AWS static GUI resources and Auth0

Release 0.0.1

Alessandra Bilardi

Feb 22, 2022

CONTENTS:

CHAPTER

ONE

GETTING STARTED

The static website with auth0 is a nice static website in ReactJS. In this repository there is only the application for deploying the AWS resources of a static website with auth0. AWS static website is implemented in **AWS CDK** with **Python**.

It uses the packages

- aws_simple_pipeline for managing the Continuous Deployment
- aws_static_website for deploying the Website resources
- aws_saving for saving on AWS costs

It is part of the educational repositories to learn how to write stardard 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 static website:

```
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-gui-resources
cd aws-static-gui-resources/
pip3 install --upgrade -r requirements.txt
export AWS_PROFILE=your-account
cdk deploy -a 'python app_pipeline.py' -c stage=sample
```

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 GUI resources contains the scripts for deploying all you need for CI / CD management.

- this repo inherits all scripts of aws_simple_pipeline
- it works with aws_static_website
- and it takes advantage of aws_saving

The packages allow you to manage many environments in parallel by the parameter named **stage**:

- it can be a contextual string parameter as described in *Development section*
- or it can be a parameter of the package initialiazed as implemented in the **app_pipeline.py** where it is the branch name

2.1 Example

You need to create the infrastructure of your static website and you want to use an Auth0 application by Google

- you have to create an Connect Apps to Google
- and then, you can use the domain created by Auth0 and clientId for logging in your static website

2.1.1 Connect Apps to Google

When you have

- created your ID client OAuth 2.0 on API credentials section,
- and configured your Auth0 connection,

You can configure your Auth0 application with the names of your buckets used on **Allowed Callback URLs**:

- for your local tests when you run your static website by `run start` (see its README.md), http://localhost:3000/callback
- for your environment named **sample** that you run by app_pipeline.py `-c stage=sample` (see *Getting started*), you have to add the domain name of your buckets, in this example they are
  - http://staging-sample-bucket.domain.name.s3-website-eu-west-1.amazonaws.com/callback
  - http://production-sample-bucket.domain.name.s3-website-eu-west-1.amazonaws.com/callback
- for your production environment that you run without stage, in this example, the domain names are
  - http://staging-bucket.domain.name.s3-website-eu-west-1.amazonaws.com/callback

  - http://bucket.domain.name.s3-website-eu-west-1.amazonaws.com/callback

2.1.2 Changes

The files that you have to update on your static website are three:

- **reactJS/src/Auth/Auth.js**, for managing more environment and so more callback URLs
- **reactJS/src/Auth/auth0-variables.js**, for changing the Auth0 details
- **serverless/serverless.yml**,
  - for reducing the service name that it has not to have more than 64 characters
  - for upgrading the nodejs version
  - for changing the Auth0 details

In this commit, you can find an example of a change.

2.1.3 Saving

It is simple to use **aws_saving**: you only have to add some tags and deploy it!

In these commits, you can find an example of where to change:

- on pipeline and website resources by AWS CDK
- on application resources by Serverless framework

CHAPTER

THREE

DEVELOPMENT

This repo contains,

- **app.py** files of the **aws_simple_pipeline** and **aws_static_website** packages
- bash scripts for automation of **aws_simple_pipeline**

A possible improvement is to use AWS CDK system for replacing the Serverless Framework. Then, the unit test and integration test scripts will work.

3.1 Run scripts

For running all scripts, you need only your client: you can use a virtual environment

```
cd aws-static-gui-resources/
STAGE=my-development bash local.sh
```

This step is important for testing all process from building to deploying.

3.2 Deploy on AWS

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

This step is also useful when you need to update a policy for AWS Codebuild or other Pipeline configuration.

```
cd aws-simple-pipeline/
export AWS_PROFILE=your-account
export STAGE=my-development
cdk deploy -a 'python app_pipeline.py' -c stage=${STAGE}
```

or, if you want to use the branch name like the stage name, here is the example with branch named **my-development**

```
cd aws-simple-pipeline/
git checkout -b my-development
export AWS_PROFILE=your-account
cdk deploy -a 'python app_pipeline.py'
```

3.3 Remove on AWS

If you use the saving tags, you can forget to destroy all resources because they will be deleted at time that you will have specified by saving lambda that you will have deployed.

Alternatively, you can destroy the resources with a few commands

```
cd aws-static-gui-resources/
export AWS_PROFILE=your-account
export STAGE=my-development
cdk destroy -a 'python app_pipeline.py' -c stage=${STAGE}
cdk destroy -a 'python app_website.py' -c stage=staging-${STAGE} # it is created by aws_
↪simple_pipeline
cdk destroy -a 'python app_website.py' -c stage=production-${STAGE} # it is created by␣
↪aws_simple_pipeline
```

or, if you want to use the branch name like the stage name, here is the example with branch named **my-development**

```
cd aws-static-gui-resources/
git checkout my-development
export AWS_PROFILE=your-account
export STAGE=my-development
cdk destroy -a 'python app_pipeline.py'
cdk destroy -a 'python app_website.py' -c stage=staging-${STAGE} # it is created by aws_
↪simple_pipeline
cdk destroy -a 'python app_website.py' -c stage=production-${STAGE} # it is created by␣
↪aws_simple_pipeline
```

The methods above, they are missing to destroy some objects, so alternatively you can use these commands

```
cd aws-static-gui-resources/
git checkout my-development
export AWS_PROFILE=your-account
export STAGE=my-development
STAGE=${STAGE} bash destroy.sh
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search