

---

# **AWS simple pipeline**

***Release 0.1.0***

**Alessandra Bilardi**

**Apr 10, 2021**



**CONTENTS:**

<b>1</b>	<b>Getting started</b>	<b>3</b>
1.1	Prerequisites . . . . .	3
1.2	Installation . . . . .	3
1.3	Change Log . . . . .	4
1.4	License . . . . .	4
<b>2</b>	<b>Usage</b>	<b>5</b>
2.1	Example . . . . .	5
<b>3</b>	<b>Development</b>	<b>7</b>
3.1	Run tests . . . . .	7
3.2	Deploy on AWS . . . . .	7
3.3	Remove on AWS . . . . .	7
<b>4</b>	<b>Indices and tables</b>	<b>9</b>



This package contains the classes for deploying an AWS simple pipeline.



## GETTING STARTED

AWS simple pipeline package is implemented for deploying a Continuous Deployment or Delivery system (CD) by AWS CodePipeline service.

You can use this simple pipeline for deploying your personal solution in 2 environments: staging and production.

It is part of the [educational repositories](#) to learn how to write standard code and common uses of the TDD, CI and CD.

### 1.1 Prerequisites

You have to install the [AWS Cloud Development Kit](#) (AWS CDK) for deploying the AWS simple pipeline:

```
npm install -g aws-cdk # for installing AWS CDK
cdk --help # for printing its commands
```

And you need an AWS account, in this repository called **your-account**.

### 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 on AWS:

```
git clone https://github.com/bilardi/aws-simple-pipeline
cd aws-simple-pipeline/
pip3 install --upgrade -r requirements.txt
export AWS_PROFILE=your-account
cdk deploy
```

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

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

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.



## USAGE

The **aws\_simple\_pipeline** package reads the file named **buildspec.yml** that it finds in the same directory of **app.py** file, where you have to initialize its PipelineStack class.

You can describe all steps that you need, directly in the **buildspec.yml** file, or you can run an external script for each step, that you can test it on your client.

You have to manage a git token in **app.py**, and you can create it by [AWS console](#) or **aws-cli**:

```
aws secretsmanager create-secret \
  --name /aws-simple-pipeline/secrets/github/token \
  --secret-string '{"github-token":"YOUR_TOKEN"}'
```

There are many methods for creating a secret object because it can be replicated automatically, but it is not the purpose of this guide. Now, we only need to create it once for all our implementations.

## 2.1 Example

You need to create the infrastructure of your **aws-saving** solution for

- your test, because you want to improve a feature
- a CI/CD system, because you want to use **aws-saving** solution on your AWS account

### 2.1.1 For the Continuous Integration (CI)

You can use some bash scripts for testing each step

- **local.sh**, for running all bash scripts with one command
- **build.sh**, for loading all requirements
- **unit\_test.sh**, for testing the code
- **deploy.sh**, for deploying on AWS account the infrastructure of your **aws-saving** solution
- **integration\_test.sh**, for testing the resources integration

## 2.1.2 For the CD system

You have to use the files **app.py** and **buildspec.yml**

- CD is Continuous Delivery, if you set `manual_approval_exists = True` on the file **app.py**
- CD is Continuous Deployment, if you set `manual_approval_exists = False` on the file **app.py**

You can save the **buildspec.yml** file in the same directory of **app.py** file, and it will be loaded without defining anything.

Or you can also save it in another folder,

- you have to set `buildspec_path = 'relative/path/from/repo/root/buildspec.yml'` on the file **app.py**
- you can find some examples on the follow repositories
  - [aws-tool-comparison/cdk/python/app.py](#), where the `buildspec_path` is defined
  - [aws-static-gui-resources/app\\_pipeline.py](#), where the `buildspec_path` is not defined because it is the default

## 2.1.3 For managing many environments in parallel

If you use the command `cdk deploy`, you will create a pipeline with that project name with two environments: one named **staging** and one named **production**.

But if you need to manage more environments, like for `my-development`, `your-development`, and so on, you can use at least two methods:

- you can use the command `cdk deploy -c stage=my-development`, as described in [Development section](#)
- or you can use the property **stage** at the initialization, as used in [aws-static-gui-resources](#) where the stage is the branch name

## 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**

With AWS CDK system, you can create an AWS CodePipeline for each environment!

### 3.1 Run tests

For running the unit tests, you need only your client: you can use a [virtual environment](#)

```
cd aws-simple-pipeline/  
pip3 install --upgrade -r requirements.txt  
python3 -m unittest discover -v
```

### 3.2 Deploy on AWS

AWS CDK system allows you to create an AWS CodePipeline for each environment by adding a contextual string parameter (in the sample is **stage**) !

```
cd aws-simple-pipeline/  
export AWS_PROFILE=your-account  
export STAGE=my-development  
cdk deploy '*' -c stage=${STAGE}
```

### 3.3 Remove on AWS

You can destroy the resources with a simple command

```
cd aws-simple-pipeline/  
export AWS_PROFILE=your-account  
export STAGE=my-development  
cdk destroy '*' -c stage=${STAGE}
```

If you want to see other sample of AWS CDK commands, you can see

- the repository named `aws-tool-comparison` or its documentation
- the repository named `aws-static-gui-resources` or its documentation

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`