

# 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-UOrs --server=https://c109-e.us-east.containers.cloud.ibm.com:31656'  
OC\_LOGIN\_COMMAND='\_replace\_this\_string\_by\_pasting\_the\_clipboard\_'  
$OC\_LOGIN\_COMMAND

We beging 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 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 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 databand 'databand[postgres,airflow]' dbnd-airflow-auto-tracking dbnd-airflow-monitor dbnd-airflow-export dbnd-airflow-versioned-dag  
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 'databand[postgres,airflow]' dbnd-airflow-auto-tracking dbnd-airflow-monitor dbnd-airflow-export dbnd-airflow-versioned-dag  
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/airflaow/tree/main/airflow/example_dags>

curl https://raw.githubusercontent.com/apache/airflow/main/airflow/example\_dags/example\_complex.py > my\_test\_dag.py  
curl https://raw.githubusercontent.com/apache/airflow/main/airflow/example\_dags/tutorial.py > tutorial.py  
  
oc cp my\_test\_dag.py airflow-worker-0:/opt/airflow/dags  
oc cp tutorial.py airflow-worker-0:/opt/airflow/dags



## 3. Integration with databand

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



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:



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



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



This message indicates that the configuration has been successfuly applied





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](./5_datastage_int.ipynb). Previous Section: [Airflow deployment](./3_airflow_deploy.ipynb)

[Return to main](../README.md)