Rapport Technique - Pokémon Scraper

1. Architecture mise en place

Vue d'ensemble

Le projet implémente un scraper automatisé qui récupère les images de Pokémon depuis Bulbapedia et les stocke dans un bucket S3 AWS, en utilisant une instance EC2 comme environnement d'exécution.

Composants principaux

Instance EC2 (t2.micro)

- Système : Ubuntu 22.04 LTS
- Rôle : Environnement d'exécution du script Python
- Configuration : Python 3.10 + bibliothèques (requests, beautifulsoup4, boto3, lxml)

Bucket S3 (pokemon-images-khalil)

- Région : eu-west-3 (Paris)
- Structure : images/pokemon/XXX_NomPokemon.png
- Accès : Public via politique de bucket

Utilisateur IAM (pokemon-scraper)

- Permissions : Accès S3 limité (principe du moindre privilège)
- Authentification : Clés d'accès configurées via AWS CLI

Flux de données

1. **Scraping** : EC2 récupère le contenu HTML de Bulbapedia
2. **Téléchargement** : EC2 télécharge les images des Pokémon
3. **Stockage** : EC2 uploade vers S3 via boto3
4. **Accès** : Les images sont accessibles via URLs HTTPS publiques

2. Choix techniques

Source de données : Bulbapedia

Justification :

- Structure HTML stable et prévisible
- Images de qualité officielle

- Respect du robots.txt

Avantages :

- Données exhaustives (tous les Pokémon par génération)
- Format standardisé des images
- URLs d'images directement accessibles

Environnement d'exécution : EC2

Justification :

- Contrôle total de l'environnement Python
- Capacité de traitement suffisante pour le scraping
- Intégration native avec les services AWS
- Coût optimisé (t2.micro Free Tier)

Alternative écartée : Lambda rejetée car timeout de 15 minutes insuffisant pour scraper ~900 Pokémon.

Stockage : Amazon S3

Justification :

- Stockage objet adapté aux images
- Accès public configurable
- Intégration parfaite avec boto3
- Durabilité et disponibilité élevées
- Coût optimisé (5GB Free Tier)

Technologies utilisées

Python 3.10

- Écosystème riche pour le web scraping
- Boto3 pour l'intégration AWS native

BeautifulSoup4

- Parsing HTML robuste et intuitif
- Gestion des encodages et malformations

Requests

- HTTP client fiable avec gestion d'erreurs
- Session persistante pour optimiser les performances

Boto3

- SDK AWS officiel

- Gestion automatique des credentials et retry

3. Gestion des erreurs et robustesse

Résilience réseau

- **Retry automatique** avec backoff exponentiel (max 2 tentatives)
- **Timeout** configuré (10 secondes par requête)
- **Validation du content-type** pour s'assurer du format image

Respect du serveur source

- **Délai entre requêtes** : 2 secondes pour éviter la surcharge
- **User-Agent identifiant** : Projet éducationnel déclaré
- **Gestion des erreurs HTTP** : 404, 500, etc.

Sécurité

- **Clés AWS via profil IAM** : Aucune clé en dur dans le code
- **Permissions minimales** : Utilisateur IAM limité à S3
- **Validation des données** : Nettoyage des noms de fichiers

Logging complet

- **Traçabilité** : Chaque étape loggée (INFO, WARNING, ERROR)
- **Débuggage** : Messages détaillés pour identifier les problèmes
- **Métriques** : Compteurs de succès/échecs

4. Limitations et optimisations possibles

Limitations actuelles

- **Volume** : Script limité à 10 Pokémon pour respecter Free Tier
- **Sequential processing** : Téléchargements un par un
- **Pas de cache** : Re-téléchargement si relance

Optimisations possibles

- **Threading** : Téléchargements parallèles
- **Cache local** : Éviter les re-téléchargements
- **Base de données** : Métadonnées Pokémon (type, génération)
- **Monitoring** : CloudWatch pour métriques

5. Démonstration de fonctionnement

Résultats obtenus

- **10 images** de Pokémon téléchargées et stockées
- **100% de taux de réussite** après correction des permissions S3
- **URLs publiques fonctionnelles** :
    - https://pokemon-images-khalil.s3.eu-west-3.amazonaws.com/images/pokemon/001_Bulbasaur.png
    - https://pokemon-images-khalil.s3.eu-west-3.amazonaws.com/images/pokemon/002_Ivysaur.png
    - etc.

Métriques de performance

- **Temps d'exécution** : ~25 secondes pour 10 Pokémon
- **Taille moyenne** : 9KB par image
- **Bande passante** : ~90KB total
- **Coût** : 0€ (Free Tier respecté)

Vérifications

- **Console S3** : 10 fichiers présents dans le bucket
- **URLs publiques** : Accès direct aux images dans le navigateur
- **Logs** : Aucune erreur dans l'exécution finale