Preparation for DAGs development

1. Connect Airflow and Postgres

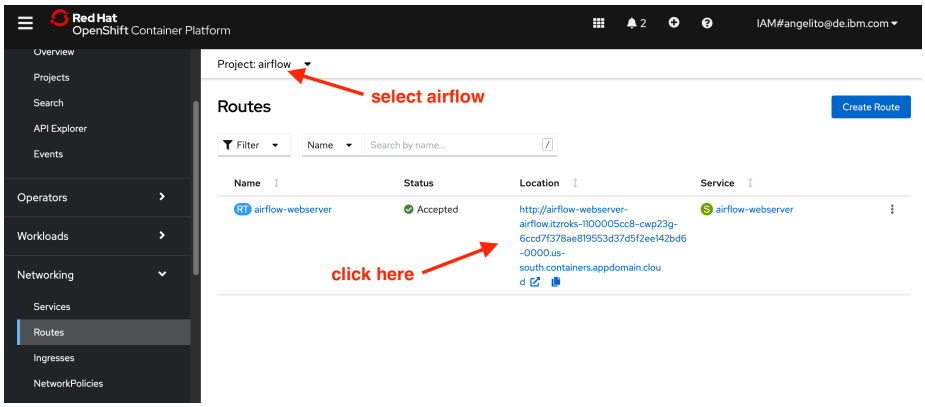
Before developing DAGs with Airflow and Postgres, we need to add a connection for Airflow to find the databases. As usual, we login to the cluster:

```
# 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-U
OC_LOGIN_COMMAND='_replace_this_string_by_pasting_the_clipboard_'
$OC_LOGIN_COMMAND
```

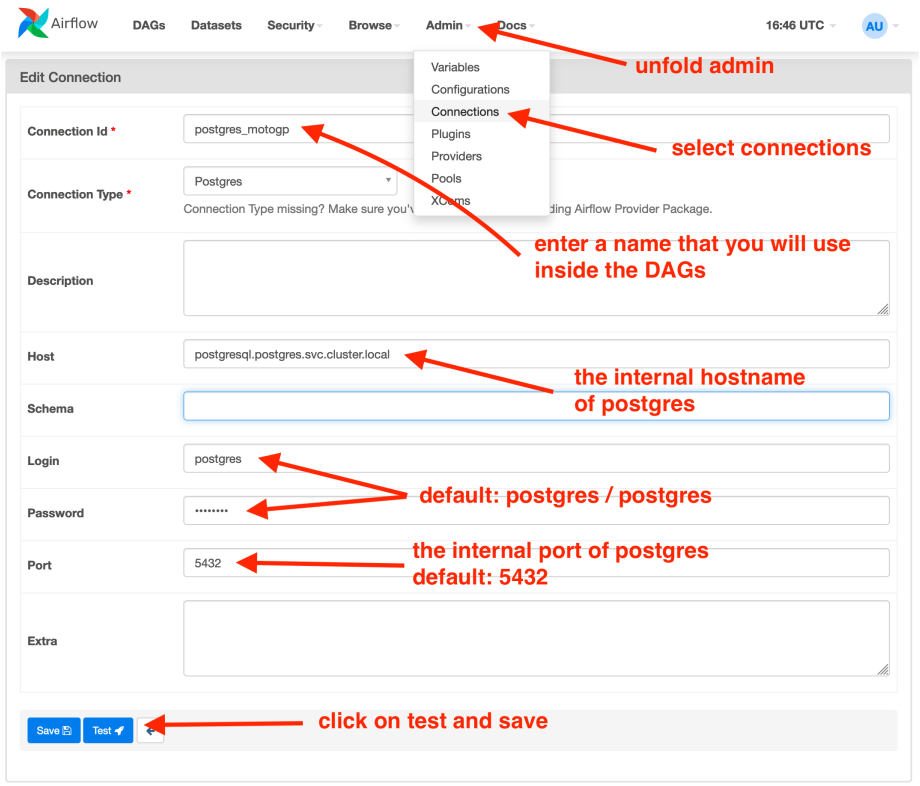
Then, we need to retrieve two values (the hostname and the port) that we will use immediately. Prepare for copy-and-paste them below in the Airflow new connection menu item.

```
oc project postgres
internalservice=$(oc get svc | grep ClusterIP | awk '{print $1}')
internalhostname=$(oc get svc $internalservice -o go-template --template='{{.metadata.name}}
internalport=$(oc get svc | grep ClusterIP | awk '{print $5}' | cut -f1 -d'/')
echo Internal hostname of Postgres: $internalhostname
echo Internal port of Postgres: $internalport
```

In order to create the connection, we need to access the Airflow admin interface as we did during the **Airflow Deployment** section:



Copy-and-paste the values we obtained before in the new connection menu:



2. Install the databand monitoring packages

Airflow will report the pipeline information to Databand and it will be done via the python packages that we will install now. Actually, we already installed Databand packages during the chapter Airflow integration. The following commands are an alternative way that uses a bundled installation syntax and states explicitly the airflow and postgres features:

```
# Install python package to report Postgres and Airflow information to Databand
oc project airflow

oc rsh  --shell=/bin/bash airflow-worker-0 /home/airflow/.local/bin/pip install 'databand[ai
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[airf

echo 'databand[airflow,postgres]'installed in airflow-worker-0 and $POD_SCHEDULER
```

Notice that you would never touch a running container like this to install python packages or additional software in a real production environment. The right way is customizing or extending the docker image as documented here

3. Transfer of DAGs to Airflow

Now, we will transfer some files from our local machine to the Airflow containers. Please ensure that you are in the local directory where the our sample DAGs are located. If you cloned this git repository, the directory is simply called `dags`, under the root level (go up if you are in the jupyter directory)

```
# you may need to modify the cd command to place yourself in the DAGs directory
pwd
cd ../dags
ls -l
```

If you did it right you will see several python file and the `sql`subdirectory. Something like this:

```
# Do NOT try to execeute this cell. It is just for information

-rw-r--r--  1 Angel  wheel   152 Mar 13 11:06 databand_airflow_monitor.py
-rw-r--r--  1 Angel  wheel  2128 Mar 17 15:57 motogp_dag.py
-rw-r--r--  1 Angel  wheel  3371 Mar 17 18:05 pythondag.py
-rw-r--r--  1 Angel  wheel  3968 Mar 17 17:48 pythondag_airflow.py
drwxr-xr-x  8 Angel  wheel   256 Mar 13 11:06 sql
-rw-r--r--  1 Angel  wheel  2110 Mar 17 17:51 sql_airflow_dag.py
```

Now, we will transfer some files:

```
oc rsh airflow-worker-0  mkdir -p /opt/airflow/dags/sql

for file in *.py
do
  oc cp $file airflow-worker-0:dags/
done

for file in sql/*
do
  oc cp $file airflow-worker-0:dags/sql
done
```

Note that this is just one of the possibilities to add customized DAGs to Airflow. Other options, some of them more elegant, are documented here

---

Next Section: SQL Airflow pipelines

Return to main