AIRFLOW SUMMIT

Deadline Alerts In Airflow 3.1

Dennis Ferruzzi | OSS Software Engineer @ AWS

Ramit Kataria | OSS Software Engineer @ AWS

AIRFLOW SUMMIT

Beyond SLA

Deadline Alerts in Airflow 3.1

3.0

Dennis Ferruzzi

Ramit Kataria

What was an SLA?

In Airflow 2, an SLA allowed users to set a duration for a Dag or Task and get a notification if they ran longer.

What is a Deadline Alert?

In Airflow 3, Deadline Alerts allow you to set time thresholds for your Dag runs, and automatically respond when those thresholds are exceeded.

That sounds awfully similar, what’s the difference?

SLA

- **sla_miss** was evaluated **only** if the task_id was ever in SUCCESS or SKIPPED state
- SLA was defined as a timedelta relative to the dagrun.data_interval_end
- Callback was an attribute of the **Dag**, but the SLA was an attribute of individual **tasks**
- Callback was executed after the Dag run finished
- Dag files were parsed **each time** the callback was sent to the DagFileProcessor
- DagFileProcessor was overloaded

Deadline Alerts

- Evaluated every scheduler pass
- You can define the **Reference** (starting point) and **Interval** (timedelta)
- Callback is an attribute of the **Deadline**
- Callback is executed within seconds of the missed deadline
- Deadline is calculated at Dag run creation and stored, no need for multiple passes
- DagFileProcessor can focus on its job

SLA

Deadline Alerts

SLA

User defines a length of time. When (if?) the Dag run finishes: if it took longer than that amount of time, run the callback.

More flexible and timely!

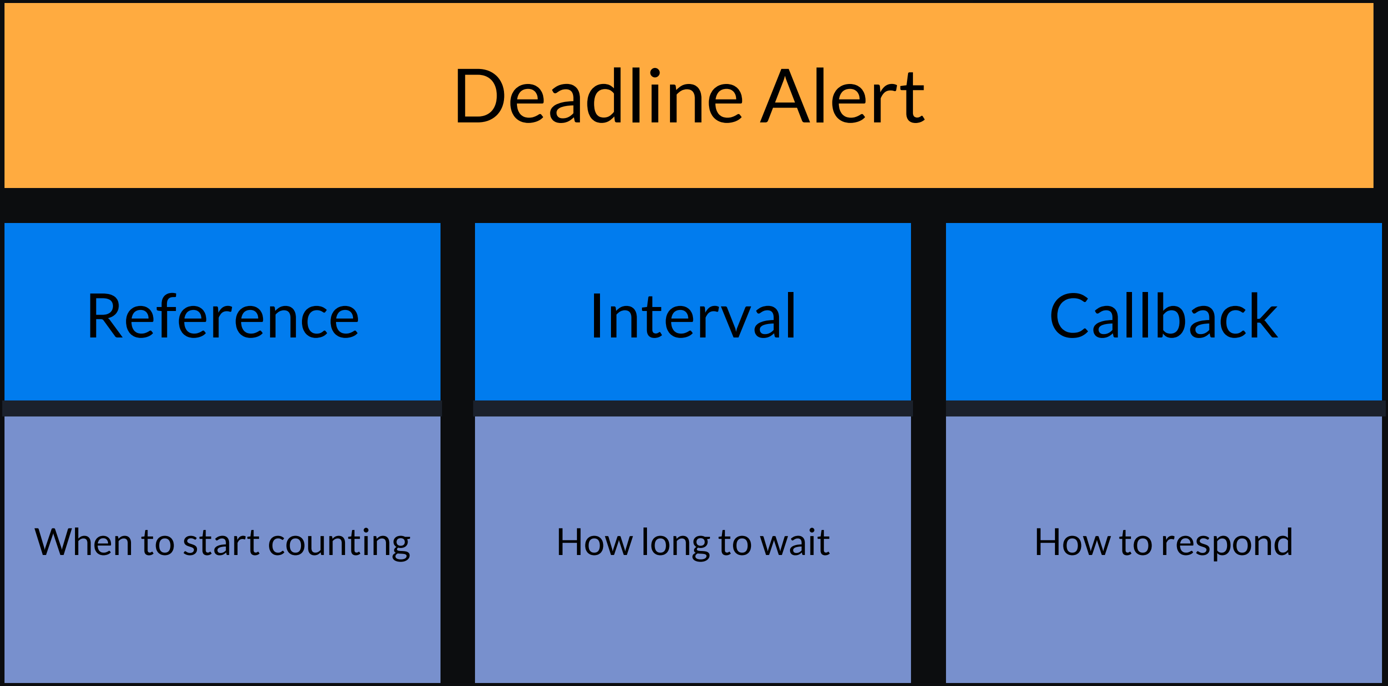
Deadline Alerts

User defines a length of time and a reference point from which to start counting. When a Dag run is created, calculate that expiration time.

Each scheduler pass (5 seconds by default), if that time has passed: run the callback, even if the Dag is still currently running.

I’m Sold!
How do Deadlines Work?

How is a Deadline Alert Defined?



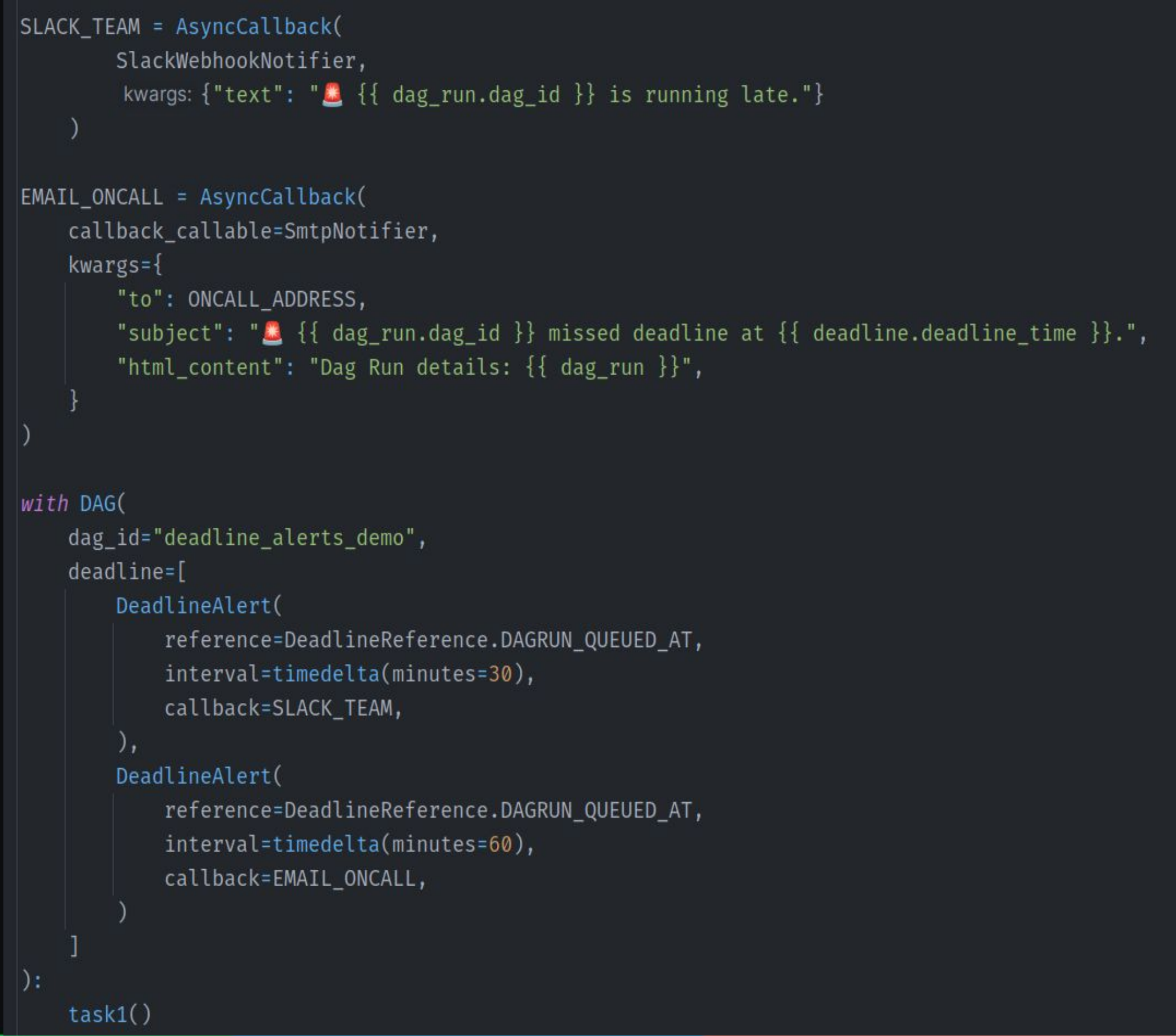
Deadline Alert

Reference	Interval	Callback
When to start counting	How long to wait	How to respond

What Does This Look Like?

If the DAGRUN has not finished
    [Interval] 30 minutes after
    [Reference] the DAGRUN_QUEUED_AT
Then
    [Callback] send a slack message

If the DAGRUN has still not finished
    [Interval] 60 minutes after
    [Reference] the DAGRUN_QUEUED_AT
Then
    [Callback] send an email message

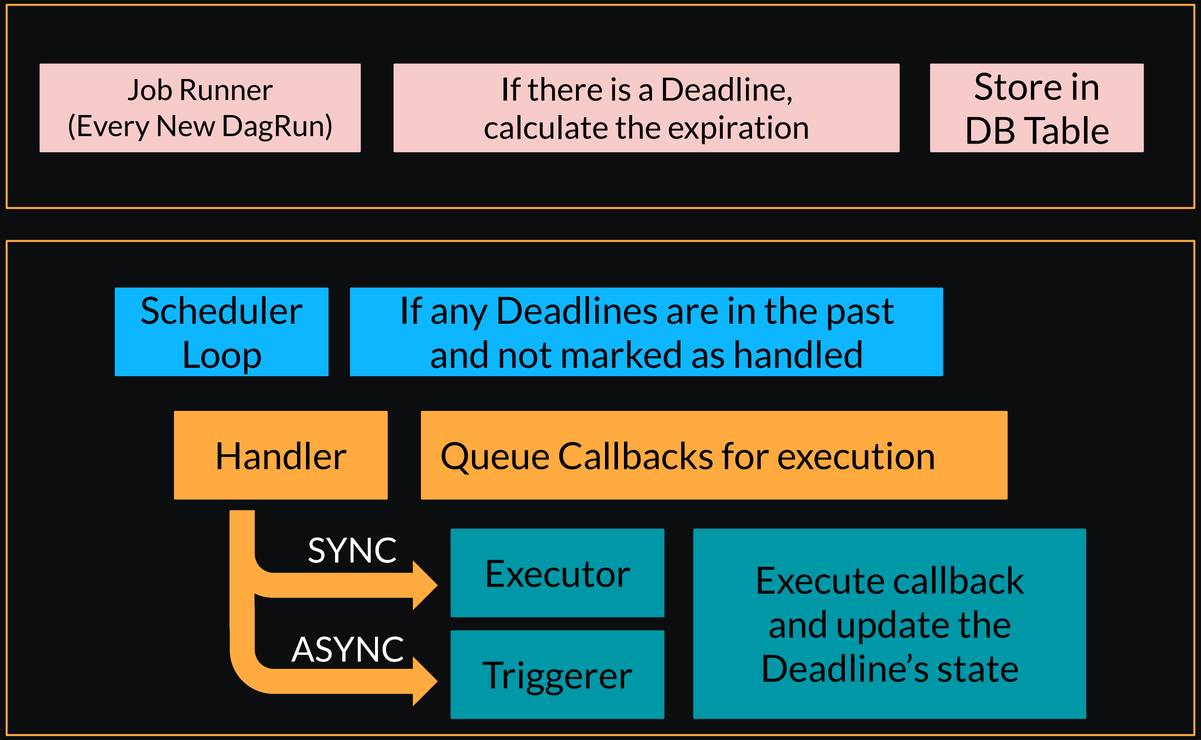


```python
SLACK_TEAM = AsyncCallback(
        SlackWebhookNotifier,
         kwargs: {"text": "🚨 {{ dag_run.dag_id }} is running late."}
    )

EMAIL_ONCALL = AsyncCallback(
    callback_callable=SmtpNotifier,
    kwargs={
        "to": ONCALL_ADDRESS,
        "subject": "🚨 {{ dag_run.dag_id }} missed deadline at {{ deadline.deadline_time }}.",
        "html_content": "Dag Run details: {{ dag_run }}",
    }
)

with DAG(
    dag_id="deadline_alerts_demo",
    deadline=[
        DeadlineAlert(
            reference=DeadlineReference.DAGRUN_QUEUED_AT,
            interval=timedelta(minutes=30),
            callback=SLACK_TEAM,
        ),
        DeadlineAlert(
            reference=DeadlineReference.DAGRUN_QUEUED_AT,
            interval=timedelta(minutes=60),
            callback=EMAIL_ONCALL,
        )
    ]
):
    task1()
```

How Does It Work?



Job Runner
(Every New DagRun)

If there is a Deadline,
calculate the expiration

Store in
DB Table

Scheduler
Loop

If any Deadlines are in the past
and not marked as handled

Handler

Queue Callbacks for execution

SYNC

ASYNC

Executor

Triggerer

Execute callback
and update the
Deadline's state

Built-in Deadline References

DAGRUN_QUEUED_AT	DAGRUN_LOGICAL_DATE	FIXED_DATETIME	AVERAGE_RUNTIME
Measures time from when the DagRun was queued.	Measures time from when the Dag run was scheduled to start.	Specifies a fixed point in time.	Calculates the average historical runtime of the Dag.
Useful for monitoring resource constraints.	Useful for ensuring scheduled Dags complete before their next scheduled run.	Useful when Dags must complete by a specific time.	Useful for detecting unusual activity in a Dag run or environment.

Callback Support

Asynchronous Callbacks (Available in 3.1)	Synchronous Callbacks (Coming in 3.2)
Run in the Triggerer	Run in the Executor/worker
Existing Notifiers - Async Notifiers Work Custom Callbacks - Must be in the Triggerer’s sys.path (for example, the Plugins folder)	Existing Notifiers - All Work Custom Callbacks - Can be placed anywhere in the Dag Bundle

Async vs Sync Callbacks

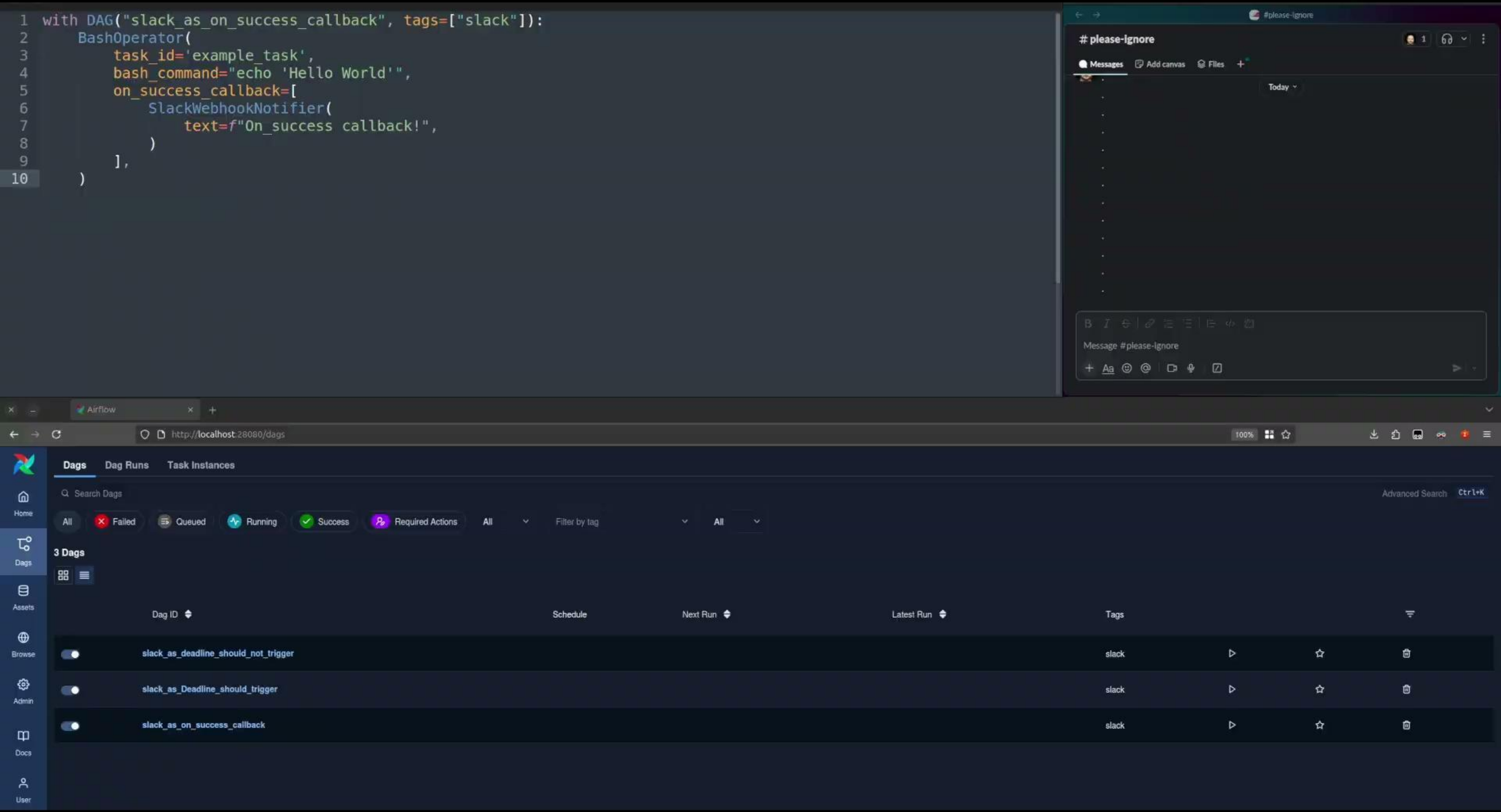
Asynchronous Callbacks (Available in 3.1)	Synchronous Callbacks (Coming in 3.2)
Lower runtime overhead - runs on triggerer	Higher runtime overhead - runs on worker/executor
Requires restarting the triggerer to apply changes in callback	Automatically uses the callback definition from the Dag bundle
Requires async callable	Runs any python callable

Callback Support

Asynchronous Callbacks	Synchronous Callbacks
Existing Notifiers with async support: Slack, Email, AWS (SES, SNS, SQS) + more coming soon	All existing Notifiers
Custom Callbacks - Must be in the Triggerer’s sys.path (for example, the Plugins folder)	Custom Callbacks - Can be placed anywhere in the Dag Bundle

DEMO!

3.0



```
with DAG("slack_as_on_success_callback", tags=["slack"]):
    BashOperator(
        task_id='example_task',
        bash_command="echo 'Hello World'",
        on_success_callback=[
            SlackWebhookNotifier(
                text=f"On success callback!",
            )
        ],
    )
```

3 Dags

Dag ID	Schedule	Next Run	Latest Run	Tags
slack_as_deadline_should_not_trigger				slack
slack_as_Deadline_should_trigger				slack
slack_as_on_success_callback				slack

What’s To Come?

Available in 3.1

- DAG-level Deadlines
  - Dagrun: Started
  - Dagrun: Queued
  - Fixed datetime (every day at 9AM)
  - Average runtime
- Multiple Deadlines per DAG
- Async Callbacks and Notifiers
  - Executed by the Triggerer

Future Work

- Synchronous Callbacks
  - Pick your Executor!
- Task-Level Deadlines
- Expand trigger options:
  - Dataset: Created
  - Dataset: Updated
  - Asset-Driven
  - ???
- Add more Asynchronous Notifiers

Questions?

3.0

Get Involved

@Ferruzzi

@Ramit Kataria

Slide Deck