

# Rerunning a failed job

- My DAG run has failed. What do I do?
- Airflow state
- Step-by-step
  - Rerunning a failed Dag run or task using the Tree view
  - Rerunning a failed task using the Graph view

## My DAG run has failed. What do I do?

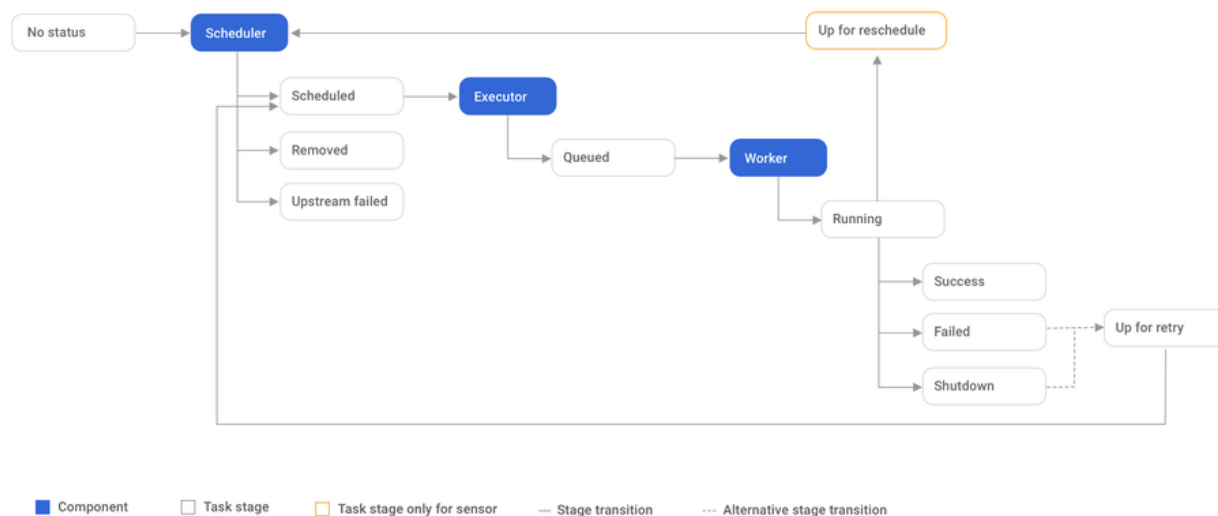
So you noticed that one of your DAG runs has failed, and now you are unsure on how to proceed. This page will describe the state flow processes in Airflow and provide you a few Step-by-step guides to get your tasks running again!

## Airflow state

Your task now has a *failed* status. But how did it get there? What was its initial and/or previous state?

As a way to identify the status of a DAG run or task, Airflow defines various states. These states have certain limitations:

- There can be no dual states. A DAG run or a task can only have one state.
- A state has a predefined lifecycle. See the illustration below for a task's lifecycle.



**Note :** Clearing a task's state is the only way to bring it back to 'No status', which isn't depicted in the above diagram.

What are these various Airflow states?

- **DAG runs** : running, success, failed.
- **Tasks** : none, scheduled, skipped, upstream\_failed, queued, success, failed, up\_for\_retry, up\_for\_reschedule.

**State definitions:**

▼ running

The DAG run or task is currently executing and has not yet reached a final state.

▼ success (finished state)

The DAG run or task ran successfully without encountering any errors.

▼ failed (finished state)

**DAG run** : A task within the DAG run has failed.

**Task** : An error occurred while executing the task. The task has exceeded its maximum amount of retries and cannot be rescheduled. Furthermore, all downstream tasks will have their state set to *upstream\_failed*.

▼ upstream\_failed (finished state)

An error occurred in one of the upstream tasks. This task cannot be scheduled as its dependencies cannot be met.

Using options such as *one\_success* can provide more flexibility when expecting some tasks to fail upstream.

▼ skipped (finished state)

This task has been ignored by the scheduler. This can be done manually or programmatically through branch operations. It's important to note that all downstream tasks will be set to skipped as well.

▼ up\_for\_retry

A task's attempt has failed but it has not exceeded the maximum amount of retries. Its state is set to *up\_for\_retry* to indicate to the scheduler that it can be reschedule for its next attempt.

▼ up\_for\_reschedule

A special state used by sensors (a type of operator) to prevent sensor tasks from blocking a worker slot during their poke operation.

Sensors used to take up a worker slot when running their poke operation. If you have many sensors running every X seconds, this used to cause "sensor deadlocks" where all worker slots were in use by sensors. (*This doesn't necessarily apply to our Airflow strategy as we use the KubernetesExecutor*).

▼ queued

A task is in queue waiting to be processed by the executor. The order in which a task is picked up by the executor may depend on its priority.

▼ no\_status

Newly created tasks will default to *no\_status*, which simply means that the task has just been created and is yet to be handled by the Airflow scheduler.

▼ scheduled

The Airflow scheduler will set a task's state to scheduled if it meets certain conditions (worker slots, etc.). Once a task's state is set to *scheduled*, it will be sent to the executor where it will eventually be queued and executed.

Understanding the differences between each state is the key to mastering Airflow DAG development. It will allow you to go further with your DAGs by using branching and/or flexible task dependencies. Furthermore, knowing the various task states will help you identify if your DAG run is going smoothly or if its encountering any issues.




## Step-by-step

So you now know which of your tasks have failed. You've identified the culprit and corrected the underlying code (ex: GitPythonOperator). You're now ready to rerun your task.

## Rerunning a failed Dag run or task using the Tree view

**i** Does your DAG have many tasks? We recommend using the Graph view for better usability (see *the next guide*).

1. Navigate to the Airflow where your DAG run has failed (unclass/PB).
2. On the home page, locate your DAG in the list either by using the DAG column or the search bar. Once you have found your DAG, click the 'Tree view' icon under the 'Links' column.

DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
<input checked="" type="checkbox"/> On <code>_test</code>	<code>example</code>	1 day 0:00:00	airflow, me		2020-10-15 00:00:00 <b>1</b> <b>1</b>	      

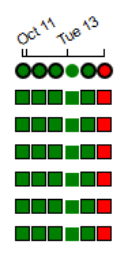
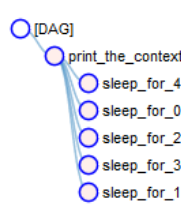
3. From the DAG's Tree view, locate the execution that you wish to rerun (*each column represents a DAG run*). Hovering over individual shapes will provide more details about the execution.

Graph View **Tree View** Task Duration Task Tries Landing Times Gantt Details

Code Trigger DAG Refresh Delete

Base date:  Number of runs:

☐ PythonOperator ☐ scheduled ☐ skipped ☐ upstream\_failed ☐ up\_for\_reschedule ☐ up\_for\_retry ☐ failed ☐ success ☐ running ☐ queued ☐ no\_status



4. Once you identified the column that corresponds to your execution date,
  - i. To clear a **DAG run** entirely, click the circle at the top of your execution column.

Click on 'Clear' to reset your DAG run's state.

**\_test**

2020-10-15T00:00:00Z Number of runs: 25

ii. To clear a specific **task's** state, find and click the corresponding task square.

Under the 'Clear' button, make sure that 'Downstream' and 'Recursive' are *pressed*. This will make sure all downstream tasks have their state cleared. Click the 'Clear' button once you are ready.

The screenshot shows a modal window for the task instance 'print\_the\_context' on 2020-10-15T00:00:00+00:00. At the top, there are four buttons: 'Task Instance Details' (active), 'Rendered', 'Task Instances', and 'View Log'. Below these is a section 'Download Log (by attempts):' with a single attempt '1'. The next row contains four buttons: 'Run', 'Ignore All Deps', 'Ignore Task State', and 'Ignore Task Deps'. Below that is a 'Clear' button followed by a row of six buttons: 'Past', 'Future', 'Upstream', 'Downstream', 'Recursive', and 'Failed'. The next row has a 'Mark Failed' button followed by 'Past', 'Future', 'Upstream', and 'Downstream'. The final row has a 'Mark Success' button followed by 'Past', 'Future', 'Upstream', and 'Downstream'. A 'Close' button is at the bottom right.

5. Your task(s) will soon be picked up by the scheduler to be rerun.

## Rerunning a failed task using the Graph view

You cannot clear a DAG run's state from the Graph view - only a task's state.

1. Navigate to the Airflow where your DAG run has failed (unclass/PB).
2. On the home page, locate your DAG in the list either by using the DAG column or the search bar. Once you have found your DAG, click the 'Graph view' icon under the 'Links' column.

DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
test	example 1 day, 0:00:00	airflow, me		2020-10-15, 00:00:00		

3. In the Graph view, select the execution date you wish to view in the *Run* dropdown then click the 'Go' button.

failed Base date: 2020-10-15T00:00:01Z Number of runs: 25 Run: scheduled\_2020-10-15T00:00:00+00:00 Layout: Left->Right Go

Search for...

PythonOperator

scheduled
skipped
upstream\_failed
up\_for\_reschedule
up\_for\_retry
failed
success
running
queued
no\_status

- In the updated Graph view, click on the task you wish to rerun. **Note:** Each task is represented by a node in the graph view.
- Under the 'Clear' button, make sure that 'Downstream' and 'Recursive' are *pressed*. This will make sure all downstream tasks have their state cleared. Click the 'Clear' button once you are ready.

print\_the\_context on 2020-10-15T00:00:00+00:00

Task Instance Details Rendered Task Instances View Log

Download Log (by attempts):

1

Run Ignore All Deps Ignore Task State Ignore Task Deps

Clear

Past Future Upstream Downstream Recursive Failed

Mark Failed Past Future Upstream Downstream

Mark Success Past Future Upstream Downstream

Close

- Your task will soon be picked up by the scheduler to be rerun.

