



BDaaS

---

# Big Data as a Service

---

Réalisé par :

Dehbi Yassine

Doufare Anas

Azizi Mounia

Taleb Ayyoub

2021-2022

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Travaux réalisés : Python Flask</b>	<b>2</b>
2.1	Interface Web . . . . .	2
2.2	Déploiement de l'architecture : Python + Shell Script . . . . .	2
2.3	Configuration . . . . .	2
2.4	Return . . . . .	4
<b>3</b>	<b>Difficultés rencontrées</b>	<b>4</b>
<b>4</b>	<b>Participation du groupe</b>	<b>4</b>
<b>5</b>	<b>Conclusion</b>	<b>4</b>

# 1 Introduction

Ce projet consiste à implémenter une plateforme de Big Data as a Service (BDaaS). L'utilisateur peut accéder au service via une interface web et lancer le calcul sur une infrastructure Spark qui est dynamique (qu'on loue chez AWS).

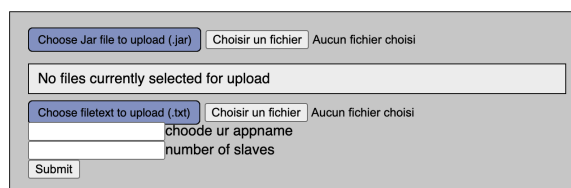
## 2 Travaux réalisés : Python Flask

### 2.1 Interface Web

Le portail permet de réaliser 4 opérations:

- Chargement de données: L'utilisateur devra pouvoir charger des données vers notre infrastructure Spark, plus précisément vers HDFS.
- Uploader un programme (un jar) qui pourra être utilisé par la suite pour lancer un job
- Choisir le nom de l'application à exécuter
- Télécharger le fichier output contenant les résultats
- Choisir le nombre de slaves à exécuter l'application

#### BDaaS



The screenshot shows a web form with the following elements:

- A button labeled "Choose Jar file to upload (.jar)" next to the text "Choisir un fichier" and "Aucun fichier choisi".
- A text box containing "No files currently selected for upload".
- A button labeled "Choose file/text to upload (.txt)" next to the text "Choisir un fichier" and "Aucun fichier choisi".
- An input field labeled "choode ur appname" (likely a typo for "choose your appname").
- An input field labeled "number of slaves".
- A "Submit" button at the bottom.

Figure 1: Interface

### 2.2 Déploiement de l'architecture : Python + Shell Script

Le déploiement de l'architecture est automatisé sur cluster de VM AWS. Pour ceci, on a utilisé la bibliothèque d'AWS boto3 qui permet de créer des instances, les configurer.... On réalise donc la configuration des VMs pour Spark et HDFS en local. Puis on les clone sur les VMs du cluster. À l'aide de scp, et paramiko (python lib pour envoyer des commandes ssh)

### 2.3 Configuration

On a créé une class Python `Mapp(jarfile, datasamples, appname, nslaves)`

Méthodes:

- `createKeyPairFile()` : créer une clé et la stocker dans un fichier.
- `createSecurityGroup()` : crée un groupe de sécurité dans AWS avec la permission des règles entrantes sur ports 80(HTTP), 22(SSH),9000
- `createInstances(self.nbslaves)` : création des instances
- `getIps()` : récupérer les adresses ips des machines
- `executeCommand()` : execute une commande ou un script via des connexions ssh
- `setslaves()` : pour le fichiers de conf slaves
- `edit_ssh_conf()` : configurer les connexions ssh entre les VMs
- `lunch()` : lance les script d'installation de spark, hadoop et java(`executeCommand()`).+ modification des fichiers de configuration(`core-site.xml`, `hdfs-site.xml` slaves `hadoop-env.sh` `spark-env.sh` `.bashrc`)et envoie au VM via scp  
Deux méthodes ont été utilisé :
  - `sed` (Shell command) pour modifier les valeurs des variables.
  - `hardcode` a xml file
- `confmaster.py` : contient une méthode `config(ipmaster, ipslaves, ipdest, jarfile, datasamples)` qui envoie les fichier de conf vers les instances. (**cette boucle est parallélisée**)
- `lancer.sh` script qui permet de formater le namenode lancer hdfs, executer l'appli
- `cleanup()`: scripts du prof pour arrêter les instances

## 2.4 Return

Un fichier d'archive compressé avec gzip

1. Récupération du résultats de hdfs au local(hdfs dfs -get)
2. Compression et envoie par le master vers la machine qui héberge l'interface
3. Envoie du fichier à l'utilisateur

## 3 Difficultés rencontrées

- Erreur de configuration réseau des security group (port 9000)
- Obligation de détruire les instances à chaque test pour éviter de payer amazon
- Le temps que ça prend l'instanciation des instances(on a parallélisée les confi sur les machines quand même)

## 4 Participation du groupe

Tout le groupe à participé sur chaque étape et partie.

## 5 Conclusion

Ce fut un projet très enrichissant. Ce projet nous a permis de manipuler des "objets" aws à travers l'API boto3. Nous avons pu mettre en place un service qui nous permet d'instancier un architecture scalable qui permet d'exécuter une application spark avec hdfs