



## Apache Airflow Integration

### 1. Airflow deployment

As usual, we login to the OpenShift cluster

```
# Replace the command with your own one inside the single quotes and run the cell  
# Example OC_LOGIN_COMMAND='oc login --token=sha256~3bR5KXgwiUoaQiph2_kIXCDQnVfm_HQy3YwU2m-  
OC_LOGIN_COMMAND='_replace_this_string_by_pasting_the_clipboard_'  
$OC_LOGIN_COMMAND
```

We begin by allocating a small piece of storage for our DAGs. We simply call it `my-volume-claim`

```
# This command creates a small persistent volume claim (1 GB, NFS)
```

```
oc apply -f - << EOF  
kind: PersistentVolumeClaim  
apiVersion: v1  
metadata:  
  name: my-volume-claim  
  namespace: airflow  
spec:  
  accessModes:  
    - ReadWriteMany  
  resources:  
    requests:  
      storage: 1Gi  
    storageClassName: managed-nfs-storage  
    volumeMode: Filesystem  
status:  
  accessModes:  
    - ReadWriteMany  
  capacity:
```

```
storage: 1Gi
EOF
```

Now, we reconfigure Airflow to look in our storage to find the DAGs. Additionally, we change one parameter (`lazy_load`) that is mandatory for the monitoring to work properly

```
helm upgrade --install airflow apache-airflow/airflow \
--set core.lazy_load_plugins=False \
--set dags.persistence.enabled=true \
--set dags.persistence.existingClaim=my-volume-claim \
--set dags.gitSync.enabled=false
```

## 2. Airflow customization for Databand

The monitors of databand need a python package that we need to install in two pods of Airflow.

**Warning:** in a production system, you should extend the official container with the package and not install it directly into the pod. For educational purposes, it is OK to modify directly the pod but be aware that these changes will be lost after a redeployment / restart / etc.

*# Install the monitoring package. Expect a long output*

```
oc rsh --shell=/bin/bash airflow-worker-0 /home/airflow/.local/bin/pip install dbnd-airflow
POD_SCHEDULER=$(oc get pods | grep airflow-scheduler | awk '{print $1}')
oc rsh --shell=/bin/bash $POD_SCHEDULER /home/airflow/.local/bin/pip install dbnd-airflow
echo dbnd-airflow-auto-tracking installed in airflow-worker-0 and $POD_SCHEDULER
```

That is the simple DAG that databand needs to initiate the monitors. We copy it into the default directory for the dags. Again, this file will be lost every time you redeploy / restart / etc. the pod. Never do this in production

```
oc project airflow
echo '# This DAG is used by Databand to monitor your Airflow installation.
from airflow_monitor.monitor_as_dag import get_monitor_dag
dag = get_monitor_dag()
' > databand_airflow_monitor.py
```


```
oc cp databand_airflow_monitor.py airflow-worker-0:/opt/airflow/dags
```

After some minutes, you should see a DAG in the Airflow console. Please activate it as indicated in the picture:

[illegible]

```
curl https://raw.githubusercontent.com/apache/airflow/main/airflow/example_dags/example_com
curl https://raw.githubusercontent.com/apache/airflow/main/airflow/example_dags/tutorial.py
```

```
oc cp my_test_dag.py airflow-worker-0:/opt/airflow/dags
oc cp tutorial.py airflow-worker-0:/opt/airflow/dags
```



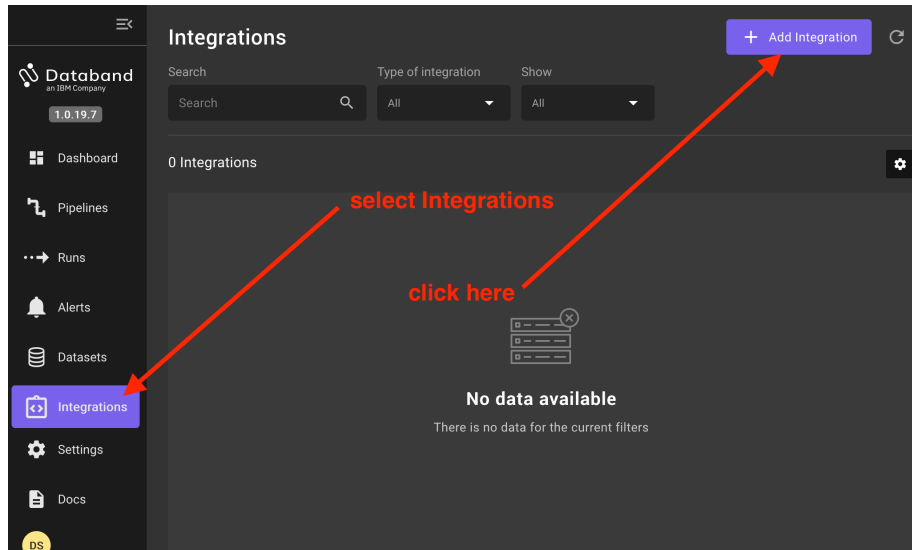
The screenshot displays the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Datasets, Security, Browse, Admin, and Docs. The current time is 12:26 UTC. Below the navigation bar, the main heading is "DAGs". A filter section allows users to select between All, Active, or Paused DAGs, with a search input field labeled "Search DAGs". An "Auto-refresh" toggle is also present. The central area contains a table listing DAGs:

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
<a href="#">databand_airflow_monitor</a>	databand	(1) [Success]	[Cron: * * * * *]	2023-03-01, 10:25:00	2023-03-01, 10:25:00	[Task: project.airflow-monitor] (1)	[Refresh] [Delete]
<a href="#">example_complex</a>	airflow	(1) [Success]	None	2023-03-01, 12:03:30		[Task: example.example2] (2)	[Refresh] [Delete]
<a href="#">tutorial</a>	airflow	(1) [Success]	1 day, 0:00:00	2023-02-28, 12:25:07	2023-02-28, 12:25:07	[Task: tutorial.example] (3)	[Refresh] [Delete]

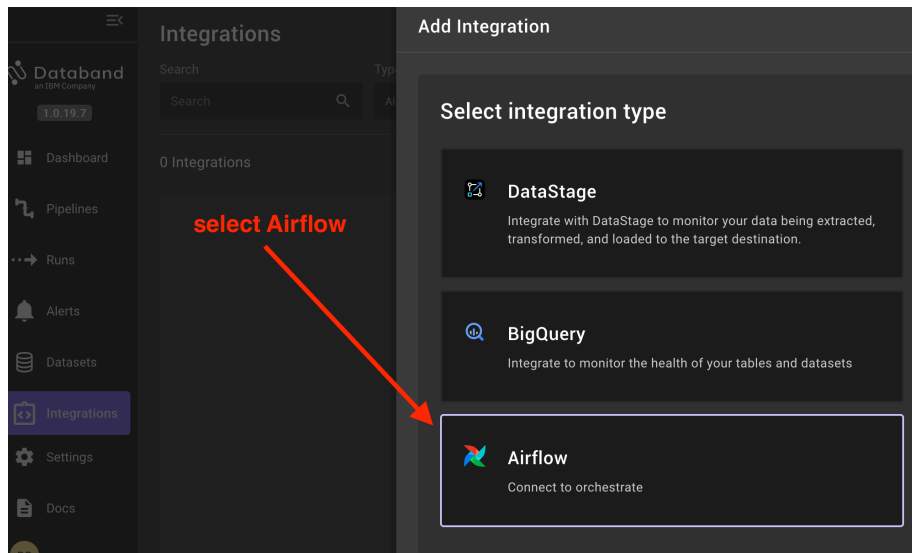
At the bottom right, it indicates "Showing 1-3 of 3 DAGs".

### 3. Integration with databand

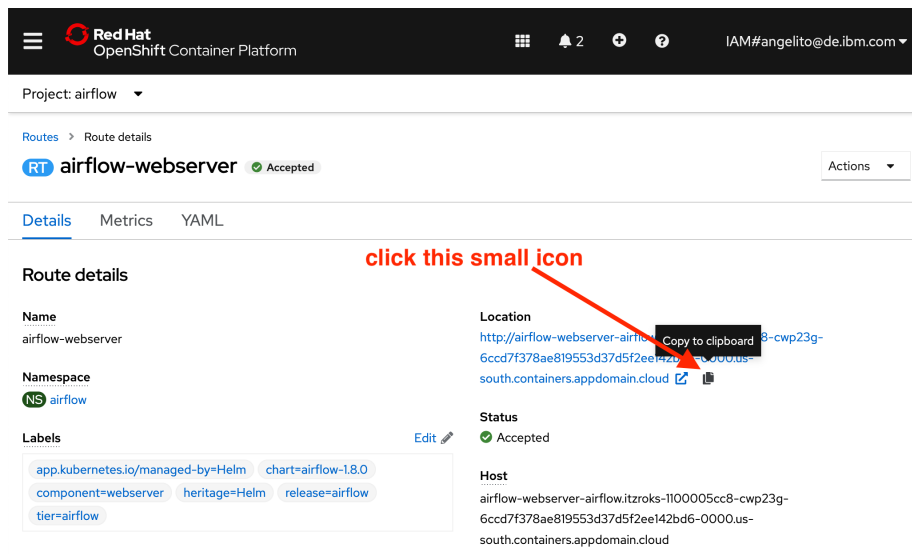
Now, we will connect Databand to Airflow. Start the Databand console and go to the Integrations section



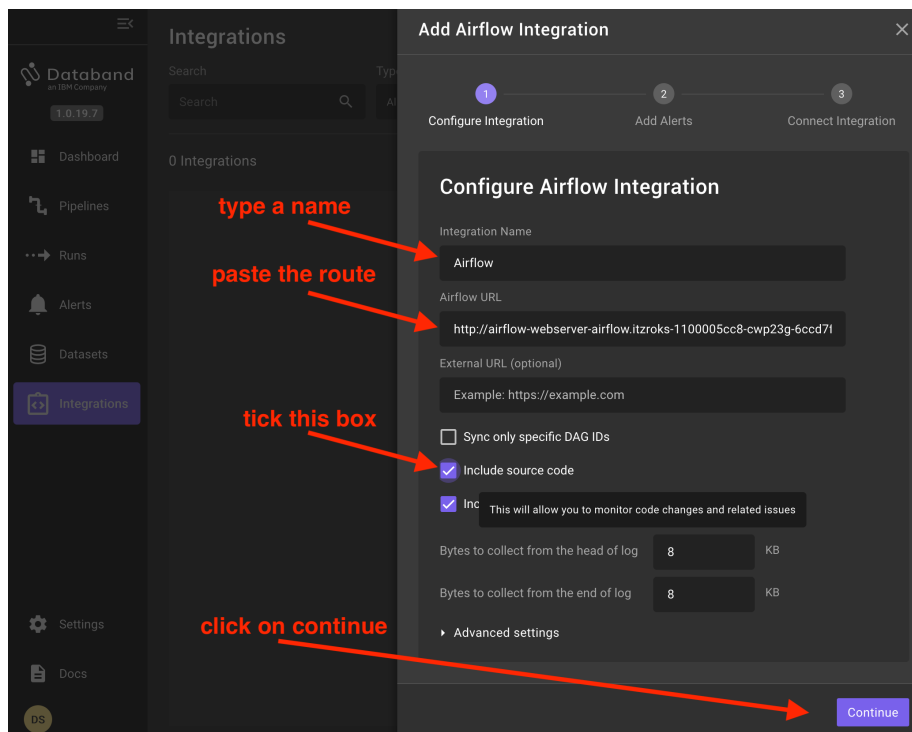
Select Airflow



Open the OpenShift console in a separate window and pick the address of the Airflow route



Paste the route of Airflow route in the Airflow URL field.



Complete the next section as follows:

## Add Airflow Integration

✓

2


3


Configure IntegrationAdd AlertsConnect Integration


### Add Alerts to the new integration


Alerts will be created for all pipelines connected to this new integration. You can always edit or delete those alerts in the alerts screen.

**ensure the activation**

**State Alerts**  
Alerts will be triggered when pipelines fail.

**Schema Change Alerts**  
Alerts will be triggered when run schemas will change.

 You must be logging dataset operations for the pipelines in the new integration. [Learn more about tracking dataset operations](#)

 [Learn more about adding alerts](#)

**click on continue**

Back

Continue

Now, you you will have to copy-and-paste two fields to create a connection in Airflow

6

Add Airflow Integration

✓

Configure Integration

✓

Add Alerts

3

Connect Integration

Connect Airflow Integration

1. Go to Airflow [Connections](#) page and create a new connection.

2. Copy the following value into the conn\_id field of the Airflow Connection.

dbnd\_config

copy these values for the Airflow connection

Copy

The Conn type is 'HTTP'

3. Copy the following json into the extra field of the Airflow Connection.

```
{  "airflow_monitor": {    "dag_ids": "",    "is_sync_enabled": true,    "syncer_name": "Airflow"  },  "core": {    "databand_url": "http://databand-web-databand.itzroks-1",    "databand_access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9"  },  "log": {    "preview_head_bytes": 8192,    "preview_tail_bytes": 8192  },  "tracking": {    "track_source_code": true  }}
```

Copy

☒ I've applied this configuration in Airflow

Back

Test Connection

7

This is the Airflow configuration page and the boxes to paste the values picked in the last picture

The screenshot shows the 'Edit Connection' form in the Airflow web interface. The form includes the following fields:

- Connection Id \***: A text input field containing 'dbnd\_config'.
- Connection Type \***: A dropdown menu showing 'HTTP'. Below it, a message reads: 'Connection Type missing? Make sure you've installed the corresponding Airflow Provider'.
- Description**: A large text area.
- Host**: A text input field.
- Schema**: A text input field.
- Login**: A text input field.
- Password**: A text input field.
- Port**: A text input field.
- Extra**: A text area containing a JSON snippet: 

```
{  
  "airflow_monitor": {  
    "dag_ids": "",  
    "is_sync_enabled": false,  
  }  
}
```

Red annotations are present on the form:

- A red arrow points from the text 'paste here values copied in the Databand configuration' to the 'Connection Id' field.
- Another red arrow points from the same text to the 'Extra' field.
- A red arrow points from the text 'click on save' to the 'Save' button at the bottom left.

This message indicates that the configuration has been successfully applied



Add Airflow Integration

✓


Configure Integration

✓

Add Alerts

3

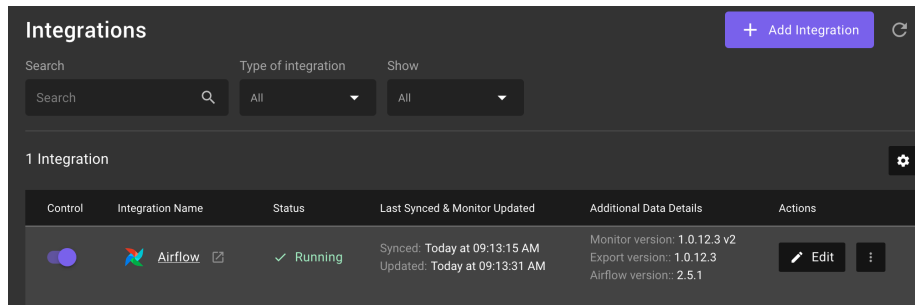
Connect Integration



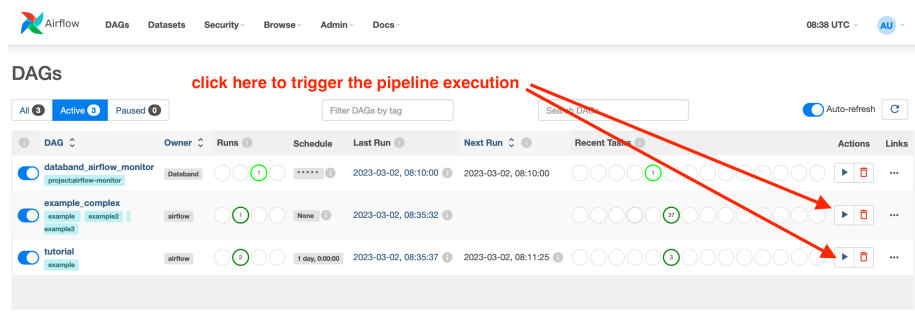
Integration has successfully connected!

Airflow integration is now synced.

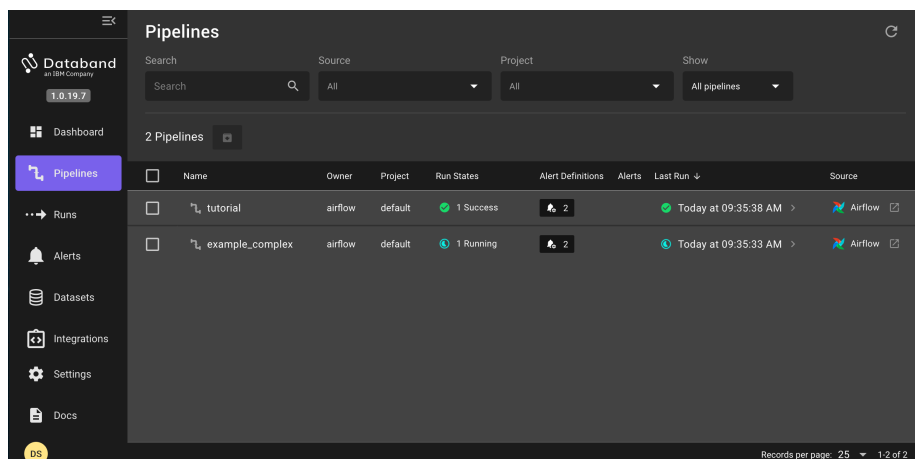
Finish



If you used the DAGs examples mentioned before, you need to trigger them manually



Finally, the two DAGs will be displayed in Databand



Next Section: Datastage integration. Previous Section: Airflow deployment  
Return to main