Rapport

Rapport de Projet - Pipeline de Données OpenFoodFacts

1. Introduction

Ce rapport présente la conception, la mise en œuvre et l'orchestration d'un pipeline de données basé sur le dataset **OpenFoodFacts**. Le projet a pour objectif d'automatiser l'ingestion, le nettoyage, la validation, l'enrichissement, ainsi que le stockage des données dans un Data Lake basé sur **HDFS**.

2. Architecture Globale du Pipeline

2.1. Vue d'ensemble

Le pipeline est structuré en plusieurs étapes interconnectées :

1. **Ingestion des données**
2. **Nettoyage et validation des données**
3. **Enrichissement des données**
4. **Orchestration des workflows (via Airflow)**

2.2. Diagramme de l'architecture

(Insérer un diagramme de flux montrant les différentes étapes du pipeline)

3. Ingestion des Données

3.1. Source des données

- Dataset téléchargé depuis : OpenFoodFacts Dataset
- API OpenFoodFacts pour les mises à jour : OpenFoodFacts API

3.2. Méthodologie

- Téléchargement automatique via des jobs programmés
- Stockage des données brutes sur HDFS dans `/user/ubuntu/off_raw/`

3.3. Commandes utilisées

```
wget https://fr.openfoodfacts.org/data -O
/home/workspace/data/off_raw/dataset.csv
hdfs dfs -put /home/workspace/data/off_raw/dataset.csv /user/ubuntu/off_raw/
```

(Inclure des captures d'écran de HDFS montrant les données ingérées)

4. Nettoyage et Validation des Données

4.1. Objectifs

- Suppression des doublons
- Gestion des valeurs manquantes
- Uniformisation des formats (dates, unités de mesure)

4.2. Traitements effectués

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

spark = SparkSession.builder.appName("DataCleaning").getOrCreate()

# Chargement des données
df = spark.read.csv("hdfs:///user/ubuntu/off_raw/dataset.csv", header=True,
inferSchema=True)

# Nettoyage
df_clean = df.dropDuplicates().fillna({"energy_100g": 0, "product_name":
"Unknown"})
df_clean = df_clean.withColumn("date", col("date").cast("timestamp"))

# Stockage des données nettoyées
df_clean.write.parquet("hdfs:///user/ubuntu/off_clean/cleaned_data.parquet")
```

(Inclure des captures d'écran montrant la structure des données nettoyées sur HDFS)

5. Enrichissement des Données

5.1. Méthodologie

- Jointures avec des datasets externes (par exemple, données démographiques par pays)

- Calcul d'indicateurs dérivés (scores nutritionnels, etc.)

5.2. Exemples de transformations

```
# Enrichissement par jointure avec un dataset démographique
external_data =
spark.read.csv("hdfs:///user/ubuntu/external_data/demographics.csv",
header=True)
df_enriched = df_clean.join(external_data, on="country", how="left")

# Calcul d'un indicateur dérivé
df_enriched = df_enriched.withColumn("score_per_person",
col("nutrition_score") / col("population"))

# Stockage des données enrichies
df_enriched.write.parquet("hdfs:///user/ubuntu/off_enriched/enriched_data.parq
uet")
```

(Inclure des captures d'écran montrant les données enrichies sur HDFS)

6. Orchestration avec Airflow

6.1. Configuration

- Déploiement de DAGs Airflow pour orchestrer les différentes étapes du pipeline
- Gestion des dépendances entre les tâches : ingestion → nettoyage → enrichissement

6.2. Exemple de DAG Airflow

```
from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2024, 1, 1),
    'retries': 1
}

dag = DAG('openfoodfacts_pipeline', default_args=default_args,
schedule_interval='@weekly')

ingestion = BashOperator(
```

```
    task_id='ingest_data',
    bash_command='python3 /path/to/ingestion.py',
    dag=dag
)

cleaning = BashOperator(
    task_id='clean_data',
    bash_command='python3 /path/to/cleaning.py',
    dag=dag
)

enrichment = BashOperator(
    task_id='enrich_data',
    bash_command='python3 /path/to/enrichment.py',
    dag=dag
)

ingestion >> cleaning >> enrichment
```

(Inclure des captures d'écran de l'interface Airflow avec l'état des DAGs)

7. Architecture de Stockage des Données

7.1. Structure du Data Lake

```
/user/ubuntu/
├── off_raw/
├── off_clean/
└── off_enriched/
```

(Inclure des captures d'écran des dossiers sur HDFS)

8. Choix Technologiques

- **Langages** : Python, PySpark
- **Orchestrateur** : Apache Airflow
- **Stockage** : HDFS
- **Base de données** : MySQL (si applicable)
- **Outils de traitement** : Hadoop, Spark

(Justifier les choix technologiques en fonction des performances, de la scalabilité, etc.)

9. Conclusion

- Résumé des résultats obtenus
- Améliorations possibles pour le futur

10. Annexes

- Captures d'écran supplémentaires
- Codes sources (avec des références vers les scripts Python/Spark)

Auteur : [Votre Nom]
Date : [Date de Soumission]