Practice: Pushing data from the KPO

Sharing data with the KubernetesPodOperator is a bit different than with other operators.

You have a couple of things to such as:

1. Turning on the do_xcom_push parameter
2. Configure a Dockerfile that creates the file return.json
3. Write the data to share in that file

Ready? Let's go! 👇

Prerequisites

You need access to an Airflow instance, Docker, and a Kubernetes cluster. You can watch the previous videos to have everything set up.

Create the DAG

In the folder dags/ create a new DAG share_pod_dag.py with the following code:

```
from airflow import DAG
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import KubernetesPodOperat
or
from airflow.decorators import task

from datetime import datetime

with DAG(
    dag_id="share_pod_dag",
    start_date=datetime(2022, 1 , 1),
    schedule="@once",
```

```
    catchup=False
):

    run_simple_app = KubernetesPodOperator(
        task_id="run_simple_app",
        namespace="<REPLACE WITH THE CORRECT NAMESPACE>",
        image="<REPLACE WITH THE CORRECT IMAGE>",
        name="airflow-test-pod",
        in_cluster=False,
        cluster_context="docker-desktop",
        config_file="/usr/local/airflow/include/.kube/config",
        is_delete_operator_pod=True,
        get_logs=True,
        log_events_on_failure=False
    )

    @task
    def pull_data(ti=None):
        print(ti.xcom_pull(task_ids='run_simple_app'))

    run_simple_app >> pull_data()
```

Exercise

Quick exercise for you.

Try to configure the KubernetesPodOperator correctly:

- Set the namespace (use `kubectl get ns`)
- Set the docker image (use the one you built previously)
- Set the parameter to push XCOMs from the KPO.

Take your time.

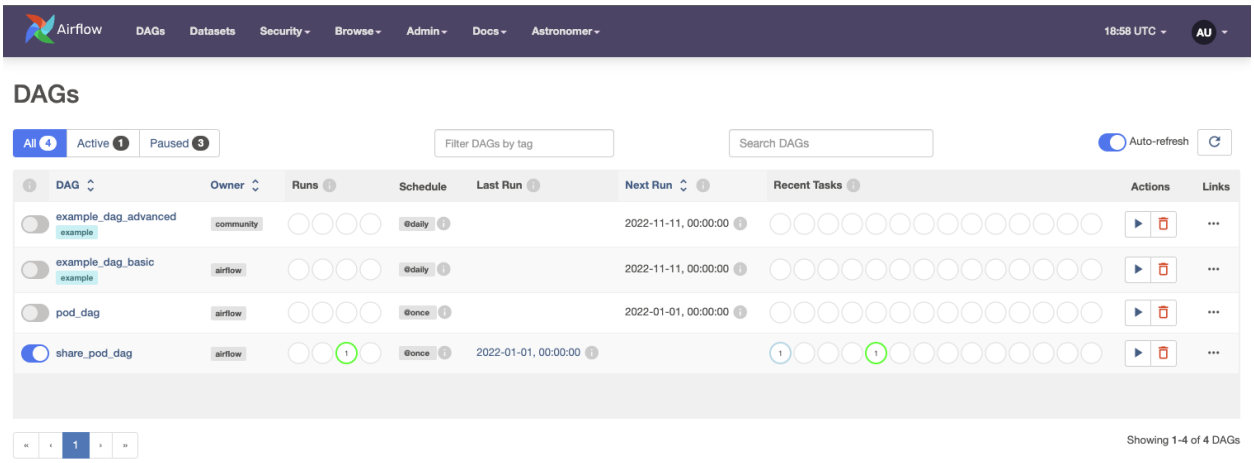
Solution

```
from airflow import DAG
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import KubernetesPodOperat
or
from airflow.decorators import task
```

```
from datetime import datetime

with DAG(
    dag_id="share_pod_dag",
    start_date=datetime(2022, 1 , 1),
    schedule="@once",
    catchup=False
):

    run_simple_app = KubernetesPodOperator(
        task_id="run_simple_app",
        namespace="default",
        image="localhost:5000/simple_app",
        name="airflow-test-pod",
        do_xcom_push=True,
        in_cluster=False,
        cluster_context="docker-desktop",
        config_file="/usr/local/airflow/include/.kube/config",
        is_delete_operator_pod=False,
        get_logs=True,
        log_events_on_failure=False
    )

    @task
    def pull_data(ti=None):
        print(ti.xcom_pull(task_ids='run_simple_app'))

    run_simple_app >> pull_data()
```

Run the DAG!

On the Airflow UI, turn on the toggle of the share_pod_dag to run it!



Wait a little bit, you should see the DAG Run in success.

Go to Browse → XCOMs

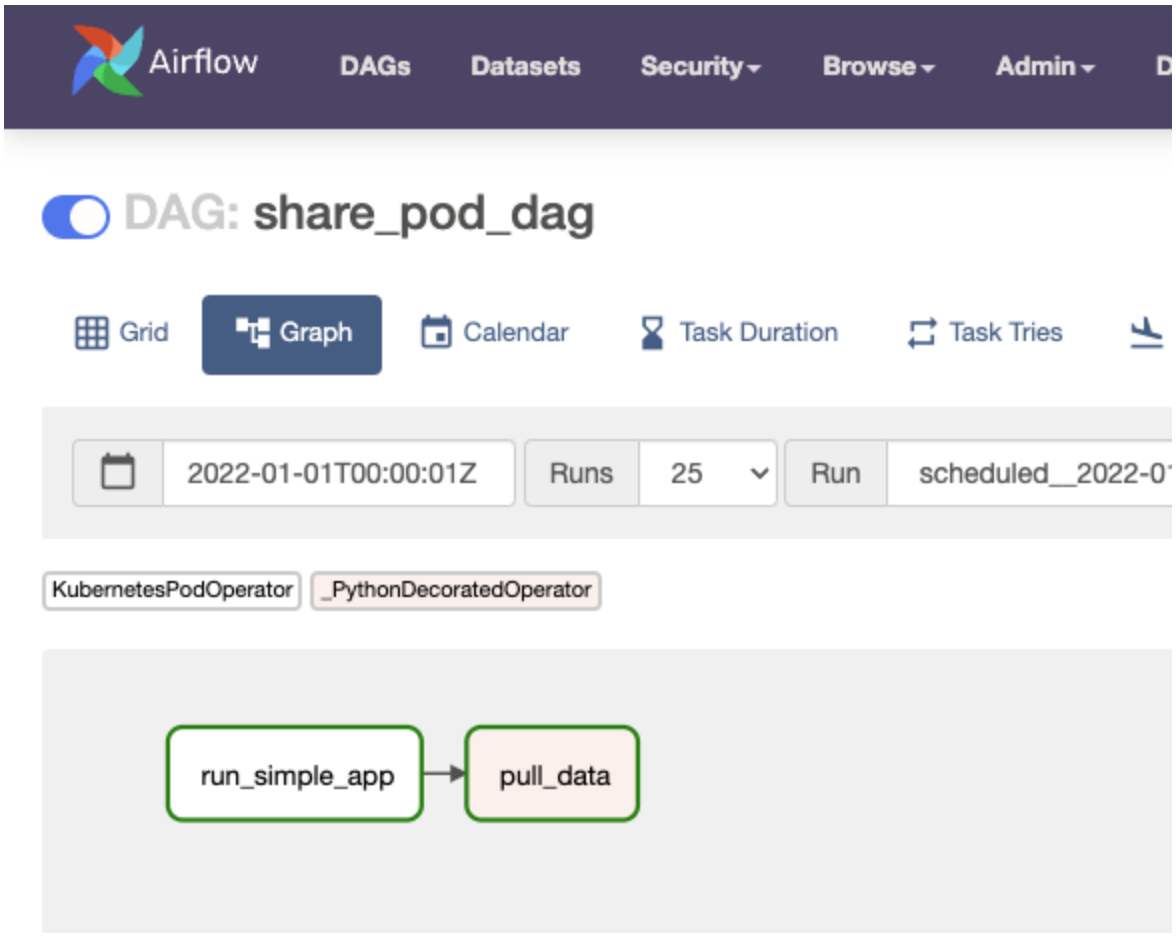
Key	Value
pod_name	airflow-test-pod-7340728f48f14990a29817d72e3aa4bb
pod_namespace	default
return_value	{'output': 42}

As you can see, the KPO created 3 XCOMs. One for the pod name, one for the pod namespace, and the last one that corresponds to the data pushed by the script we ran in the Docker image.

Remember:

```
import os
import json

# that will create a XCOM with the key return_value and the JSON in it
xcom = {
        "output": 42
    }

# write to the file checked by Airflow for XComs
with open('./airflow/xcom/return.json', 'w') as f:
    json.dump(xcom, f)
```

Finally, if you click on the DAG then Graph, and the task pull_data.



You can see in the logs:

```
AIRFLOW_CTX_DAG_OWNER=airflow
AIRFLOW_CTX_DAG_ID=share_pod_dag
AIRFLOW_CTX_TASK_ID=pull_data
AIRFLOW_CTX_EXECUTION_DATE=2022-01-01T00:00:00+00:00
AIRFLOW_CTX_TRY_NUMBER=1
AIRFLOW_CTX_DAG_RUN_ID=scheduled__2022-01-01T00:00:00+00:00
[2022-11-12, 18:58:12 UTC] {logging_mixin.py:120} INFO - {'output': 42}
[2022-11-12, 18:58:12 UTC] {python.py:177} INFO - Done. Returned value was: None
[2022-11-12, 18:58:12 UTC] {taskinstance.py:1401} INFO - Marking task as SUCCESS. da
[2022-11-12, 18:58:12 UTC] {manager.py:67} WARNING - Unable to find an extractor. ta
[2022-11-12, 18:58:12 UTC] {console.py:22} INFO - {"eventTime": "2022-11-12T18:58:12
[2022-11-12, 18:58:12 UTC] {local_task_job.py:164} INFO - Task exited with return co
[2022-11-12, 18:58:12 UTC] {local_task_job.py:273} INFO - 0 downstream tasks schedul
```

Well done! You are now able to push data from the KPO to another task. 🤩