



## 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 project airflow
```

```
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

```
oc project airflow
```

```

helm upgrade --install airflow apache-airflow/airflow \
  --set config.core.lazy_load_plugins=False \
  --set dags.persistence.enabled=true \
  --set dags.persistence.existingClaim=my-volume-claim \
  --set dags.gitSync.enabled=false -f - << EOF
env:
  - name: AIRFLOW__CORE__LAZY_LOAD_PLUGINS
    value: 'False'
  - name: _PIP_ADDITIONAL_REQUIREMENTS
    value: 'dbnd-airflow-auto-tracking'
EOF

```

## 2. Airflow customization for Databand

There are several python libraries that activate specialized monitoring features. Although the previous command installed everything we need, you can optionally install the following additional packages, as you may want to do on a real system:

**Warning:** in a production system, you should extend the official container with the additional packages and not install them 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 project airflow
oc rsh --shell=/bin/bash airflow-worker-0 /home/airflow/.local/bin/pip install databand 'data
POD_SCHEDULER=$(oc get pods | grep airflow-scheduler | awk '{print $1}')
oc rsh --shell=/bin/bash $POD_SCHEDULER /home/airflow/.local/bin/pip install databand 'data
echo dbnd-airflow-auto-tracking installed in airflow-worker-0 and $POD_SCHEDULER

```

The following cell would add a simply DAG that databand needs to initiate the monitors. We copy it into the default directory for the dags:

```

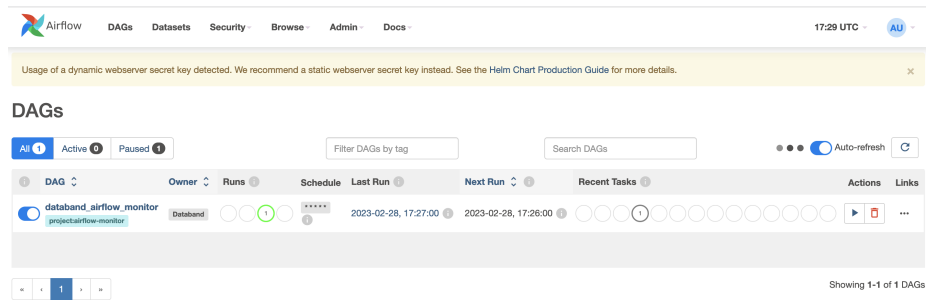
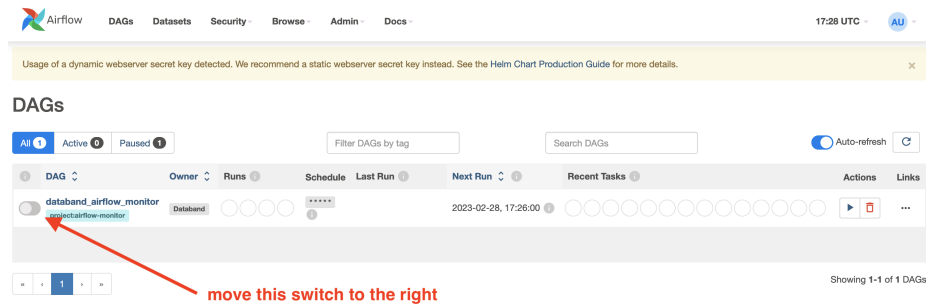
oc project airflow
echo '# This DAG is used by Databand to monitor your Airflow installation.
from airflow_monitor.monitor_as_dag import get_monitor_dag
dag = get_monitor_dag()

```

```
' > databand_airflow_monitor.py
```

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

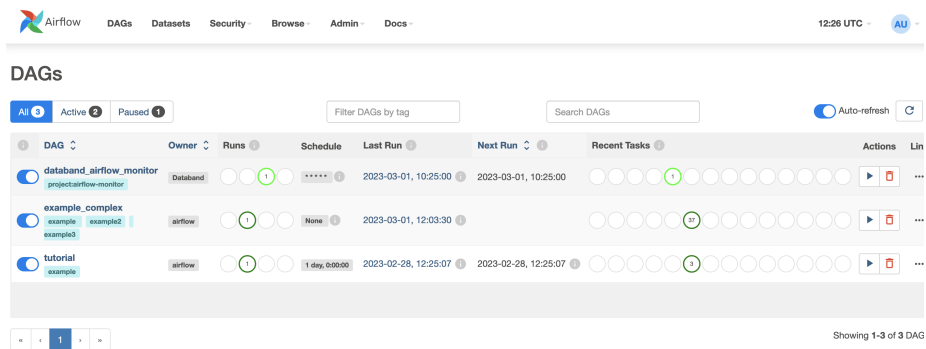
After some minutes, you should see a DAG in the Airflow console. Please activate it as indicated in the picture:



Actually, this is an auxiliary DAG of databand. Leave it as-is and you may want to experiment with your own ones or simply try a few examples located here [https://github.com/apache/airflow/tree/main/airflow/example\\_dags](https://github.com/apache/airflow/tree/main/airflow/example_dags)

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

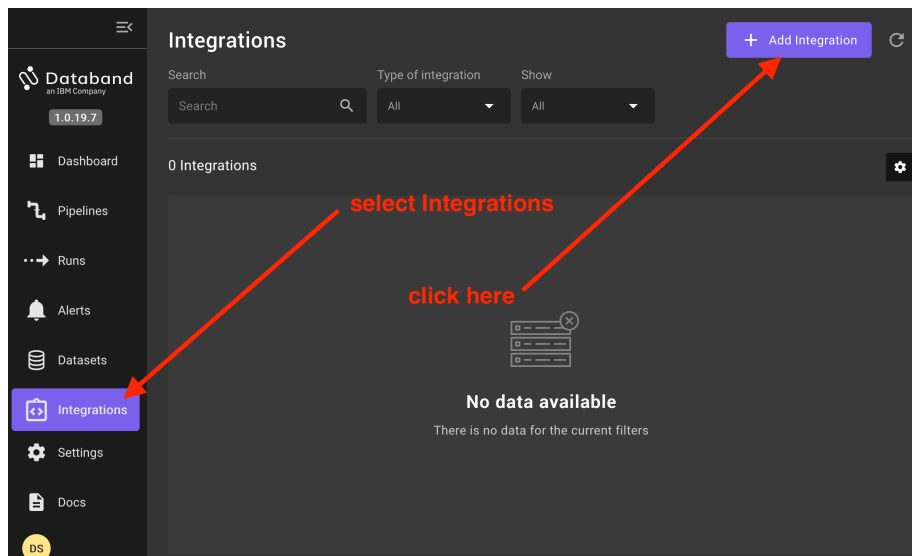


The screenshot shows the Airflow web interface. At the top, there's a navigation bar with links to DAGs, Datasets, Security, Browse, Admin, and Docs. The main header shows '12:26 UTC' and a user profile 'AU'. Below the header, the 'DAGs' section is active, displaying a table of DAGs. The table has columns for DAG, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, and Actions. Three DAGs are listed: 'databand\_airflow\_monitor' (owner: databand), 'example\_complex' (owner: example), and 'tutorial' (owner: example). The 'databand\_airflow\_monitor' DAG is highlighted with a blue background. Below the table, there's a pagination bar showing 'Showing 1-3 of 3 DAGs'.

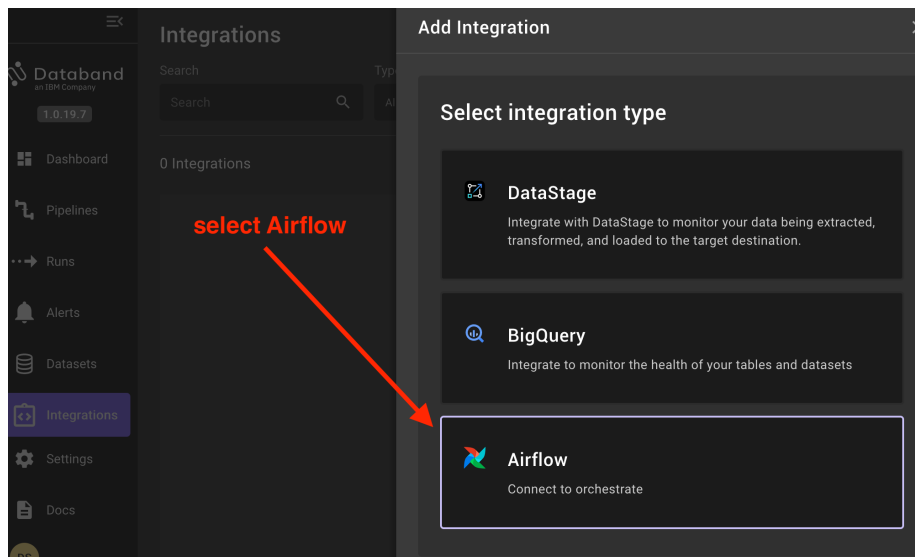
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
databand_airflow_monitor projectairflow-monitor	databand	1	*****	2023-03-01, 10:25:00	2023-03-01, 10:25:00	1	▶ 🗑
example_complex example example2 example3	airflow	1	None	2023-03-01, 12:03:30		1	▶ 🗑
tutorial example	airflow	1	1 day, 00:00:00	2023-02-28, 12:25:07	2023-02-28, 12:25:07	1	▶ 🗑

### 3. Integration with databand

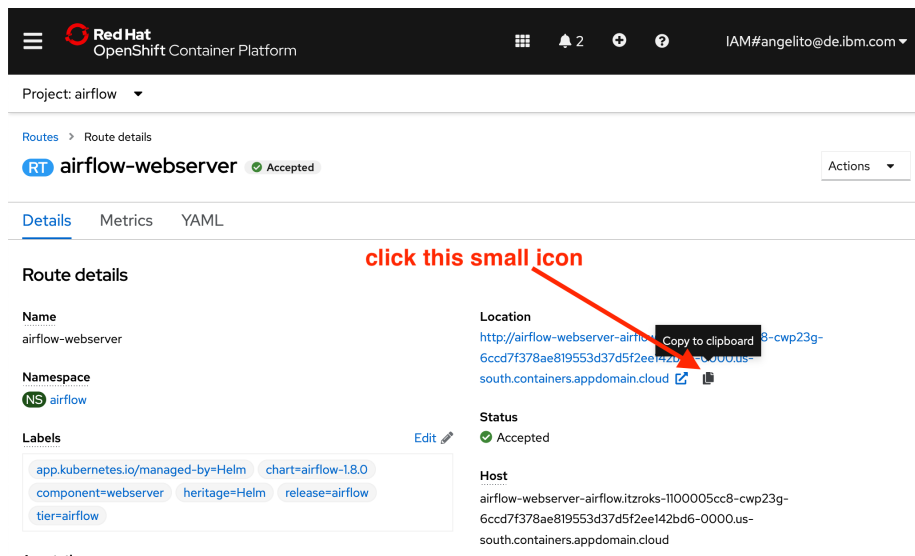
Now, we will connect Databand to Airflow. Start the Databand console and go to the Integrations section



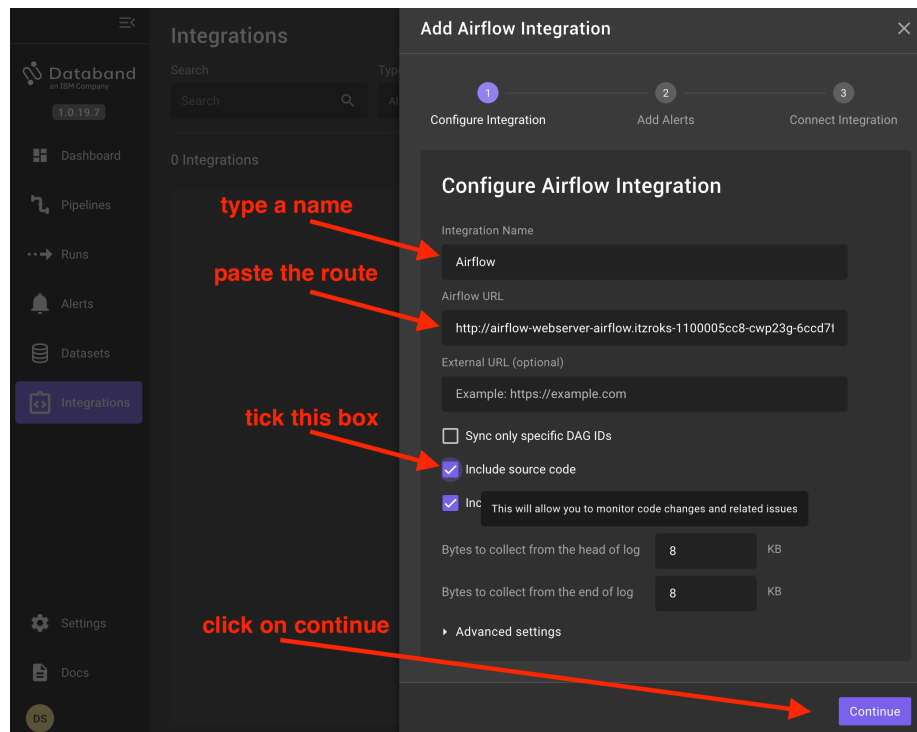
Select Airflow



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

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 third 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

✓


✓

3

Configure Integration

Add Alerts

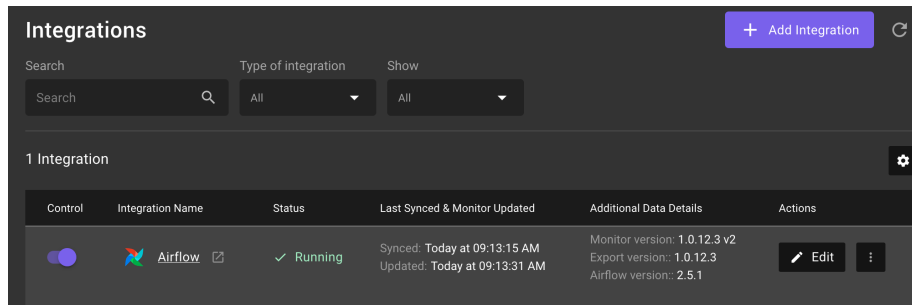
Connect Integration



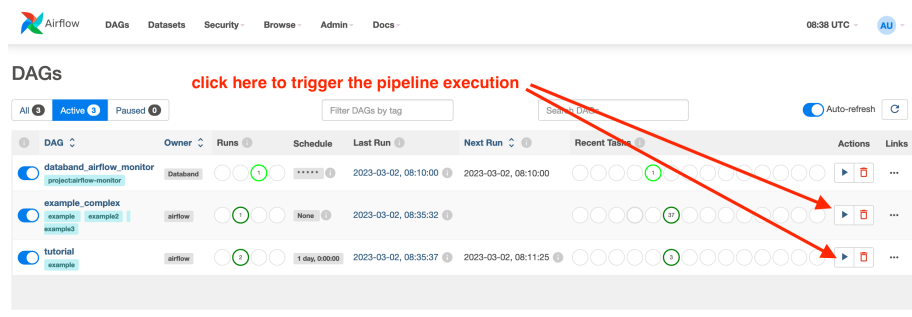
Integration has successfully connected!

Airflow integration is now synced.

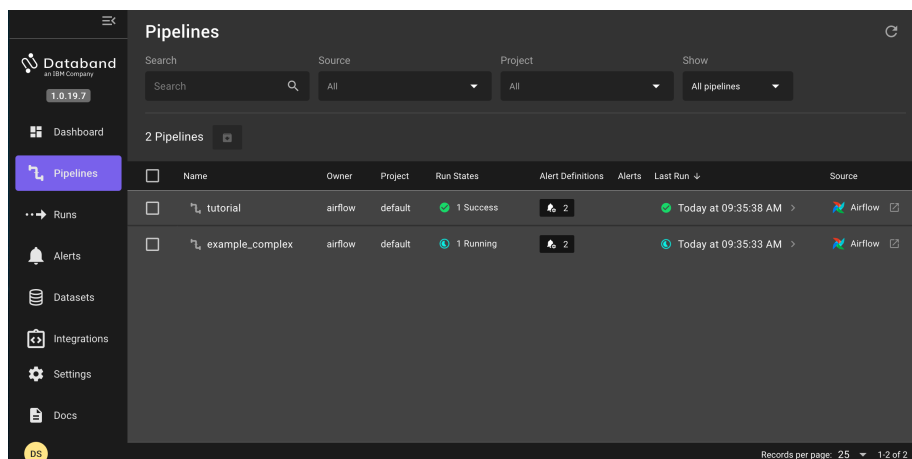
Finish



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