AWS MLOps Python package

Release 1.2.2

Alessandra Bilardi

Feb 22, 2022

CONTENTS:

This package contains the classes for managing the saving of the AWS services.

CHAPTER

ONE

GETTING STARTED

AWS MLOps package is implemented to help you like a framework for deploying what it is necessary to manage your models.

The goal is to implement this package to keep your focus on your prep and test code using the AWS SageMaker services, AWS Step Functions and AWS Lambda.

It is part of the educational repositories to learn how to write stardard code and common uses of the TDD and CI / CD.

1.1 Prerequisites

You can use Serverless framework for deploying the AWS services: if you want to use the guide below, you have to install npm (Node Package Manager) before.

If you want to use another AWS tool, you can see the repository aws-tool-comparison before to implement your version.

1.2 Installation

The package is not self-consistent. So you have to download the package by github and to install the requirements before to deploy the **example** on AWS:

```
git clone https://github.com/bilardi/aws-mlops
cd aws-mlops/
npm install
export AWS_PROFILE=your-account
export STAGE=studio
bash example/deploy.sh
```

Or if you want to use this package into your code, you can install by python3-pip:

```
pip3 install aws_mlops
python3
>>> import aws_mlops
>>> help(aws_mlops)
```

Read the documentation on readthedocs for

- Usage
- Development

1.3 Change Log

See CHANGELOG.md for details.

1.4 License

This package is released under the MIT license. See LICENSE for details.

CHAPTER

TWO

USAGE

The AWS Step Functions manages your ML cycle of data processing, modeling and testing or prediction with your best model. You can find all python scripts that you have to prepare for your MLOps solution in the folder **example**:

- **config.py**, it contains all you want to configure for using the aws_mlops library
- **definitions.py**, it contains all you want to configure for creating your step functions
- **processing.py**, probably you can copy it and it is not necessary to modify it
- **Dockerfile**, it contains what you need for running the script named processing.py
- **prep_with_pandas.py**, it contains your code for data processing and it is loaded by processing.py
- **test_with_pandas.py**, it contains your code for testing and it is the script loaded by processing.py

You can also find the *ipynb* files that they are useful to prepare your python scripts:

- **prep_with_pandas.ipynb** for **prep_with_pandas.py**
- **test_with_pandas.ipynb** for **test_with_pandas.py**

And there are some bash scripts for creating your CI / CD system:

- **test.sh** for testing python library and step functions deployment
- **test_docker.sh** for testing the scripts you can call from processing.py
- **build_image.sh** for building your image and saving it on AWS ECR
- **deploy.sh** for deploying step functions and lambda of your infrastructure

2.1 Example

You need an infrastructure with a process for

- preparing the raw data for your training by AWS Sagemaker Autopilot
- running Autopilot
- inference with your best model and your test data
- testing and saving the prediction data, metrics and attributes used

When you know the initial hyperparameters that you can use, you can setup the config.py and

- preparing the raw data for your training
- training and tuning your model
- inference with your best model and your test data

- testing and saving the prediction data, metrics and attributes used

And the last two points have to be usable for

- inference with your best model and your new data
- saving the prediction data, metrics and attributes used

When you have prepared the python scripts listed above, you have to

- commit your changes and push on your repo
- proceed with the commands for deploying described in the Development Section and paragraph **Deploy on AWS**

When you have deployed the infrastructure, you can use the **example/mlops.ipynb** for calling the whole cycle or only a specific piece.

The secret is to version all: data, code and model that you use for defining that prediction. It is important to version any change for the analysis step.

If you need to improve your configuration or your scripts, the best way is

- commit any change of your python scripts listed above, thus the s3 key will be different for commit
- if you change only the raw/new data, the s3 key would be different for datatime, but you also can fix it for your testing
- if you have to test your change, deploy an infrastructure for your branch, thus the s3 key will be different from production

CHAPTER

THREE

DEVELOPMENT

The environments for development can be many: you can organize a **CI/CD system** with your favorite software. The primary features of your CI/CD are: having a **complete environment for**

- **development** for each developer, to implement something and for running unit tests
- **staging** for running unit and integration tests, to check everything before release
- **production**

If you want to use AWS CDK and AWS CodePipeline, you can see these repositories before to implement your version

- aws-simple-pipeline for using a library ready
- aws-tool-comparison for seeing its implementation

When you add the data management in your CD cycle, you have to add the data versioning:

- the system improved in the folder named example, it provides s3 key with branch, environment, commit and datatime
- so you can have a **complete environment for** each combo of them

This is important to commit any change for the analysis step.

3.1 Run tests

```
cd aws-mlops/
npm install
pip3 install --upgrade -r example/requirements.txt
python3 -m unittest discover -v
# even with functional and infrastructure tests
export AWS_PROFILE=your-account
bash example/test.sh
```

3.2 Improve your python scripts for processing by Jupyter

```
cd aws-mlops/
docker run --rm -p 8888:8888 -e JUPYTER_ENABLE_LAB=yes -e AWS_PROFILE=your-account -v
↪$HOME/.aws/credentials:/home/jovyan/.aws/credentials:ro -v "$PWD":/home/jovyan/␣
↪jupyter/datascience-notebook
```

You can find two ipynb files in the folder named example: they can help you to improve your code for the processing steps.

3.3 Test your python scripts

If you did never push an image on your repository, run the commands of **Deploy on AWS** paragraph before run the docker.

```
cd aws-mlops/
export AWS_PROFILE=your-account
export STAGE=development
bash example/test_docker.sh # and with bash example/test.sh for all
```

3.4 Deploy on AWS

```
cd aws-mlops/
export AWS_PROFILE=your-account
export STAGE=development
bash example/deploy.sh
```

3.5 Remove on AWS

The stack has the tags necessary for being deleted itself, if you use the aws-saving. Or you can run the commands below to remove by Serverless only the environment that you want to delete:

```
cd aws-mlops/
export AWS_PROFILE=your-account
SLS_DEBUG=* sls remove --stage development
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search