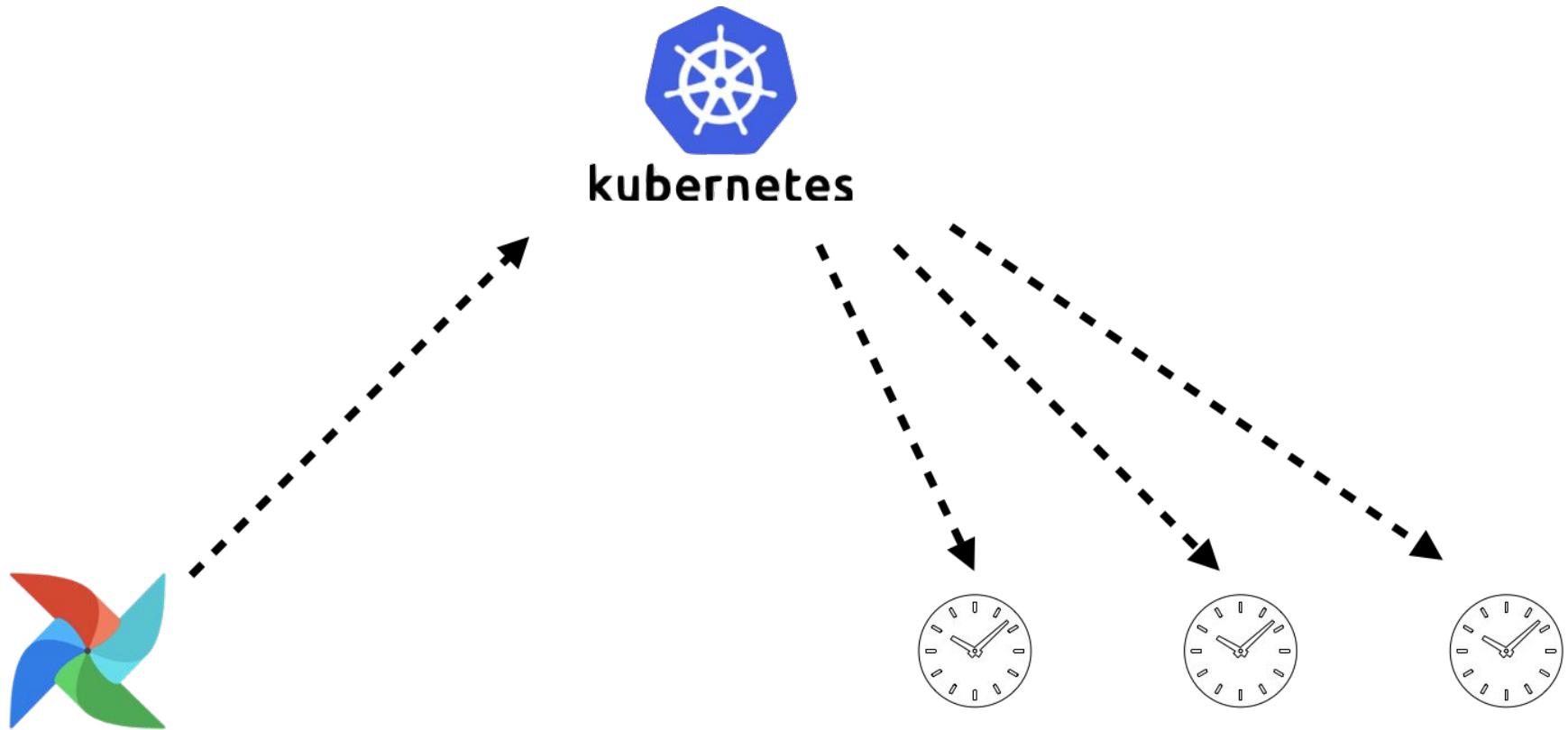
KubernetesExecutor



KubernetesExecutor



KubernetesExecutor



KubernetesExecutor vs. CeleryExecutor

KubernetesExecutor

- Dynamic Allocation
- executor_config

CeleryExecutor

- Immediate SLAs
- Multiple tasks per-worker

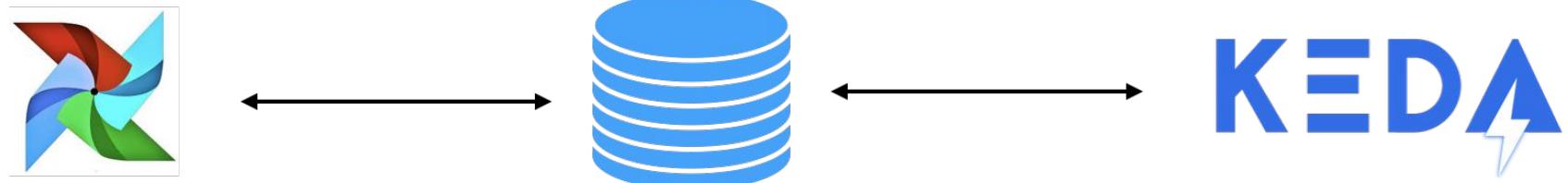
KEDA

A blue lightning bolt icon is positioned at the bottom right of the word "KEDA". The bolt is rendered with three distinct strokes of varying lengths, all pointing upwards and to the left.

KEDA Autoscaling

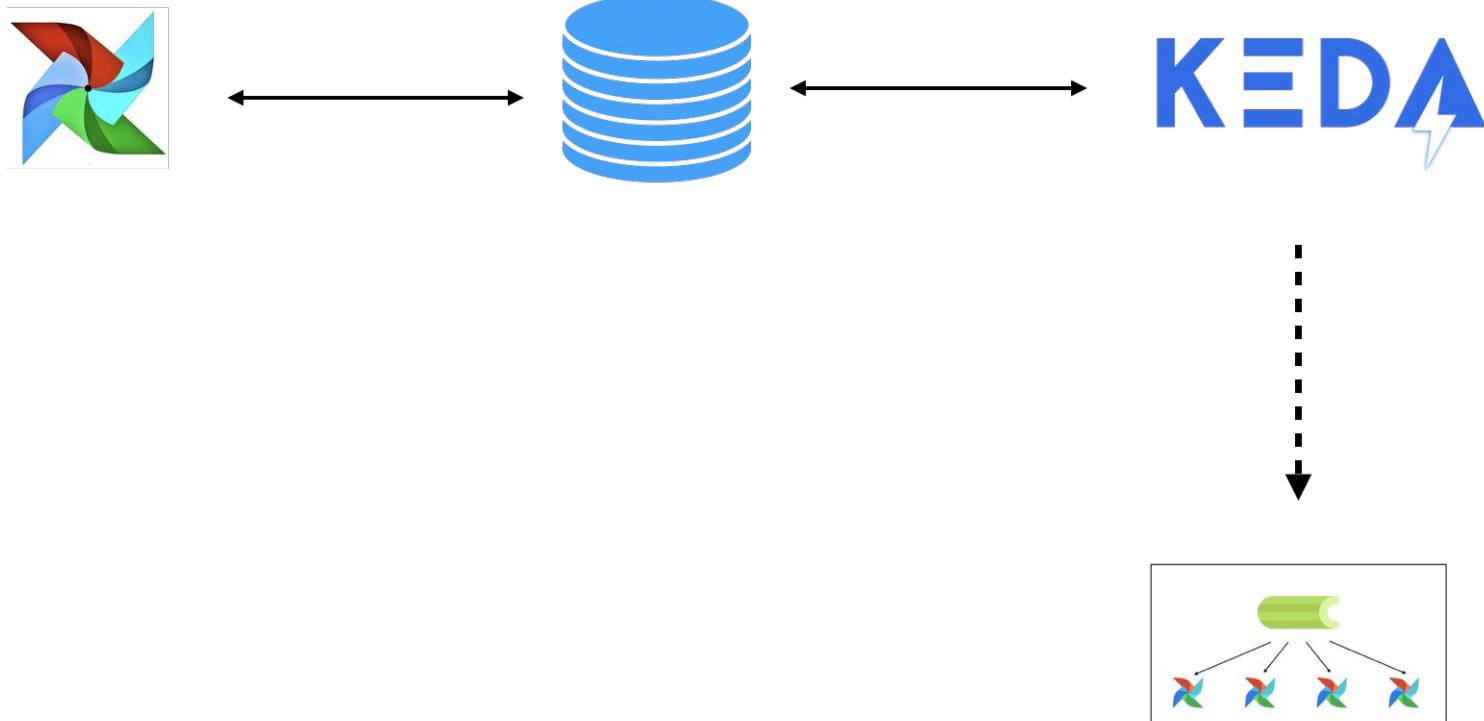
- Kubernetes Event-driven Autoscaler
- Scales based on # of RUNNING and QUEUED tasks in PostgreSQL backend

KEDA Autoscaling



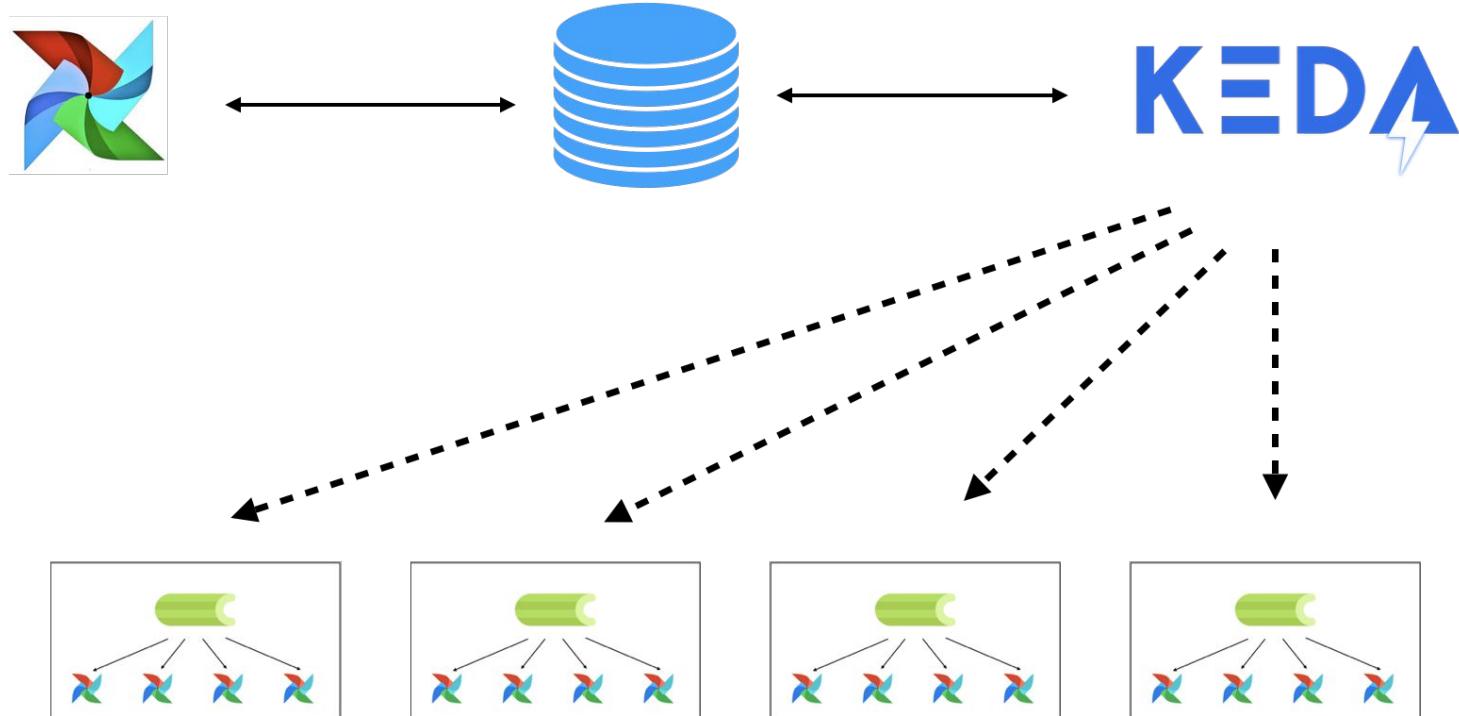
CEIL((0 RUNNING + 0 QUEUED)/16) = 0 workers

KEDA Autoscaling



CEIL((0 RUNNING + 1 QUEUED)/16) = 1 workers

KEDA Autoscaling



CEIL((20 RUNNING + 20 QUEUED)/16) = 4 workers

KEDA Queues

- Historically Queues were expensive and hard to allocate
- With KEDA, queues are free! (can have 100 queues)
- KEDA works with k8s deployments so any customization you can make in a k8s pod, you can make in a k8s queue (worker size, GPU, secrets, etc.)

KubernetesExecutor Pod Templating from YAML/JSON



KubernetesExecutor Pod Templating

- In the K8sExecutor currently, users can modify certain parts of the pod, but many features of the k8s API are abstracted away
- We did this because at the time the airflow community was not well acquainted with the k8s API
- We want to enable users to modify their worker pods to better match their use-cases

KubernetesExecutor Pod Templating

- Users can now set the `pod_template_file` config in their `airflow.cfg`
- Given a path, the KubernetesExecutor will now parse the yaml file when launching a worker pod
- Huge thank you to `@davlum` for this feature

Official Airflow Helm Chart



Helm Chart

- Donated by astronomer.io.
- This is the official helm chart that we have used both in our enterprise and in our cloud offerings (thousands of deployments of varying sizes)
- Users can turn on KEDA autoscaling through helm variables

Helm Chart

- Chart will cut new releases with each airflow release
- Will be tested on official docker image
- Significantly simplifies airflow onboarding process for Kubernetes users

DAG authoring "sugar"



Functional DAGs

```
● ● ●  
def get_cat_pictures(num: int) -> List[Dict]:  
    response = requests.get("https://cat_pictures.com", params={"num": num})  
    return response.json()["cats"]  
  
def save_cats(list_of_cats: List[Dict]) -> None:  
    for cat in list_of_cats:  
        save_it_somewhere(cat)  
  
with DAG("cat_fetcher"):  
    get_task = PythonOperator(  
        task_id="get_task", python_callable=get_cat_pictures, op_args=[42]  
    )  
    cats = "{{ task_instance.xcom_pull('get_task') }}"  
    save_task = PythonOperator(  
        task_id="save_task", python_callable=save_cats, op_args=[cats]  
    )  
    get_task >> save_task
```

- PythonOperator boilerplate code
- Define order and data relation separately
- Writing jinja strings by hand

Functional DAGs

```
● ● ●  
  
def get_cat_pictures(num: int) -> List[Dict]:  
    response = requests.get("https://cat_pictures.com", params={"num": num})  
    return response.json()["cats"]  
  
def save_cats(list_of_cats: List[Dict]) -> None:  
    for cat in list_of_cats:  
        save_it_somewhat(cat)  
  
with DAG("cat_fetcher"):  
    get_task = PythonOperator(  
        task_id="get_task", python_callable=get_cat_pictures, op_args=[42]  
    )  
    cats = "{{ task_instance.xcom_pull('get_task') }}"  
    save_task = PythonOperator(  
        task_id="save_task", python_callable=save_cats, op_args=[cats]  
    )  
    get_task >> save_task
```

No PythonOperator boilerplate code!

```
@task  
def get_cat_pictures(num: int) -> List[Dict]:  
    response = requests.get("https://cat_pictures.com", params={"num": num})  
    return response.json()["cats"]  
  
@task  
def save_cats(list_of_cats: List[Dict]) -> None:  
    for cat in list_of_cats:  
        save_it_somewhat(cat)  
  
with DAG("cat_fetcher"):  
    get_task = get_cat_pictures(42)  
    save_task = save_cats(get_task)
```

Data and order relationship are same!

And works for all operators

Functional DAGs

AIP-31: Airflow functional DAG definition

- Easy way to convert a function to an operator
- Simplified way of writing DAGs
- Pluggable XCom Storage engine

```
@task
def get_cat_pictures(num: int) -> List[Dict]:
    response = requests.get("https://cat_pictures.com", params={"num": num})
    return response.json()["cats"]

@task
def save_cats(list_of_cats: List[Dict]) -> None:
    for cat in list_of_cats:
        save_it_somewhere(cat)

with DAG("cat_fetcher"):
    get_task = get_cat_pictures(42)
    save_task = save_cats(get_task)
```

Example: store and retrieve DataFrames on GCS or S3 buckets without boilerplate code

Smaller changes



Other changes of note

- Connection IDs now need to be unique

It was often confusing, and there are better ways to do load balancing

- Python 3 only

Python 2.7 unsupported upstream since Jan 1, 2020

- "RBAC" UI is now the only UI.

Was a config option before, now only option. Charts/data profiling removed due to security risks

Road to Airflow 2.0



When will Airflow 2.0 be available?



Airflow 2.0 – deprecate, but (try) not to remove

- Breaking changes should be avoided where we can – if upgrade is too difficult users will be left behind
- Release "backport providers" to make new code layout available "now":

```
pip install apache-airflow-backport-providers-aws \
          apache-airflow-backport-providers-google
```

- Before 2.0 we want to make sure we've fixed everything we want to remove or break.

How to upgrade to 2.0 safely

- Install the latest 1.10 release
- Run `airflow upgrade-check` (doesn't exist yet)
- Fix any warnings
- Upgrade Airflow

Airflow Summit 2020



July 6th-17th, 2020.
Join from anywhere.
<https://airflowsummit.org>

Thank you!

Time for Q & A

