AWS static website

Release 0.0.3

Alessandra Bilardi

Feb 22, 2022

CONTENTS:

CHAPTER ONE

GETTING STARTED

AWS static website package is implemented for deploying a bucket with its Cloudfront distribution and its domain name.

You can use this package for deploying a static website on the bucket deployed.

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

1.1 Prerequisites

You have to install the AWS Cloud Development Kit (AWS CDK) for deploying the AWS resources:

```
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-static-website
cd aws-static-website/
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_static_website
python3
>>> import aws_static_website
>>> help(aws_static_website)
```

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_static_website** package deploys the resources by the file named **app.py** file, where you have to initialize its WebsiteStack class.

You can manage all configuration that you need, directly in the **app.py** file.

2.1 Example

You have chosen the domain name named **domain.name** and your subname will be **bucket**: your `bucket` has to be named **bucket.domain.name**.

2.1.1 Only S3 and Cloudfront

If you want to use the url provided from S3 service, you only have to configure the `index_document` and `error_document` properties

```
project_name = "aws-static-website"
website_params = {
    "index_document": "index.html",
    "error_document": "index.html"
}

app = core.App()
WebsiteStack(app,
    id=project_name,
    bucket_name="bucket.domain.name",
    website_params=website_params
)
```

You can find a complete example in this repo.

2.1.2 Even DNS

If you want to use the url **bucket.domain.name**, you also have to configure the hosted zone:

- you can pass the hosted both `zone_name` and `zone_id`, and the package will only deploy the **DNS record type A**

```
project_name = "aws-static-website"
website_params = {
    "index_document": "index.html",
    "error_document": "index.html"
}
hosted_params = {
    "zone_name": "domain.name",
    "zone_id": "Z23ABC4XYZL05B"
}

app = core.App()
WebsiteStack(app,
    id=project_name,
    bucket_name="bucket.domain.name",
    website_params=website_params,
    hosted_params=hosted_params
)
```

- or you can only pass the hosted `zone_name`, and the package will deploy the **Hosted Zone** and the **DNS record type A**

```
project_name = "aws-static-website"
website_params = {
    "index_document": "index.html",
    "error_document": "index.html"
}
hosted_params = {
    "zone_name": "domain.name"
}

app = core.App()
WebsiteStack(app,
    id=project_name,
    bucket_name="bucket.domain.name",
    website_params=website_params,
    hosted_params=hosted_params
)
```

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**

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-static-website/
pip3 install --upgrade -r requirements.txt
python3 -m unittest discover -v
```

3.2 Deploy on AWS

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

```
cd aws-static-website/
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-static-website/
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-static-gui-resources or its documentation.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search