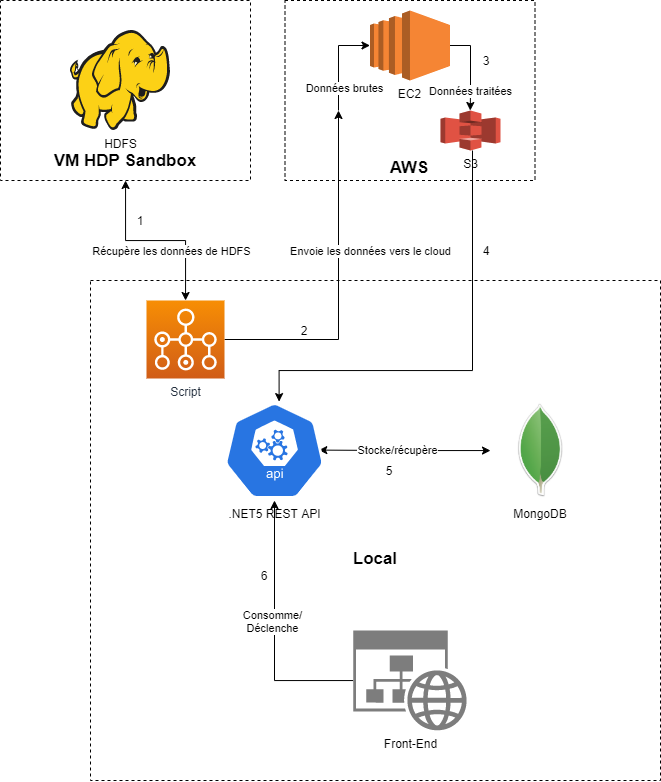
**Dossier d'architecture technique**

Ce document a pour but d’expliquer l’architecture technique utilisée pour JobPredictor. Conformément aux spécifications, les composants cloud sont hébergés sur AWS us-east-1.

# **Architecture générale**



1. Les données brutes sont récupérées de HDFS et rapatriées en local
2. Les données brutes sont envoyées vers l’instance EC2
3. Les données traitées sont stockées durablement sur S3
4. L’API récupère les données traitées
5. L’API stocke les résultats d’analyses dans MongoDB
6. Le Front-End consomme l’API pour récupérer les résultats d’analyse

# **HDFS et transfert vers EC2**

Les données d’entrées sont à stocker sur la VM HortonWorks SandBox HDP 3.0 sous /user/$clouderaUser/rawdata/

Ces données doivent être présentés sous forme d’un fichier JSON avec la structure suivante :

* Id
* description
* gender

Pour envoyer ces données vers le cloud on lance le script ScpTransfer.sh

Afin d’assurer que les données ne soit envoyées qu’une seule fois, le script va tout d’abord se connecter en SSH sur la VM et déplacer l’intégralité du dossier rawdata vers un dossier sous /user/$clouderaUser/data/$(date +%Y)/$(date +%m)/$(date +%d)/$hourstamp

L’export est donc daté et ciblé sur les données qui n’avaient encore pas été exportées.

Le script va ensuite récupérer les données de hdfs vers le file system de la VM via un hdfs dfs get vers ./data/ResumeAnalyzer$timestamp

Ces données sont ensuite transférées via SCP vers la machine hôte vers un dossier temporaire.

Le script se connecte ensuite en SSH sur l’instance EC2 avec sa clé ssh privée pour créer le dossier de réception /home/ec2-user/data/$timestamp

Le transfert des données vers ce dossier se fait par SCP

L’exécution du script peut-être lancé par une tâche CRON afin d’automatiser le transfert de donnée

# **EC2**

L’instance EC2 utilise une image t2.micro afin d’alléger les coûts. Le système d’exploitation est Amazon Linux 2.

Il faut dans un premier temps installer Python et pip sur cette instance.

Ensuite, il y aura besoin de plusieurs dépendances installées, via pip, sur notre instance EC2 :

* Pandas
* SKLearn
* Boto3
* Nltk
* Numpy

Sur notre instance EC2, nous aurons :

* Un dossier data avec
  + Le dossier arrivant de HDFS contenant le ou les fichiers JSON pour lesquels il faut effectuer une prédiction. Ce dossier est nommé sous la forme : **yyyy-MM-dd\_hh-mm-ss**
  + Le fichier label.csv contenant les associations entre l’ID des lignes et le résultats du métier que nous sommes censés avoir.

Nous prenons donc les fichiers JSON 1 par 1, avant de prétraiter la description du poste pour chacune des lignes, et enfin prédire le métier à partir de notre modèle déjà entraîné.

En moyenne, il faut moins de 10mn pour effectuer 5 000 lignes de prédictions.

Une fois la prédiction effectuée, on enregistre dans un fichier CSV les résultats. Le fichier sera nommé “predict\_NomAncienFichier\_NomDossier.csv”. Les données seront représentées sous ce format :

* Id
* Text (texte issu du pré-traitement)
* Category (issu de label.csv)
* Category\_Predict (issu de notre modèle)

Chaque fichier de prédiction est ensuite envoyé sur un bucket S3.

Une fois le fichier envoyé, le fichier JSON de départ et sa prédiction sont supprimés de EC2.

Quand tous les fichiers du dossier ont été traités, on supprime le dossier, et on passe au dossier suivant s’il en existe un.

# **S3**

Les données de prédictions générées par le modèle sont déposées sur un bucket S3. Ce bucket est chiffré avec le chiffrement par défaut d’Amazon, les données y sont stockées durablement.

# **MongoDB**

La base de données choisie est une base orienté document tournant sur MongoDB.

La base de données contient quatre collections :

* La collection Category :

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| Number | int | id de la catégorie |
| Label | string | Label de la catégorie |

* La collection AnalysisResult, qui stocke les résultats d’analyses envoyées par l’API

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| OriginalId | string | Identifiant du CV |
| Category | string | Vraie catégorie |
| CategoryPredict | string | Catégorie prédite par le modèle |

* La collection UpdateTime

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| LastUpdated | DateTime | Date de la dernière mise à jour des données |

* La collection WordCount

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| CountByWord | Dictionary<string,int> | Dictionnaire répertoriant le nombre d’occurence de chaque mot dans le texte lemmatisé |
| Category | string | Catégorie à analyser |

Les données contenues dans la collection Category sont des données de référence qui sont à déposer manuellement en base.

# **API**

L’API est une API REST ASP .NET Core ciblant le framework .NET 5. Elle possède un projet de test associé.

L’API est utilisée pour :

* Récupérer les données stockées sur S3
* Analyser les données
* Stocker le résultat des analyses en base MongoDB
* Servir les données au front-end

L’api comporte 4 endpoints :

/update GET : déclenche la mise à jour des données. L’api récupère la date de dernière mise à jour et télécharge les fichiers qui ont été ajoutés dans le bucket S3 depuis cette date. Chaque ligne du fichier de prédiction constitue une entité AnalysisResult qu’on va stocker en base. Au passage on va mettre à jour les registres WordCount, un par catégorie, avec les nombres d'occurrences de chaque mot dans le texte lemmatisé. Enfin, la date de dernière modification du fichier le plus récent récupéré devient la dernière date de mise à jour en base Mongo.

/analysis GET : Récupération des entités résultats d’analyses.

/wordcount GET : Récupération des entités wordcount en base

/lastupdate GET : Récupération de la dernière date de mise à jour

# **Sécurité**

Tous les transferts de données sont sécurisés, les transferts se font soit en SCP soit en HTTPS. Le stockage durable est quant à lui chiffré via le chiffrement par défaut géré par S3. Pour accéder aux données et faire fonctionner l’API, il est nécessaire de renseigner ses credentials sous ~/.aws dans le fichier credentials.