Hidden Words – Projet Final

Dossier d'exploitation

Version 1.1

Auteur
Axel Mesnard
Développeur Python

TABLE DES MATIERES

Versions

Auteur	Date	Description	Version
Axel	13/10/2021	Création du document	1.0
Axel	19/10/2021	Ajout des derniers éléments	1.1

1 - Introduction

1.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application Hidden Words.

Contexte :

Faisant partie d'un groupe qui est fan des mots fléchés, mots cachés/mêlés, etc, je me suis rendu compte qu'il pouvait être intéressant de créer un générateur de mots cachés/mêlés afin qu'ils n'aient plus à en acheter. Après m'être rendu compte qu'un générateur de mots cachés/mêlés, afin que nous n'ayons plus à en acheter en commerce.

Objectif du document :

Ce dossier permet la mise en place de l'application.

1.2 - Références

Pour de plus amples informations, se référer :

1. **DCF** : Dossier de conception fonctionnelle.
2. **DCT** : Dossier conception technique.

2 - PROCEDURE DE DEPLOIEMENT

2.1 - Configuration d'AWS

Comme vu dans le dossier de conception technique, nous allons utiliser presque exclusivement AWS afin de déployer notre application Hidden Words.

2.1.1 - Procédure de configuration de AWS CLI

Nous allons principalement utiliser l'interface GUI d'AWS pour pouvoir déployer notre application mais il est quand même important de connecter notre terminal à AWS CLI afin de pouvoir exécuter quelques commandes comme par exemple un makefile qui permets d'uploader directement notre code sur AWS Lambda. Une automatisation très utile. Ou pour pouvoir lancer l'insertion de la database sur DynamoDB.

Pour installer AWS CLI sur Ubuntu :

```
sudo apt-get update
sudo apt-get install awscli
```

Pour Mac et Windows, vous pouvez suivre les instructions sur la doc officielle d'AWS :

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

Pour connecter votre compte AWS à AWS CLI :

```
aws configure
```

Puis vous pouvez suivre les instructions sur le terminal afin de terminer la procédure. Pour information, la "Secret Access Key ID" peut être créé dans la Management Console d'AWS.

Pour une explication plus détaillée, vous pouvez suivre la documentation officielle d'AWS :

https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html

2.1.2 - Insertion de la database dans DynamoDB

Premièrement vous devez créer une table sur votre compte AWS via DynamoDB. La procédure est simple, vous pouvez utiliser la documentation AWS pour cela :

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-1.html

Une fois la table créée, mettez son nom dans le fichier constants.py, dans "TABLE_NAME". Vous pouvez maintenant lancer l'insertion de la DB directement via AWS CLI, avec cette commande :

```
python3 database/init_database.py
```

!!! ATTENTION !!! Cela peut prendre du temps, le scraping des mots du dictionnaire est long, attendez-vous à plusieurs heures. Le temps peut être réduit si vous modifiez le site de scraping et que vous enlevez les définitions par exemple.

2.1.3 - Fonction Lambda et connexion avec API Gateway

La création de la fonction lambda peut s'effectuer simplement en suivant la documentation d'AWS :

https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html#gettingstarted-zip-function

La connexion avec API Gateway se fait simplement, la procédure entière est trouvable sur la documentation d'AWS :

https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway.html

2.1.4 - Création de vos buckets pour le site web statique

Afin de pouvoir héberger votre site web statique, nous allons utiliser le service d'hébergement Amazon S3.

La procédure de création peut-être trouver ici :

https://docs.aws.amazon.com/AmazonS3/latest/userguide/create-bucket-overview.html

Attention, ici je vous conseille fortement de créer deux buckets séparés :

- Le premier pour héberger le site web en statique, le bucket devra donc être ouvert au public afin que le site soit accessible. Pour cela, la procédure est la suivante : https://aws.amazon.com/fr/premiumsupport/knowledge-center/read-access-objects-s3-bucket/
- Le deuxième bucket servira à héberger les PDF générés par l'application. Afin de télécharger un PDF généré, un lien pré-signé est généré, qui expire en 3600 secondes. Ce bucket restera évidemment privé.

2.1.5 - Ajout du code sur la fonction Lambda

Maintenant que tous nos services sont configurés, il faut maintenant uploader notre code sur la fonction Lambda qui fera tournée le tout.

Pour cela c'est très simple, un "makefile" est disponible à la racine. Il permet de zipper la totalité du code avant de l'envoyer directement sur votre Lambda si vous êtes bien connecté via AWS CLI.

Si le nom de votre lambda n'est pas "hidden_words", faites bien attention à changer le nom dans le makefile afin de mettre le nom de votre fonction lambda.

2.2 - Dernières mises en place

Afin de pouvoir connecter le bouton html qui lance votre fonction lambda, il faut updater l'URL de la fonction dans "index.html" ainsi que "main.js". Changer l'adresse d'amazonaws par la vôtre. Vous pourrez trouver la vôtre dans API Gateway, quand vous avez créé votre requete GET. Dans la partie "Stages", vous trouverez votre "Invoke URL".

Une fois l'adresse de la fonction lambda updatée, vous pouvez accéder à votre site via le bucket S3 du website, où vous l'adresse direct vers l'index.html vous ai donné.


3 - Integration continue

3.1 - GitHub Actions

Grâce à GitHub Actions qui est un service d'automatisation qui s'intègre parfaitement à Github, vous pourrez générer un environnement de développement qui permettra de tester votre application et voir si les changements que vous venez d'apporter à cette dernière ne casse rien. Cela permet de pousser du code sur Github, le tester pour voir si tout fonctionne correctement, et si c'est le cas vous pourrez le passer en production.

Voici le fichier de configurations de GitHub Actions (github-ci.yaml dans .github/workflows/) :

```
# This workflow will install Python dependencies, run tests and lint with a single version
of Python
# For more information see: https://help.github.com/actions/language-and-framework-
guides/using-python-with-github-actions

name: Python application

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python 3.8.10
      uses: actions/setup-python@v2
      with:
        python-version: "3.8.10"
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install flake8 pytest boto3 scrapy beautifulsoup4 requests unidecode pytest-cov
moto pylint selenium webdriver-manager reportlab
    - name: Lint with flake8
      run: |
        # stop the build if there are Python syntax errors or undefined names
        flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
        # exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide
        flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics
```

```
    - name: Test with pytest
      run: |
        pytest
```

Avec ce fichier de configuration, quand du code sera push dans GitHub, vous trouverez une task dans l'onglet GitHub Actions qui lancera les tests pytest mais aussi un check flake8 pour voir si votre code respecte les normes d'écriture.

4 - Supervision / Monitoring

4.1 - Sentry

Sentry est une plate-forme de suivi des erreurs qui vous permet de surveiller les problèmes dans vos déploiements de production. Il prend en charge les langages de programmation et les frameworks les plus populaires, il y a aussi une option pour pouvoir le configurer sur AWS Lambda.

Installation :

Vous avez simplement à vous inscrire sur Sentry.io. Une fois fait, vous allez ajouter votre DSN fourni par Sentry, dans le fichier constants.py.

Documentation : https://docs.sentry.io/platforms/python/guides/django/