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 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 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 'da
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
```

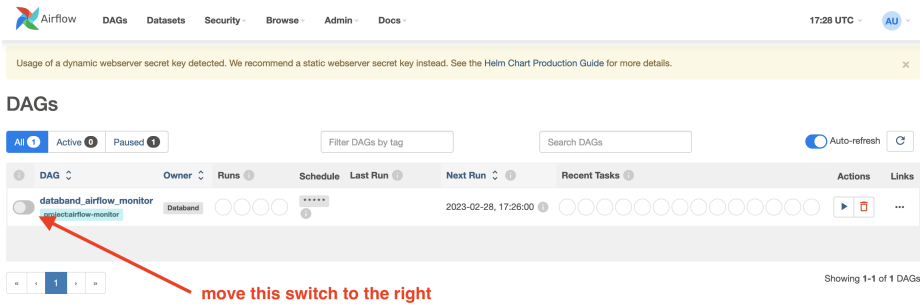
The following cell would add a simply DAG that databand needs to initiate the monitors. We copy it into the default directory for the dags:

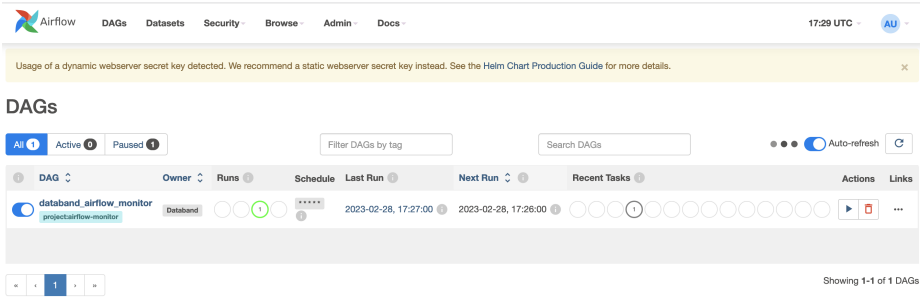
```
oc project airflow
echo '# This DAG is used by Databand to monitor your Airflow installation.
from airflow_monitor.monitor_as_dag import get_monitor_dag
dag = get_monitor_dag()
```

```
' > 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:

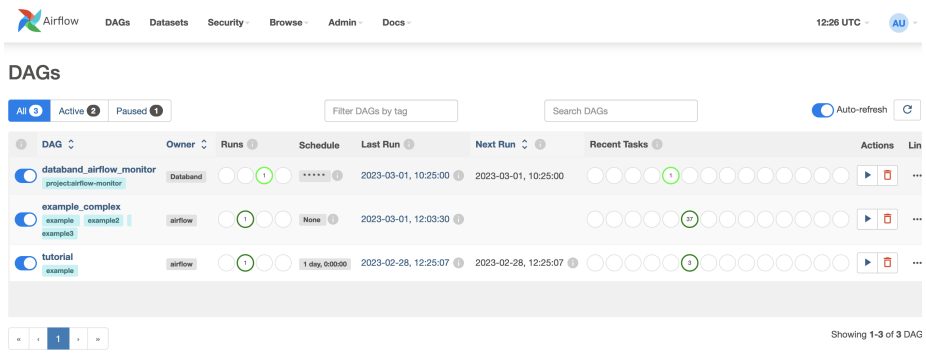




Actually, this is an auxiliary DAG of databand. Leave it as-is and you may want to experiment with your own ones or simply try a few examples located here https://github.com/apache/airflaow/tree/main/airflow/example_dags

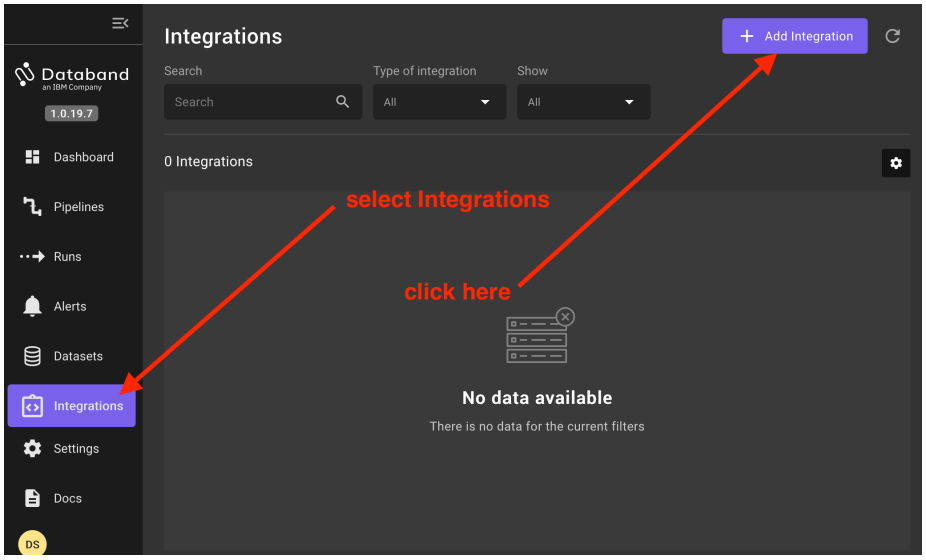
```
curl https://raw.githubusercontent.com/apache/airflow/main/airflow/example_dags/example_comp
curl https://raw.githubusercontent.com/apache/airflow/main/airflow/example_dags/tutorial.py

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
```

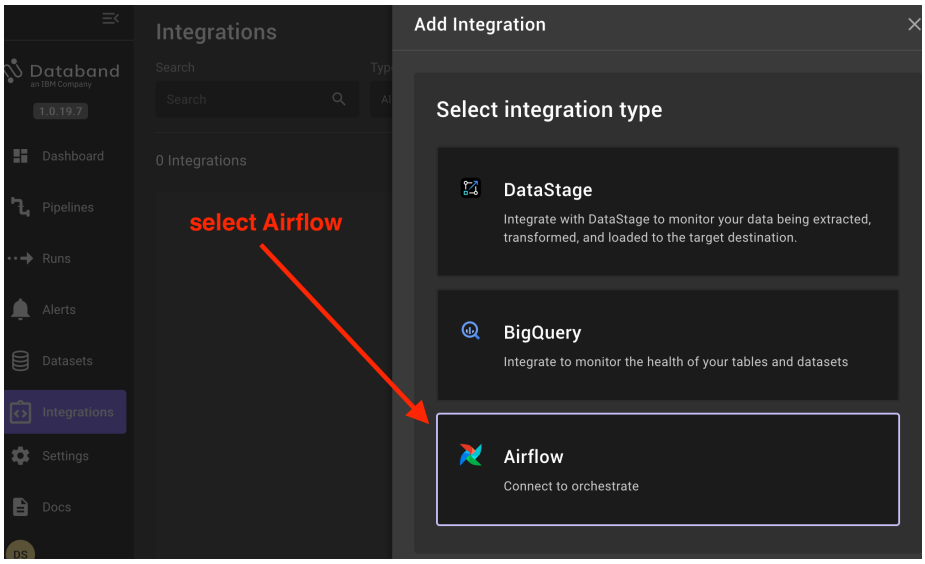


3. Integration with databand

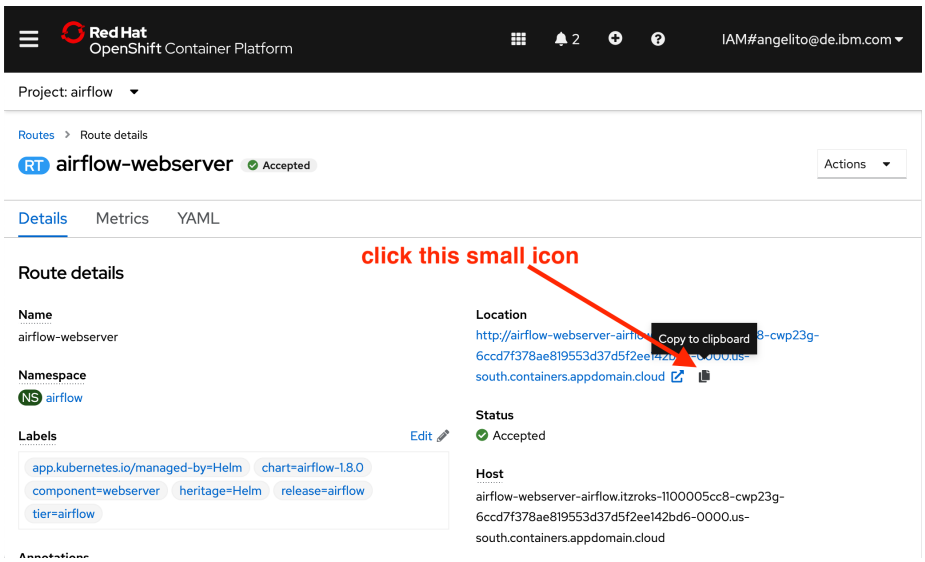
Now, we will connect Databand to Airflow. Start the Databand console and go to the Integrations secion



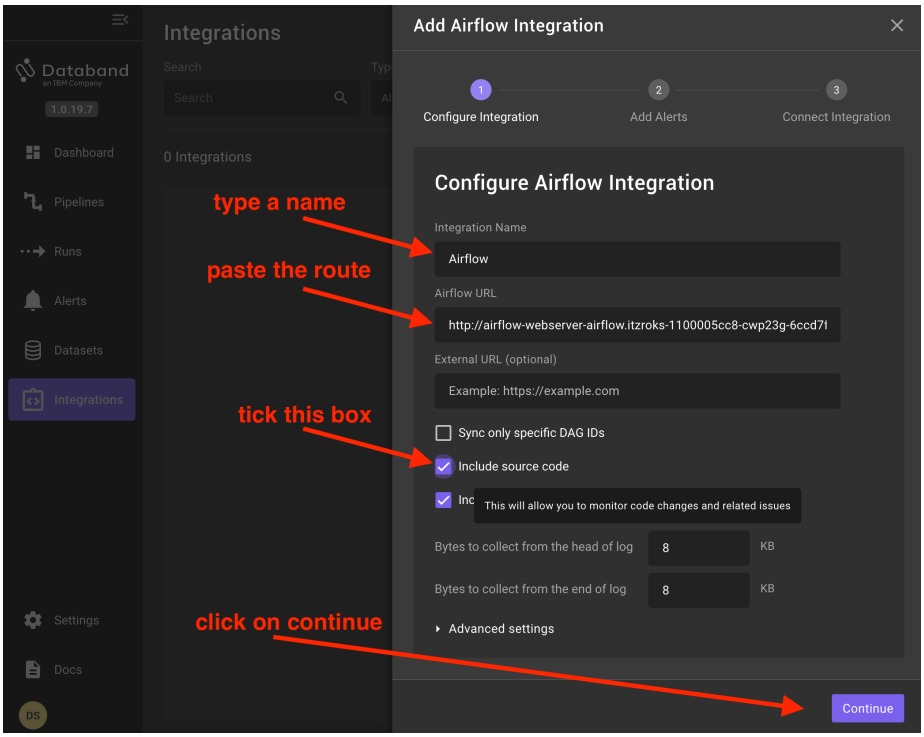
Select Airflow



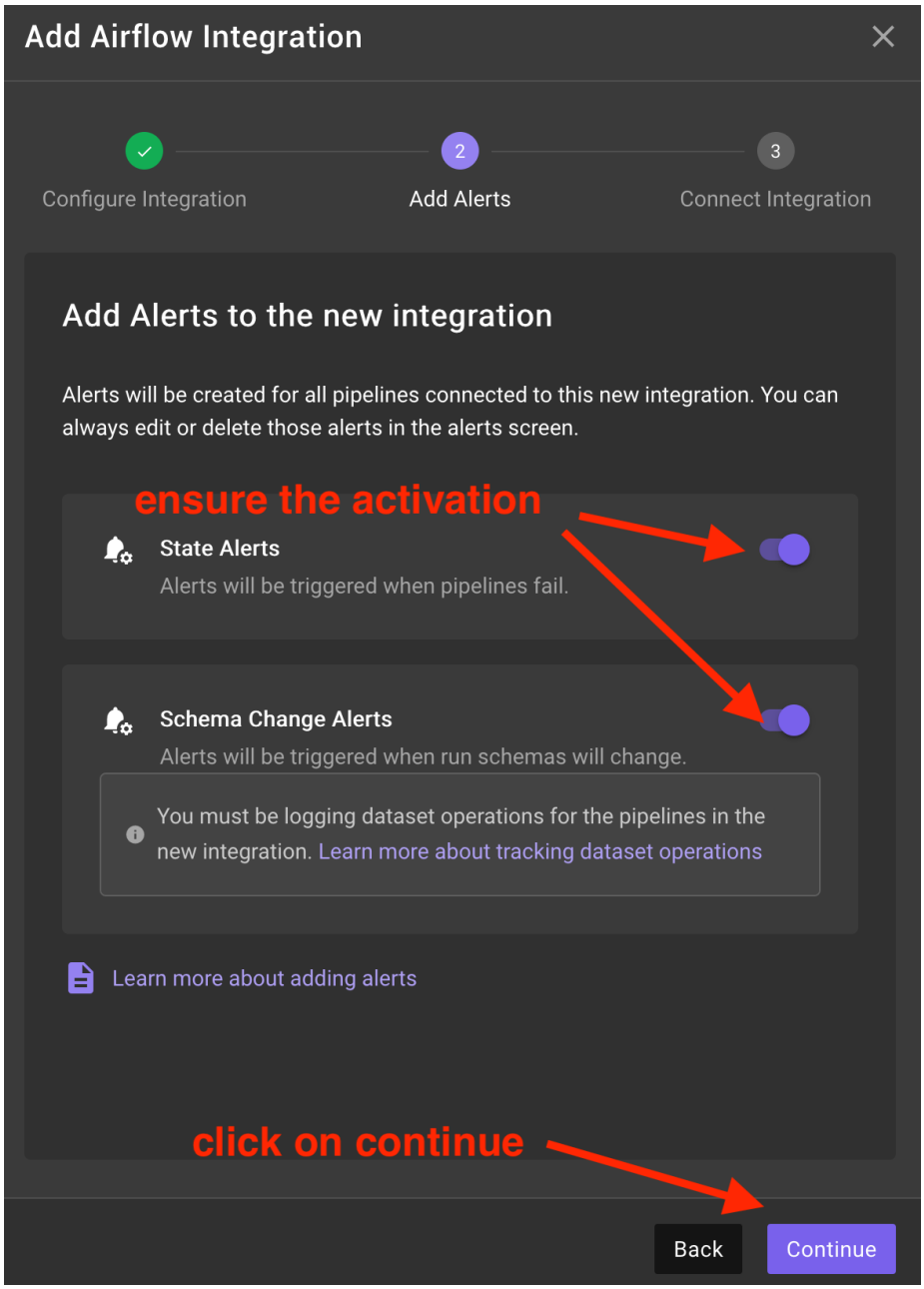
Open the OpenShift console in a separate window and pick the address of the Airflow route



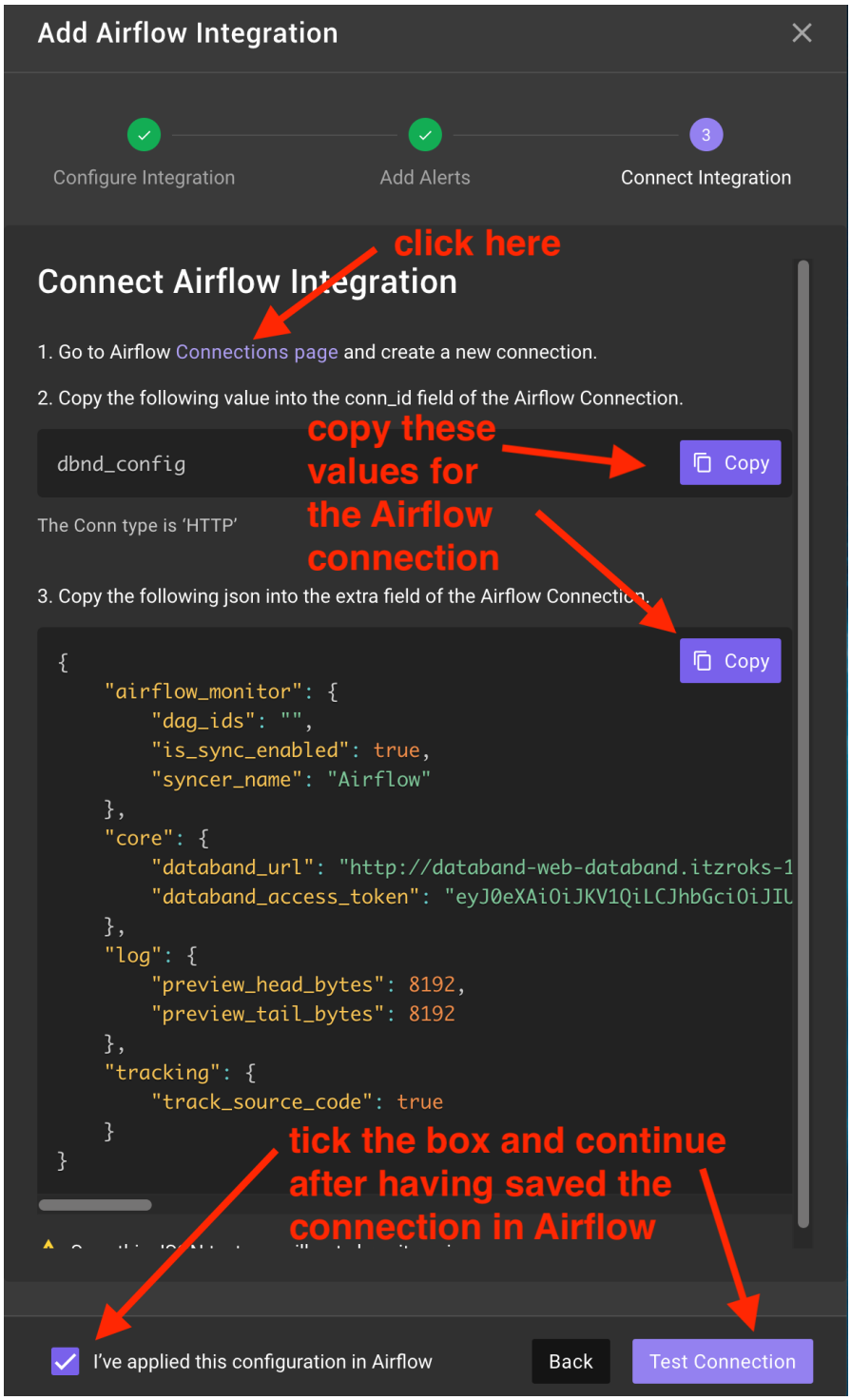
Paste the route of Airflow route in the `Airflow URL` field.



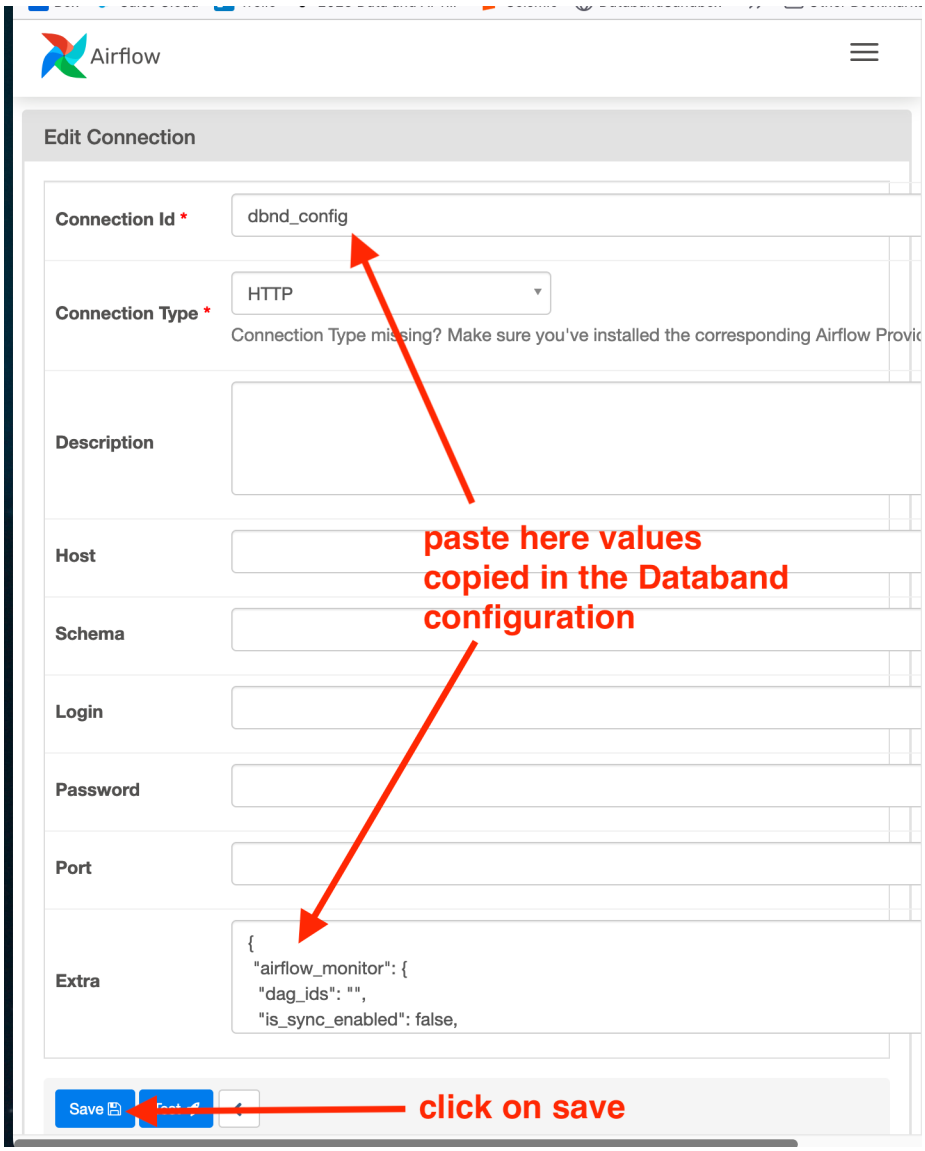
Complete the next section as follows:



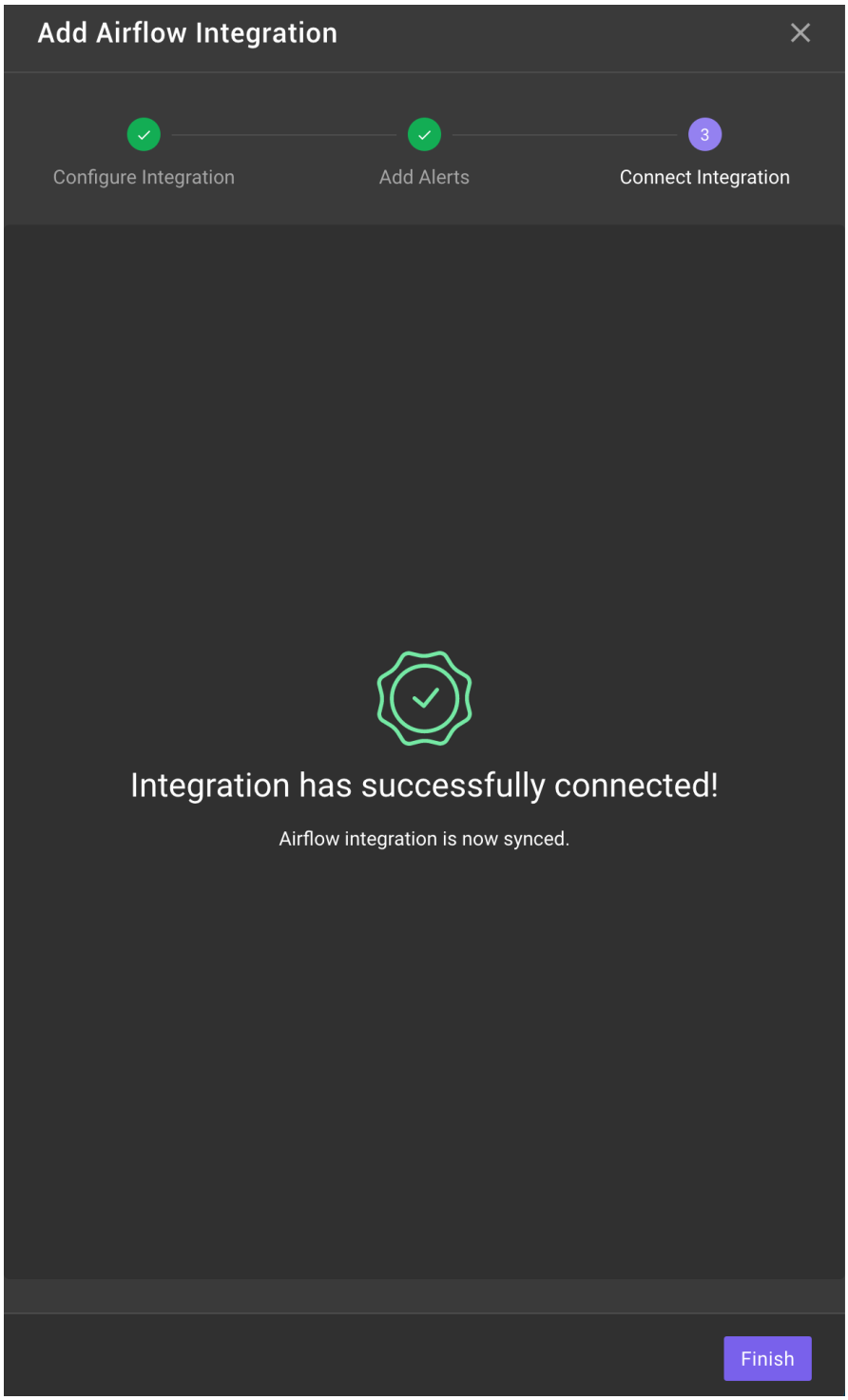
Now, you you will have to copy-and-paste two fields to create a connection in Airflow

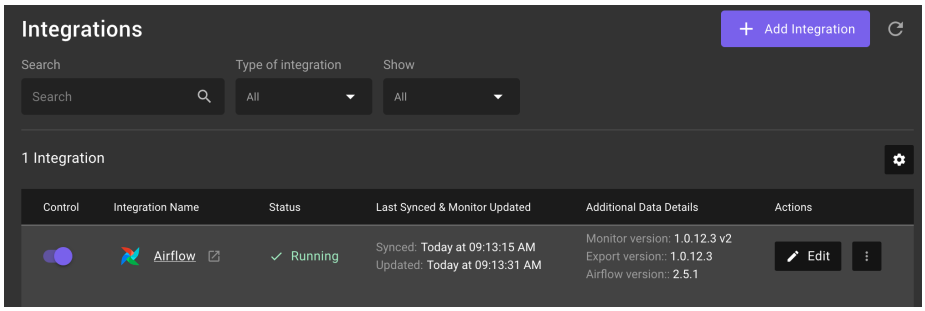

Add Airflow Integration
Configure Integration
Add Alerts
3
Connect Integration
click here
Connect Airflow Integration
1. Go to Airflow Connections page and create a new connection.
2. Copy the following value into the conn_id field of the Airflow Connection.
dbnd_config
Copy
copy these values for the Airflow connection
The Conn type is 'HTTP'
3. Copy the following json into the extra field of the Airflow Connection.
Copy
{
"airflow_monitor": {
"dag_ids": "",
"is_sync_enabled": true,
"syncer_name": "Airflow"
},
"core": {
"databand_url": "http://databand-web-databand.itzroks-1
"databand_access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIL
},
"log": {
"preview_head_bytes": 8192,
"preview_tail_bytes": 8192
},
"tracking": {
"track_source_code": true
}
}
tick the box and continue after having saved the connection in Airflow
I've applied this configuration in Airflow
Back
Test Connection

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

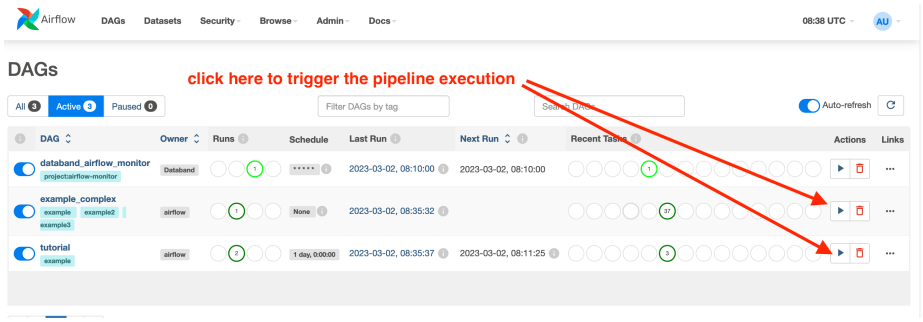


This message indicates that the configuration has been successfuly applied

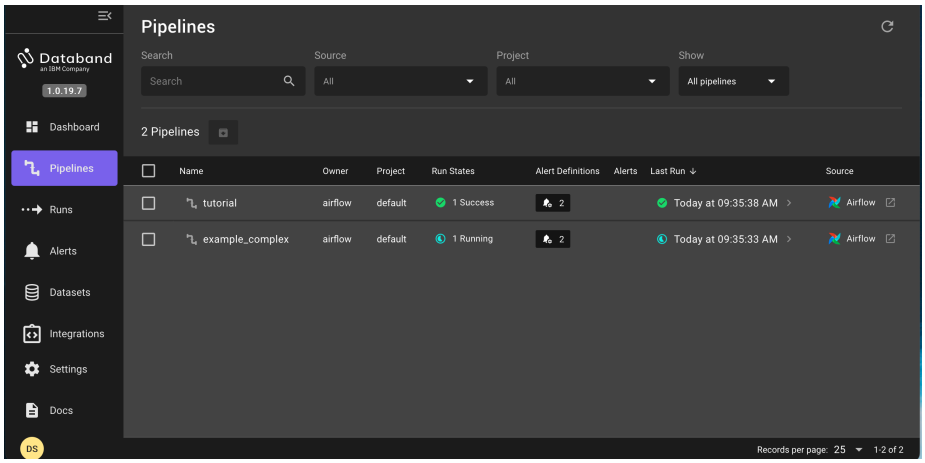

Add Airflow Integration
Configure Integration
Add Alerts
3
Connect Integration
Integration has successfully connected!
Airflow integration is now synced.
Finish



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