Kedro Airflow K8S Plugin

Release 0.1.2

GetInData

Mar 24, 2021

CONTENTS:

1 Introduction **1**
- 1.1 What is Airflow? . . . 1
- 1.2 What is Kubernetes? . . . 1
- 1.3 Why to integrate Kedro project with Airflow nad Kubernetes? . . . 1

2 Installation **3**
- 2.1 Installation guide . . . 3
- 2.2 Configuration . . . 4

3 Getting started **5**
- 3.1 Quickstart . . . 5
- 3.2 GCP AI Platform support . . . 8
- 3.3 Mlflow support . . . 8

4 Indices and tables **9**

CHAPTER

ONE

INTRODUCTION

1.1 What is Airflow?

Airflow is a platform to programmatically author, schedule and monitor workflows. Workflows are represented as DAGs. Each DAG is represented by nodes, that define job to be executed. The DAGs are stored in the file storage, allowing user to run the pipeline once or schedule the recurring run.

1.2 What is Kubernetes?

Kubernetes is a platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

1.3 Why to integrate Kedro project with Airflow nad Kubernetes?

Airflow's main attitude is the portability. Once you define a pipeline, it can be started on any Kubernetes cluster. The code to execute is stored inside docker images that cover not only the source itself, but all the libraries and entire execution environment. Portability is also one of key Kedro aspects, as the pieplines must be versionable and packagebale. Kedro, with Kedro-docker plugin do a fantastic job to achieve this and Airflow looks like a nice addon to run the pipelines on powerful remote Kubernetes clusters.

CHAPTER

TWO

INSTALLATION

2.1 Installation guide

2.1.1 Kedro setup

First, you need to install base Kedro package in `<17.0` version

> Kedro 17.0 is supported by kedro-airflow-k8s, but not by kedro-mlflow yet, so the latest version from 0.16 family is recommended.

```
$ pip install 'kedro<0.17'
```

2.1.2 Plugin installation

Install from PyPI

You can install `kedro-airflow-k8s` plugin from `PyPi` with `pip`:

```
pip install --upgrade kedro-airflow-k8s
```

Install from sources

You may want to install the develop branch which has unreleased features:

```
pip install git+https://github.com/getindata/kedro-airflow-k8s.git@develop
```

2.1.3 Available commands

You can check available commands by going into project directory and runnning:

```
$ kedro airflow-k8s

Usage: kedro airflow-k8s [OPTIONS] COMMAND [ARGS]...

Options:
-e, --env TEXT  Environment to use.
-h, --help      Show this message and exit.

```

(continues on next page)

(continued from previous page)

```
Commands:
  compile          Create an Airflow DAG for a project
  run-once         Uploads pipeline to Airflow and runs once
  schedule         Uploads pipeline to Airflow with given schedule
  upload-pipeline  Uploads pipeline to Airflow DAG location
```

compile

`compile` command takes one argument, which is the directory name containing configuration (relative to `conf` folder). As an outcome, `dag` directory contains python file with generated DAG.

run-once

`run-once` command generates DAG from the pipeline, uploads it Airflow DAG location and triggers the DAG run as soon as the new DAG instance is available. It optionally allows waiting for DAG run completion, checking if `success` status is returned.

schedule

`schedule` command takes three arguments, one is the directory name containing configuration (relative to `conf` folder), the second one is the output location of generated dag, the third is cron like expression that relates to Airflow DAG `schedule_interval`.

upload-pipeline

`upload-pipeline` command takes two arguments, one is the directory name containing configuration (relative to `conf` folder), the second one is the output location of generated dag.

2.2 Configuration

Plugin maintains the configuration in the `conf/base/airflow-k8s.yaml` file.

```
# Image to be used during deployment to Kubernetes cluster
image: docker.registry.com/getindata/kedro-project-image
# Kubernetes namespace in which Airflow creates pods for node execution
namespace: airflow
# K8S access policy for persistent volumes: ReadWriteOnce or ReadWriteMany
accessMode: ReadWriteOnce
# size of the temporary volume
requestStorage: 1Gi
```

CHAPTER

THREE

GETTING STARTED

3.1 Quickstart

3.1.1 Preprequisites

Although the plugin does not perform deployment, it's recommended to have access to Airflow DAG directory in order to test run the generated DAG.

3.1.2 Install the toy project with Kedro Airflow K8S support

It is a good practice to start by creating a new virtualenv before installing new packages. Therefore, use `virtalenv` command to create new env and activate it:

```
$ virtualenv venv-demo
created virtual environment CPython3.8.5.final.0-64 in 145ms
  creator CPython3Posix(dest=/home/mario/kedro/venv-demo, clear=False, no_vcs_
↪ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle,␣
↪via=copy, app_data_dir=/home/mario/.local/share/virtualenv)
    added seed packages: pip==20.3.1, setuptools==51.0.0, wheel==0.36.2
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,
↪PythonActivator,XonshActivator
$ source venv-demo/bin/activate
```

Then, `kedro` must be present to enable cloning the starter project, along with the latest version of `kedro-airflow-k8s` plugin and kedro-docker.

```
$ pip install 'kedro<0.17' kedro-airflow-k8s kedro-docker
```

With the dependencies in place, let's create a new project:

```
$ kedro new --starter=git+https://github.com/getindata/kedro-starter-spaceflights.git␣
↪--checkout allow_nodes_with_commas
Project Name:
=============
Please enter a human readable name for your new project.
Spaces and punctuation are allowed.
 [New Kedro Project]: Airflow K8S Plugin Demo

Repository Name:
================
Please enter a directory name for your new project repository.
```

(continues on next page)

(continued from previous page)

```
Alphanumeric characters, hyphens and underscores are allowed.
Lowercase is recommended.
 [airflow-k8s-plugin-demo]:

Python Package Name:
====================
Please enter a valid Python package name for your project package.
Alphanumeric characters and underscores are allowed.
Lowercase is recommended. Package name must start with a letter or underscore.
 [airflow_k8s_plugin_demo]:

Change directory to the project generated in ${CWD}/airflow-k8s-plugin-demo

A best-practice setup includes initialising git and creating a virtual environment␣
↪before running
 `kedro install` to install project-specific dependencies. Refer to the Kedro
  documentation: https://kedro.readthedocs.io/
```

TODO: switch to the official `spaceflights` starter after https://github.com/quantumblacklabs/kedro-starter-spaceflights/pull/10 is merged

Finally, go the demo project directory and ensure that kedro-airflow-k8s plugin is activated:

```
$ cd airflow-k8s-plugin-demo/
$ kedro install
(...)
Requirements installed!
$ kedro airflow-k8s --help
```
```console
$ kedro airflow-k8s

Usage: kedro airflow-k8s [OPTIONS] COMMAND [ARGS]...

Options:
-e, --env TEXT  Environment to use.
-h, --help      Show this message and exit.

Commands:
compile  Create an Airflow DAG for a project
upload-pipeline  Uploads pipeline to Airflow DAG location
```

3.1.3 Build the docker image to be used on Kubeflow Pipelines runs

First, initialize the project with `kedro-docker` configuration by running:

```
$ kedro docker init
```

This command creates a several files, including `.dockerignore`. This file ensures that transient files are not included in the docker image and it requires small adjustment. Open it in your favourite text editor and extend the section `# except the following` by adding there:

```
!data/01_raw
```

This change enforces raw data existence in the image. Also, one of the limitations of running the Kedro pipeline on Airflow (and not on local environment) is inability to use MemoryDataSets, as the pipeline nodes do not share

memory, so every artifact should be stored as file. The `spaceflights` demo configures four datasets as in-memory, so let's change the behaviour by adding these lines to `conf/base/catalog.yml`:

```
X_train:
  type: pickle.PickleDataSet
  filepath: data/05_model_input/X_train.pickle
  layer: model_input

y_train:
  type: pickle.PickleDataSet
  filepath: data/05_model_input/y_train.pickle
  layer: model_input

X_test:
  type: pickle.PickleDataSet
  filepath: data/05_model_input/X_test.pickle
  layer: model_input

y_test:
  type: pickle.PickleDataSet
  filepath: data/05_model_input/y_test.pickle
  layer: model_input
```

Finally, build the image:

```
kedro docker build
```

When execution finishes, your docker image is ready. If you don't use local cluster, you should push the image to the remote repository:

```
docker tag airflow_k8s_plugin_demo:latest remote.repo.url.com/airflow_k8s_plugin_
↪demo:latest
docker push remote.repo.url.com/airflow_k8s_plugin_demo:latest
```

3.1.4 Setup GIT repository

Plugin requires project to be under git repository. Perform repository initialization and commit project files

3.1.5 Compile DAG

Create configuration file in `conf/pipelines/airflow-k8s.yaml`:

```
image: remote.repo.url.com/airflow_k8s_plugin_demo:latest
# This should match namespace in Kubernetes cluster, where pods will be created
namespace: airflow
```

Also mlflow configuration has to be set up as described in mlflow section.

Having configuration ready, type:

```
kedro airflow-k8s -e pipelines compile
```

This command compiles pipeline and generates DAG in `dag/airflow_k8s_plugin_demo.py`. This file should be copied manually into Airflow DAG directory, that Airflow periodically scans. After it appears in airflow console, it is ready to be triggered.

As an alternative, one cas use the following:

```
kedro airflow-k8s -e pipelines upload-pipeline -o ${AIRFLOW_DAG_HOME}
```

in order to get DAG copied directly to Airflow DAG folder. Google Cloud Storage locations are also support with `gcs://` or `gs://` prefix in the parameter (this requires plugin to be installed with `pip install kedro-airflow-k8s[gcp]`).

3.2 GCP AI Platform support

Google Cloud's AI Platform offers couple services that simplify Machine Learning tasks.

3.2.1 Using `kedro` with AI Platform Notebooks

AI Platform Notebooks provides an easy way to manage and host JupyterLab based data science workbench environment. What we've found out is that the default images provided by a service cause some dependency conflicts. To avoid this issues make sure you use isolated virtual environment, e.g. virtualenv. New virual environment can be created by simply invoking `python -m virtualenv venv` command.

3.3 Mlflow support

If you use MLflow and kedro-mlflow for the Kedro pipeline runs monitoring, the plugin will automatically enable support for:

- starting the experiment when the pipeline starts,
- logging all the parameters, tags, metrics and artifacts under unified MLFlow run.

To make sure that the plugin discovery mechanism works, add `kedro-mlflow` as a dependencies to `src/requirements.in` and run:

```
$ pip-compile src/requirements.in > src/requirements.txt
$ kedro install
$ kedro mlflow init
```

Then, adjust the kedro-mlflow configuration and point to the mlflow server by editing `conf/local/mlflow.yml` and adjusting `mlflow_tracking_uri` key. Then, build the image:

```
$ kedro docker build
```

And re-push the image to the remote registry.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search