

Hands-on Lab: Monitoring a DAG

Estimated time needed: **20** minutes

Objectives

After completing this lab you will be able to:

- Search for a DAG.
- Pause/Unpause a DAG.
- Get the Details of a DAG.
- Explore grid view of a DAG.
- Explore graph view of a DAG.
- Explore Calendar view of a DAG.
- Explore Task Duration view of a DAG.
- Explore Details view of a DAG.
- View the source code of a DAG.
- Delete a DAG.

About Skills Network Cloud IDE

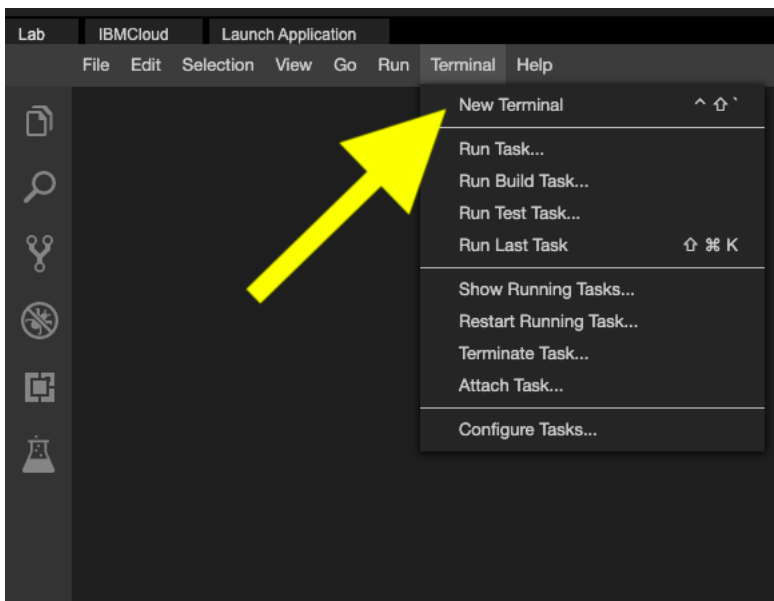
Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open source IDE (Integrated Development Environment), that can be run on desktop or on the cloud. to complete this lab, we will be using the Cloud IDE based on Theia running in a Docker container.

Important Notice about this lab environment

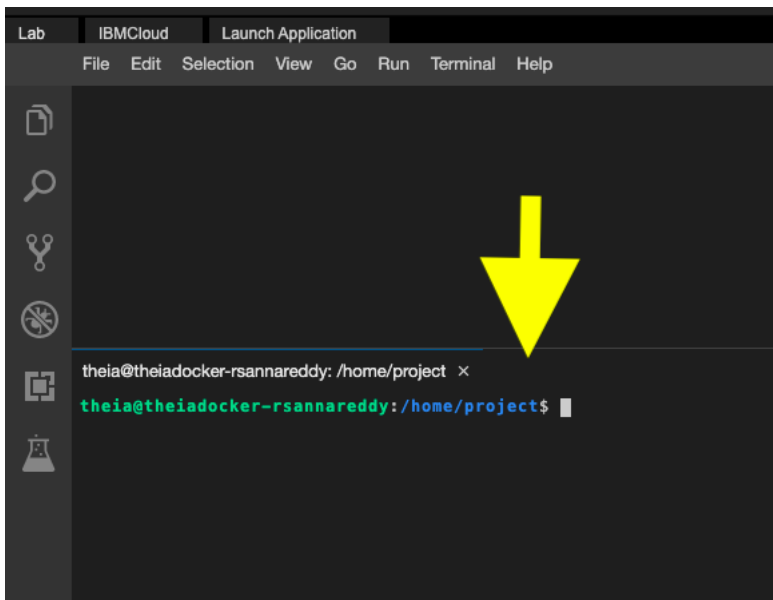
Please be aware that sessions for this lab environment are not persistent. A new environment is created for you every time you connect to this lab. Any data you may have saved in an earlier session will get lost. To avoid losing your data, please plan to complete these labs in a single session.

Exercise 1 - Getting the environment ready

Step 1.1. Open a new terminal by clicking on the menu bar and selecting **Terminal->New Terminal**, as shown in the image below.



This will open a new terminal at the bottom of the screen.



Run the commands below on the newly opened terminal. (You can copy the code by clicking on the little copy button on the bottom right of the codeblock below and then paste it, wherever you wish.)

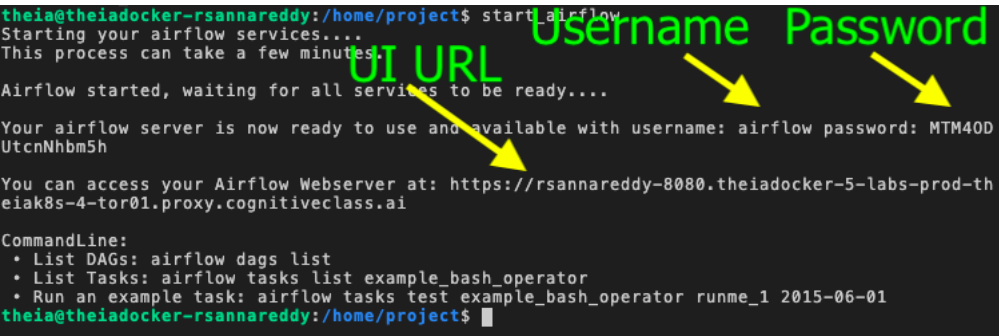
Start Apache Airflow in the lab environment.

1. 1
1. start_airflow

Copied!

Please be patient, it will take a few minutes for airflow to get started.

When airflow starts successfully, you should see an output similar to the one below:



Step 1.2. Open the Airflow Web UI

Copy the Web-UI URL and paste it on a new browser tab. Or you can click on the URL by holding the control key (Command key in case of a Mac).

You should land at a page that looks like this:

Skills Network Airflow

<div> <div>All 32</div> <div>Active 0</div> <div>Paused 32</div> <div>Filter DAGs by tag</div> </div>						
DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	
<div><input type="checkbox"/></div> <div>example_bash_operator</div> <div>example example2</div>	airflow	<div><div></div><div></div><div></div></div>	0 0 ***	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
<div><input type="checkbox"/></div> <div>example_branch_datetime_operator_2</div> <div>example</div>	airflow	<div><div></div><div></div><div></div></div>	@daily	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
<div><input type="checkbox"/></div> <div>example_branch_dop_operator_v3</div> <div>example</div>	airflow	<div><div></div><div></div><div></div></div>	* / 1 * * * *	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
<div><input type="checkbox"/></div> <div>example_branch_labels</div>	airflow	<div><div></div><div></div><div></div></div>	@daily	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
<div><input type="checkbox"/></div> <div>example_branch_operator</div> <div>example example2</div>	airflow	<div><div></div><div></div><div></div></div>	@daily	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
<div><input type="checkbox"/></div> <div>example_complex</div> <div>example example2 example3</div>	airflow	<div><div></div><div></div><div></div></div>	None	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
<div><input type="checkbox"/></div> <div>example_dag_decorator</div> <div>example</div>	airflow	<div><div></div><div></div><div></div></div>	None	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
<div><input type="checkbox"/></div> <div>example_external_task_marker_child</div> <div>example2</div>	airflow	<div><div></div><div></div><div></div></div>	None	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>	

Exercise 2 - Submit a dummy DAG

For the purpose of monitoring, let's create a dummy DAG with three tasks.

Task1 does nothing but sleep for 1 second.

Task2 sleeps for 2 seconds.

Task3 sleeps for 3 seconds.

This DAG is scheduled to run every 1 minute.

Step 2.1. Using Menu->File->New File create a new file named dummy_dag.py.

Step 2.2. Copy and paste the code below into it and save the file.

```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37

```

```

38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46
47. 47
48. 48
49. 49
50. 50
51. 51
52. 52
53. 53
54. 54
55. 55
56. 56
57. 57

1. # import the libraries
2.
3. from datetime import timedelta
4. # The DAG object; we'll need this to instantiate a DAG
5. from airflow import DAG
6. # Operators; we need this to write tasks!
7. from airflow.operators.bash_operator import BashOperator
8. # This makes scheduling easy
9. from airflow.utils.dates import days_ago
10.
11. #defining DAG arguments
12.
13. # You can override them on a per-task basis during operator initialization
14. default_args = {
15.     'owner': 'Ramesh Sannareddy',
16.     'start_date': days_ago(0),
17.     'email': ['ramesh@somemail.com'],
18.     'email_on_failure': False,
19.     'email_on_retry': False,
20.     'retries': 1,
21.     'retry_delay': timedelta(minutes=5),
22. }
23.
24. # defining the DAG
25. dag = DAG(
26.     'dummy_dag',
27.     default_args=default_args,
28.     description='My first DAG',
29.     schedule_interval=timedelta(minutes=1),
30. )
31.
32. # define the tasks
33.
34. # define the first task
35.
36. task1 = BashOperator(
37.     task_id='task1',
38.     bash_command='sleep 1',
39.     dag=dag,
40. )
41.
42. # define the second task
43. task2 = BashOperator(
44.     task_id='task2',
45.     bash_command='sleep 2',
46.     dag=dag,
47. )
48.
49. # define the third task
50. task3 = BashOperator(
51.     task_id='task3',
52.     bash_command='sleep 3',
53.     dag=dag,
54. )
55.
56. # task pipeline
57. task1 >> task2 >> task3

```

Copied!

Submitting a DAG is as simple as copying the DAG python file into dags folder in the AIRFLOW_HOME directory.

Step 2.3. Open a terminal and run the command below to submit the DAG that was created in the previous exercise.

```

1. 1
1. cp dummy_dag.py $AIRFLOW_HOME/dags

```

Copied!

Step 2.4. Verify that our DAG actually got submitted.

Run the command below to list out all the existing DAGs.

```

1. 1
1. airflow dags list

```

Copied!

Verify that dummy_dag is a part of the output.

Step 2.5. Run the command below to list out all the tasks in dummy_dag.

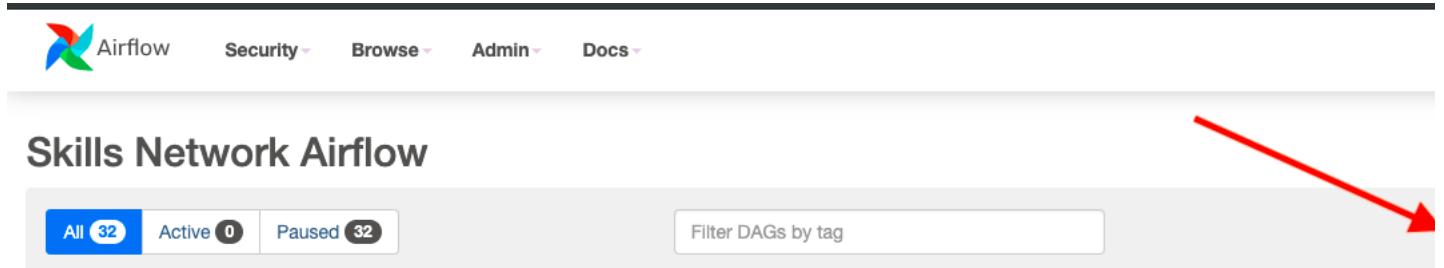
- 1
1. `airflow tasks list dummy_dag`

Copied!

You should see 3 tasks in the output.

Exercise 3 - Search for a DAG

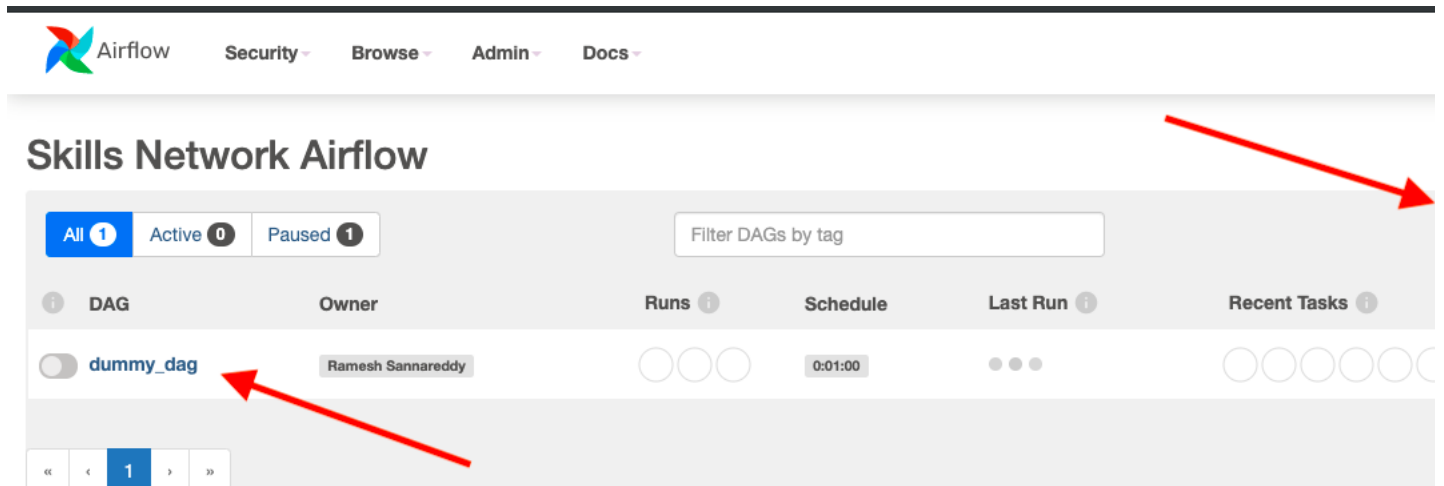
In the Web-UI, identify the Search DAGs text box as shown in the image below.



Type dummy_dag in the text box and press enter.

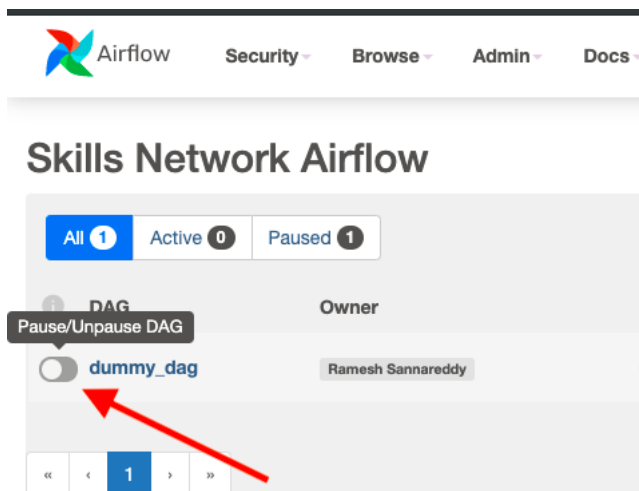
Note: It may take a couple of minutes for the dag to appear here. If you do not see your DAG, please give it a minute and try again.

You should see the dummy_dag listed as seen in the image below:



Exercise 4 - Pause/Unpause a DAG

Unpause the DAG using the Pause/Unpause button.



You should see the status as shown in the image below after you unpause the DAG.

Skills Network Airflow

All 1Active 1Paused 0

Filter DAGs by tag

DAG	Owner	Runs	Schedule	Last Run	Recent Tasks
dummy_dag	Ramesh Sannareddy	9	0:01:00	2021-07-27, 00:08:00	8 4

«

«

1

>

»

You can see the following details in this view.

- Owner of the DAG
- How many times this DAG has run.
- Schedule of the DAG
- Last run time of the DAG
- Recent task status.

Exercise 5 - DAG - Detailed view

Click on the DAG name as shown in the image below to see the detailed view of the DAG.

Skills Network Airflow

All 1Active 1Paused 0

Filter DAGs by tag

DAG	Owner
dummy_dag	Ramesh Sannareddy
My first DAG	

«

«

1

>

»

You will land a page that looks like this.

Airflow

SecurityBrowseAdminDocs

DAG: dummy_dag

My first DAG

Schedule: 0

GridGraphCalendarTask DurationTask TriesLanding TimesGanttDetailsCodeAudit

04/01/2023, 10:42:02 am

25

All Run Types

All Run States

Clear Filters

deferredfailedqueuedrunningscheduledskippedsuccessup_for_resc

Auto-refresh

DAG dummy_dag

DAG Details

DAG Summary

Total Tasks

task1

task2

task3

Exercise 6 - Explore Grid view of DAG

Click on the Grid View button to open the Grid view.

Airflow

Security

Browse

Admin

Docs

DAG: dummy_dag

My first DAG

Grid

Graph

Calendar

Task Duration

Task Tries

Landing Times

04 / 01 / 2023 , 11 : 40 : 04 am

25

All Run Types

All Run States

Clear Filters

deferred

failed

queued

running

scheduled

skipped

success

up_for_reschedule

Click on the Auto Refresh button to switch on the auto refresh feature.

The Grid view shows your DAG tasks in the form of grids as seen in the image.

It also shows the DAG run and task run status as seen below.

04 / 01 / 2023 , 11 : 42 : 11 am

25

All Run Types

All Run States

Clear Filters

deferred

failed

queued

running

scheduled

skipped

success

up_for_reschedule

Auto-refresh

task1

task2

task3

Duration

00:00:43

00:00:21

00:00:00

Jan 04, 00:47

Jan 04, 00:57

Jan 04, 01:07

DAG

dummy_dag

DAG Details

DAG Runs Summary

Total Runs Displayed25

Total success9

Total running16

First Run Start2023-01-04, 11:40:04 am

Last Run Start2023-01-04, 11:42:11 am

The grids in the image below represent a single DAG run and the color indicates the status of the DAG run. Place your mouse on any grid to see the details.

Auto-refresh

task1

task2

task3

Duration

00:00:42

00:00:21

00:00:00

Jan 10, 00:35

Jan 10, 00:55

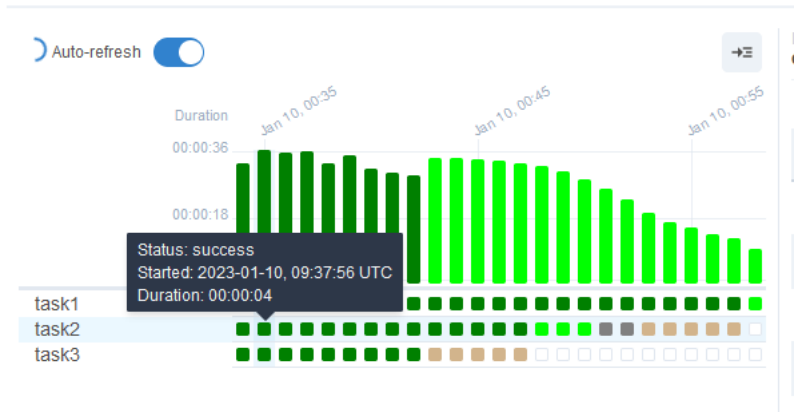
Status: running

Run: 2023-01-10, 00:45:00 UTC

Duration: 00:00:42

Type: scheduled

The squares in the image below represent a single task within a DAG run and the color indicates its status. Place your mouse on any square to see the task details.



Exercise 7 - Explore graph view of DAG

Click on the Graph View button to open the graph view.

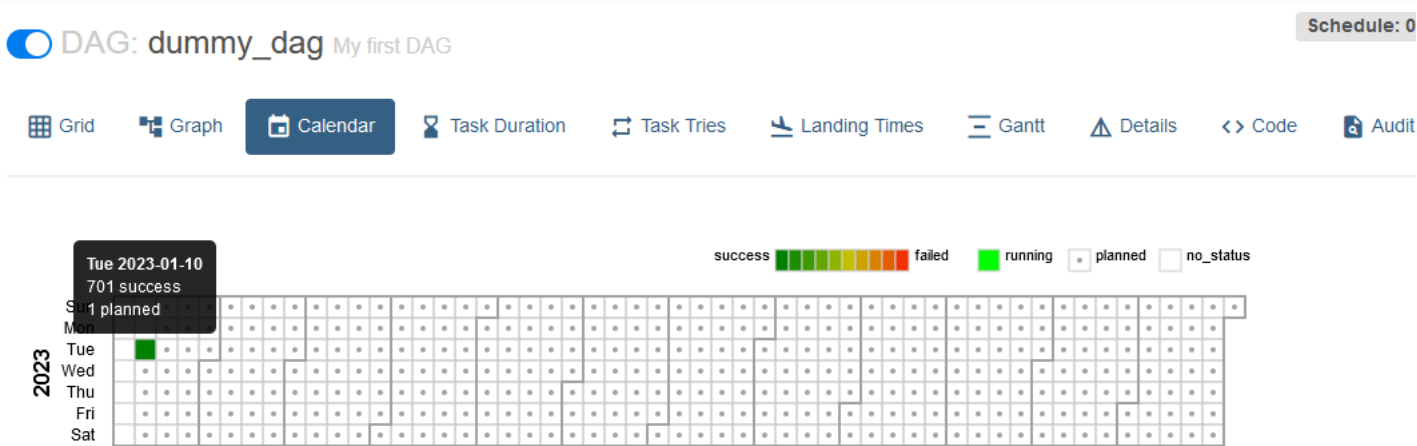
Click on the Auto Refresh button to switch on the auto refresh feature.

The graph view shows the tasks in a form of a graph. With the auto refresh on, each task status is also indicated with the color code.



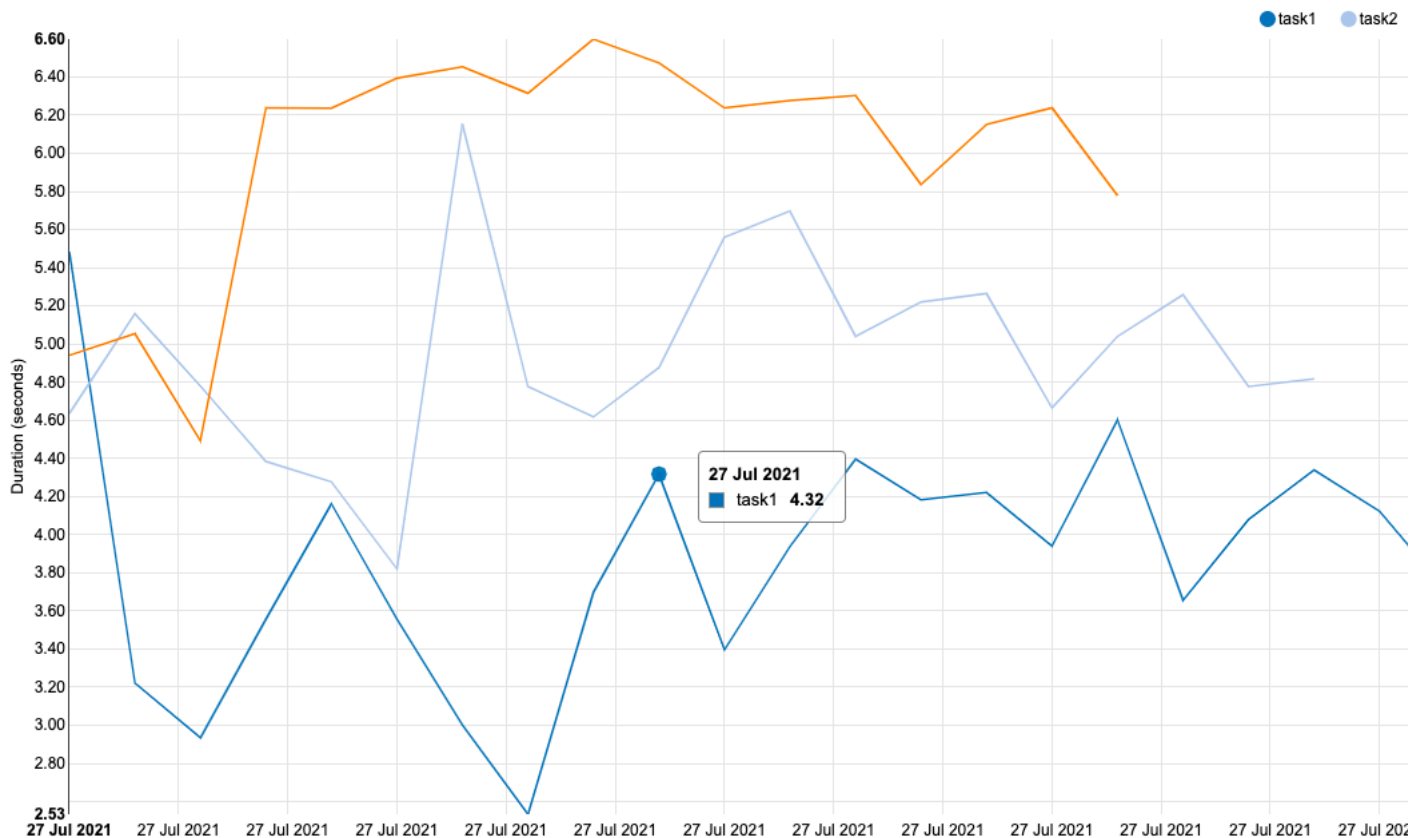
Exercise 8 - Calender view

The calender view gives you an overview of all the dates when this DAG was run along with its status as a color code.



Exercise 9 - Task Duration view

The Task Duration view gives you an overview of how much time each task took to execute, over a period of time.



Exercise 10 - Details view

The Details view give you all the details of the DAG as specified in the code of the DAG.

DAG Details

queued	3	running	3	None	7	success	215
Schedule Interval	0:01:00						
Start Date	None						
End Date	None						
Max Active Runs	8 / 16						
Concurrency	16						
Default Args	<pre>{'email': ['ramesh@somemail.com'], 'email_on_failure': False, 'email_on_retry': False, 'owner': 'Ramesh Sannareddy', 'retries': 1, 'retry_delay': datetime.timedelta(0, 300), 'start_date': DateTime(2021, 7, 27, 0, 0, 0, tzinfo=Timezone('UTC'))}</pre>						
Tasks Count	3						
Task IDs	['task1', 'task2', 'task3']						
Filepath	dummy_dag.py						
Owner	Ramesh Sannareddy						
DAG Run Timeout	None						
Tags	None						

Exercise 11 - Code view

The Code view lets you view the code of the DAG.

```

1  # import the libraries
2
3  from datetime import timedelta
4  # The DAG object; we'll need this to instantiate a DAG
5  from airflow import DAG
6  # Operators; we need this to write tasks!
7  from airflow.operators.bash_operator import BashOperator
8  # This makes scheduling easy
9  from airflow.utils.dates import days_ago
10
11 #defining DAG arguments
12
13 # You can override them on a per-task basis during operator initialization
14 default_args = {
15     'owner': 'Ramesh Sannareddy',
16     'start_date': days_ago(0),
17     'email': ['ramesh@somemail.com'],
18     'email_on_failure': False,
19     'email_on_retry': False,
20     'retries': 1,
21     'retry_delay': timedelta(minutes=5),
22 }
    
```

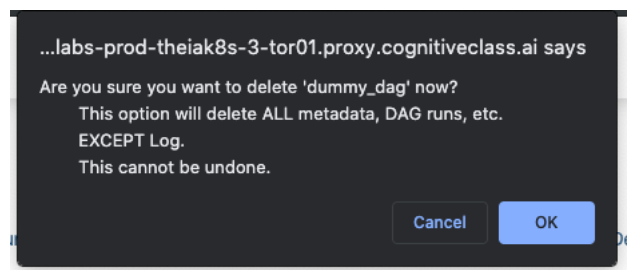
Exercise 12 - Delete a DAG

To delete a DAG click on the delete button.

```

1  # import the libraries
    
```

You will get a confirmation pop up as shown in the image below. Click OK to delete the DAG.



Practice exercises

1. Problem:

Unpause any existing DAG and monitor it.

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-01-10	0.2	Shreya Khurana	Updated screenshots
2021-07-05	0.1	Ramesh Sannareddy	Created initial version of the lab