Common Airflow mistakes

Listed below are a few common mistakes users make when creating their first DAGs on Airflow.

Your DAG isn't running when you expect it to

One of the most common mistakes when scheduling a DAG in Airflow is understanding how the scheduler works with the `execution_date` and `start_date`.

Say you have a DAG that has a `schedule_interval` of `0 9 * * *` (i.e, runs at 09:00 everyday). You add this DAG to Airflow early in the morning of March 1st, 2021 expecting it to run at 9:00 am EST, but it doesn't. You decide to be patient and wait. Finally, the DAG runs at 5:00 am EST the next day, and its `execution_date` is `2021-03-01T09:00:00+00:00`. Why is that?

There are two important points to understand about Airflow's scheduler:

1. Airflow's scheduling is UTC based (by default). Your `schedule_interval` should be written with this in mind.
2. Airflow's scheduler will only execute a DAG (i.e, a DAG run) at the **end** of a schedule interval. This is meant to allow for the data to have been processed and/or imported for that schedule interval **before** the DAG runs. With this in mind, you should expect your first DAG run to start at `start_date+schedule_interval`

I scheduled my DAG, but it isn't running. What's up?

A common mistake Airflow users make when creating their DAG is setting the `start_date` to a dynamic value without understanding the implications.

As the Airflow scheduler performs its scheduling loop, it goes through each DAG and determines if a run should be created. If a DAG has previously been scheduled, the Airflow scheduler will use its previous DAG run(s) alongside its `schedule_interval` to determine if it needs to be scheduled. If the DAG has never been scheduled before, the Airflow scheduler will use its `start_date` instead (**Note:** these rules do not apply for manually triggered DAGs).

When creating the DAG, it is therefore important for the user to understand that the `start_date` will be computed every single time the scheduler performs its scheduling check. Failure to do so might lead to the DAG never running at all!

⚠ Airflow's scheduling mechanism is a lot more complex than what was described above. If you encounter any scheduling issues which are not described here, feel free to send us a chat in the support channel.

Here are a few examples of `start_date` values that could cause some issues:

Using `datetime.now()` as your DAG's `start_date`

```python
from datetime import datetime
...
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime.now()
}

dag = DAG(
    dag_id='dag_id',
    default_args=default_args,
    schedule_interval="* * * * *"
)
```

Using `datetime.now()` as your `start_date` will prevent your DAG from ever being scheduled. The reason behind this is the following:

As mentioned in another FAQ, a DAG's first run will need to satisfy the following condition to be scheduled: `start_date + schedule_interval < datetime.now()`. Furthermore, during each scheduler loop, the `datetime.now()` code will be ran and its value will be assigned to the `start_date`. This means the `start_date` will always have the value of '*now*' during the scheduler's date comparison. In our case, we know that `start_date = datetime.now()` which means that the condition will effectively be `datetime.now() + schedule_interval < datetime.now()`. This condition can never be satisfied which means your DAG will never run!

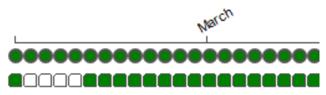
Using `days_ago(0)` as your DAG's start date

```
from airflow.utils.dates import days_ago
...
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': days_ago(0)
}

dag = DAG(
    dag_id='dag_id',
    default_args=default_args,
    schedule_interval="@weekly"
)
```

Every time the Airflow scheduler checks for DAGs to schedule, the code `days_ago(0)` will resolve to midnight of the current day (e.g. if today was April 6th, `start_date = 2021-04-06T00:00:00+00:00`). With the above `start_date`, the largest interval possible between the `start_date` and `utc.now()` during the scheduler's checks would be ~1 day. Since an interval of *one week* will never occur between those two dates, our DAG will never be scheduled.

I triggered my DAG and the run completes successfully, but no tasks are ran.



When attempting to run a DAG prior to its `start_date`, Airflow will automatically assign a `success` state to the DAG run without running its tasks (see white squares in the figure above). This is normal and can be avoided by either not triggering the DAG prior to the `start_date` or pushing back your `start_date` to an earlier point in time.

We recommend using a static date to avoid these types of situations.

Some of my DAG parameters don't seem to work (i.e. catchup, concurrency, max_active_runs)

Airflow accepts different types of configuration options to help configure your DAG's execution. It is important to put these options in the correct place otherwise they may be ignored and can lead to confusing scheduling issues.

There are several types of options you can configure within a DAG:

- DAG (DAG run) specific options;
- Task (Operator) specific options
- "Shared" options

DAG specific options

These values are passed directly to the DAG object and will help the scheduler determine when it needs to schedule a DAG run.

Example:

```
dag = DAG(
    dag_id='dag_id',
    ...,
    # DAG options
    schedule_interval='* * * * *',
    catchup=False,
    concurrency=1,
    max_active_runs=1
)
```

It is important to note that these values only apply to DAG runs and **must** be passed at the DAG level. (i.e. they cannot be passed through the *default_args* parameter).

Task specific options

These values can be passed in two ways:

- *default_args* at the DAG level;

```
default_args = {
    'owner': 'owner',
    'start_date': datetime(2021, 4, 20),

    # Task specific options
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    'end_date': datetime(2021, 4, 17),
    'wait_for_downstream': False,
    'sla': timedelta(hours=2),
    'execution_timeout': timedelta(seconds=300),
    'on_failure_callback': some_function,
    'on_success_callback': some_other_function,
    'on_retry_callback': another_function,
    'sla_miss_callback': yet_another_function,
    'trigger_rule': 'all_success'
}

dag = DAG(
    dag_id='dag_id',
    default_args=default_args,
    schedule_interval='* * * * *'
)
```

- directly in Task operators

```
python_task=PythonOperator(
    task_id='python_task',
    dag=dag,
    python_callable=some_python_function,

    # Task specific options
    retries=3,
    sla=timedelta(hours=4)
)
```

Default args are passed to each task operator at the time of its execution. Any parameters defined directly in the task operator will override these defaults.

“Shared” options

Although most options are specific to the DAG or the Task, some of them can be thought as ‘shared’. Options such as `owner` and `start_date` can both be used by a DAG and its tasks which can be quite confusing.

A bit of history - When it was created, Airflow did not have the concept of DAG runs and executed tasks individually by creating “task instances”. In these tasks, required parameters such as `owner` and `start_date` had to be passed every time. Over time, a feature was added so that users could specify a `default_args` parameter on the DAG to do this automatically. When the DAG run concept was added to Airflow (v1.6~), it allowed the Airflow developers to have the scheduling logic depend on what was defined at the DAG level as opposed to each task.

Emails aren't reaching my distribution list

Make sure your distribution list follows the following name pattern:

```
<team>-team_dl@cse-cst.gc.ca
```

Emails such as `<team>@cyber.gc.ca` don’t work. If your email issues persist, let us know in the Hogwarts support chat.