Kedro SageMaker Plugin

Release 0.3.0

GetInData

Apr 04, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 About SageMaker Pipelines

Amazon SageMaker Pipelines is a fully managed service that enables you to build, test, and deploy machine learning (ML) workflows with ease. It provides an easy-to-use Python Software Development Kit (SDK) that allows you to define and execute ML workflows, as well as visualize and manage them using Amazon SageMaker Studio.

One of the key benefits of SageMaker Pipelines is the ability to store and reuse workflow steps, which can help you be more efficient and scale faster. Additionally, SageMaker Pipelines comes with built-in templates to help you quickly set up CI/CD (continuous integration/continuous delivery) in your ML environment, so you can get started quickly and streamline your ML workflows.

1.2 Why to integrate Kedro project with SageMaker Pipelines?

The Kedro Framework is a tool for building and deploying machine learning (ML) pipelines that follows the best standards and practices for ML model development. Once the code is ready, there are now multiple tools available for automating and scaling the delivery of those ML pipelines.

These tools can be used in conjunction with Kedro to execute ML pipelines code using robust services without altering the underlying business logic. The use of these tools can provide resource benefits for handling large training datasets or complex and compute intensive models.

We currently support:

- Azure ML Pipelines kedro-azureml
- GCP Vertex AI Pipelines kedro-vertexai
- Kubeflow Pipelines kedro-kubeflow
- Airflow on Kubernetes kedro-airflow-k8s

With this **kedro-sagemaker** plugin, you can run your Kedro project on Amazon SageMaker Pipelines in a fully managed fashion.

CHAPTER

TWO

INSTALLATION GUIDE

2.1 Prerequisites

- a tool to manage Python virtual environments (e.g. venv, conda, virtualenv).
- Docker

2.2 Kedro setup

First, you need to install base Kedro package

```
$ pip install "kedro>=0.18.3,<0.19"
```

2.3 Plugin installation

2.3.1 Install from PyPI

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

```
pip install --upgrade kedro-sagemaker
```

2.3.2 Install from sources

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

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

2.4 Available commands

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

```
Usage: kedro sagemaker [OPTIONS] COMMAND [ARGS]...

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

Commands:
  compile  Compiles the pipeline to a JSON file
  init     Creates basic configuration for Kedro SageMaker plugin
  run      Runs the pipeline on SageMaker Pipelines
```

CHAPTER

THREE

QUICKSTART

You can go through the written quickstart here or watch the video on YouTube:

Before you start, make sure that you have the following:

- AWS CLI installed
- AWS SageMaker domain
- SageMaker Execution role ARN (in a form *arn:aws:iam::<ID>:role/service-role/AmazonSageMaker-ExecutionRole-<NUMBERS>*). If you don't have one, follow the [official AWS docs](https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html#sagemaker-roles-create-execution-role).
- S3 bucket that the above role has R/W access
- Docker installed
- Amazon Elastic Container Registry (Amazon ECR) repository created that the above role has read access and you have write access

1. Prepare new virtual environment with Python >=3.8. Install the packages

```
pip install "kedro>=0.18.3,<0.19" "kedro-sagemaker"
```

2. Create new project (e.g. from starter). !!! Make sure you don't name it `kedro-sagemaker` because you will overwrite Python module name.

```
kedro new --starter=spaceflights

Project Name
============
Please enter a human readable name for your new project.
Spaces, hyphens, and underscores are allowed.
 [Spaceflights]: kedro_sagemaker_demo

The project name 'kedro_sagemaker_demo' has been applied to:
- The project title in /Users/marcin/Dev/tmp/kedro-sagemaker-demo/README.md
- The folder created for your project in /Users/marcin/Dev/tmp/kedro-sagemaker-demo
- The project's python package in /Users/marcin/Dev/tmp/kedro-sagemaker-demo/src/kedro_
↪sagemaker_demo
```

3. Go to the project's directory: `cd kedro-sagemaker-demo`
4. Add `kedro-sagemaker` to `src/requirements.txt`

5. (optional) Remove `kedro-telemetry` from `src/requirements.txt` or set appropriate settings (https://github.com/kedro-org/kedro-plugins/tree/main/kedro-telemetry).
6. Install the requirements `pip install -r src/requirements.txt`
7. Initialize Kedro SageMaker plugin. Provide name of the S3 bucket and full ARN of the SageMaker Execution role (which should also have access to the S3 bucket). For `DOCKER_IMAGE` - use full name of the ECR repository that you want to push your docker image.

```
#Usage: kedro sagemaker init [OPTIONS] BUCKET EXECUTION_ROLE DOCKER_IMAGE
kedro sagemaker init <bucket-name> <role-arn> <ecr-image-uri>
```

The `init` command automatically will create:

- `conf/base/sagemaker.yml` configuration file, which controls this plugin's behaviour
- `Dockerfile` and `.dockerignore` files pre-configured to work with Amazon SageMaker

8. Adjust the Data Catalog - the default one stores all data locally, whereas the plugin will automatically use S3. Only input data is required to be read locally. Final `conf/base/catalog.yml` should look like this:

```
companies:
  type: pandas.CSVDataSet
  filepath: data/01_raw/companies.csv
  layer: raw

reviews:
  type: pandas.CSVDataSet
  filepath: data/01_raw/reviews.csv
  layer: raw

shuttles:
  type: pandas.ExcelDataSet
  filepath: data/01_raw/shuttles.xlsx
  layer: raw
```

9. (optional) Login to ECR, if you have not logged in before. You can run the following snippet in the terminal (adjust the region to match your configuration).

```
REGION=eu-central-1; aws ecr get-login-password --region $REGION | docker login --
↪username AWS --password-stdin "<AWS project ID>.dkr.ecr.$(echo $REGION).amazonaws.com"
```

10. Run your Kedro project on AWS SageMaker pipelines with a single command:

```
kedro sagemaker run --auto-build -y
```

This command will first build the docker image with your project, push it to the configured ECR and then it will run the pipeline in AWS SageMaker pipelines service.

Finally, you will see similar logs in your terminal:

```
Pipeline ARN: arn:aws:sagemaker:eu-central-1:781336771001:pipeline/kedro-sagemaker-
↪pipeline
Pipeline started successfully
```

Additionally, if you have (kedro-mlflow) plugin installed, an additional node called *start-mlflow-run* will appear on execution graph. It's job is to log the SageMaker's Pipeline Execution ARN (so you can link runs with mlflow with runs in SageMaker) and make sure that all nodes use common Mlflow run.

CHAPTER

FOUR

RESOURCE CUSTOMIZATION

You can configure resources used by your nodes in *sagemaker.yml* under *resources* key

Here is the definition of default values for nodes:

```
resources:
  __default__:
    instance_count: 1
    instance_type: ml.m5.large
    timeout_seconds: 86400
    security_group_ids: null
    subnets: null
```

To specify custom resources just provide node name or node tag below *__default__* configuration

Example custom config:

```
resources:
  __default__:
    instance_count: 1
    instance_type: ml.m5.large
    timeout_seconds: 86400
    security_group_ids: null
    subnets: null
  train_on_gpu_node:
    instance_count: 1
    instance_type: ml.p3.2xlarge
    security_group_ids: ["example-security-group-id"]
    subnets: ["example-subnet-id"]
  some_test_node:
    instance_count: 1
    instance_type: ml.t3.medium
```

The default behavior is that only values defined in node resources will override *__default__* values and the rest is inherited.

So in this example

- *train_on_gpu_node* inherits *timeout_seconds: 86400* from *__default__*
- *some_test_node* inherits *timeout_seconds: 86400*, *security_group_ids: null* and *subnets: null* from *__default__*

CHAPTER

FIVE

DEVELOPMENT

5.1 Prerequisites

- poetry `1.3.11` (as of 2022-12-22)
- Python >= 3.9
- AWS CLI - https://aws.amazon.com/cli/

5.2 Local development

It's easiest to develop the plugin by having a side project created with Kedro (e.g. spaceflights starter), managed by Poetry (since there is no `pip install -e` support in Poetry). In the side project, just add the following entry in `pyproject.toml`:

```
[tool.poetry.dependencies]
kedro-sagemaker = { path = "<full path to the plugin on local machine>", develop = true}
```

and invoke

```
poetry update
poetry install
```

and all the changes made in the plugin will be immediately visible in the side project (just as with `pip install -e` option).

5.3 Starting the job from local machine

Since you need a docker container to run the job in SageMaker Pipelines, it needs to be build first. For fast local development I suggest the following:

1. Once you decide to test the plugin, run `poetry build`. It will create `dist` folder with `.tar.gz` file in it.
2. Go to the side project folder, create a hard-link to the `.tar.gz`: `ln <full path to the plugin on local machine>/dist/kedro-sagemaker-0.1.0.tar.gz kedro-sagemaker-0.1.0.tar.gz`
3. In the `Dockerfile` of the side project add

```
COPY kedro-sagemaker-0.1.0.tar.gz .
RUN pip install ./kedro-sagemaker-0.1.0.tar.gz
```

1. Build the docker with `:latest` tag (make sure that `:latest` is specified in the plugin's config `sagemaker.yml` in `conf`), push the image and run the plugin.
2. Done!

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search