



PROJET 8 : DÉPLOYEZ UN MODÈLE DANS LE CLOUD

LAURENT CAGNIART - OPENCLASSROOMS

PROBLÉMATIQUE



Le Digital au service de l'agriculture



Préserver la biodiversité des fruits en permettant des traitements spécifiques pour chaque espèce de fruits en développant des robots cueilleurs intelligents.



Mettre à disposition une application mobile grand public permettant de **reconnaitre un fruit à partir de sa photo** et d'en obtenir des informations.

- Développer dans un environnement Big Data une première chaîne de traitement des données qui comprendra le preprocessing et une étape de réduction de dimension.
- Prendre en compte le passage à l'échelle des calculs du fait que le volume de données va augmenter très rapidement après la livraison du projet en :
 - Ecrivant des scripts en PySpark.
 - Déployant le modèle sur le Cloud



kaggle

(un fruit ou légume
par image)



■ **Training dataset :**

67,692 images

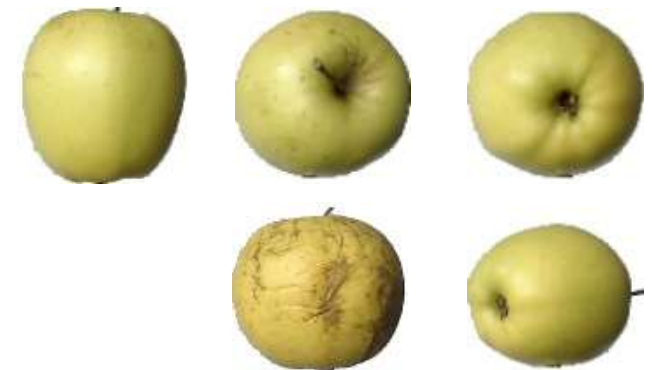


■ **Test dataset :**

22,688 images

- Variétés : 131 fruits ou légumes
- Taille image (pixels) : 100 x 100
- Format : jpg
- Photos prises sous différents angles et états (ex. « rotten »)

- Le jeu de données est un ensemble d'images de fruits et de labels associés (ex. Apple golden)



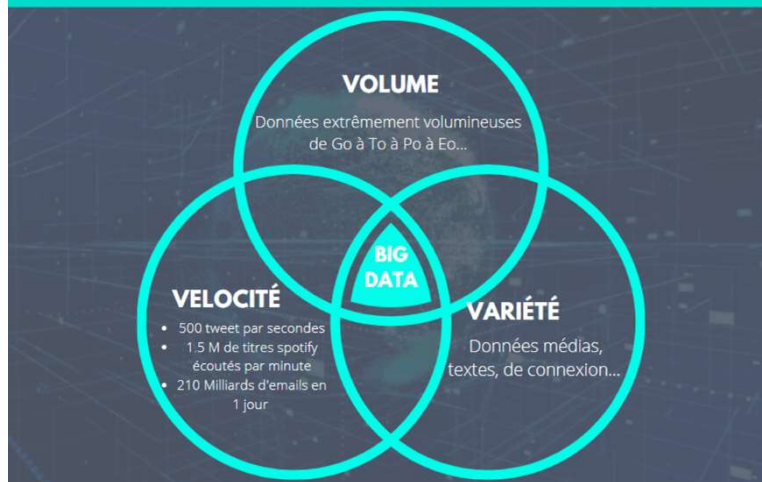
DONNÉES DE BASE

BIG DATA



Nous considérerons que l'on fait du **big data** à partir du moment où la quantité de données excède la faculté d'une machine à les stocker et les analyser en un temps acceptable

LES 3 V DU BIG DATA



- Le **Volume** des données générées nécessite de repenser la manière dont elles sont stockées.
- La **Vélocité** à laquelle nous parvenons ces données implique de mettre en place des solutions de traitement en temps réel qui ne paralysent pas le reste de l'application.
- Les données se présentent sous une grande **Variété** de formats : ces données peuvent être structurées (documents JSON), semi-structurées (fichiers de log) ou non structurées (textes, images). L'ingestion, l'analyse et la rétention de ces données prendront des formes différentes selon leur nature, ce qui implique de mettre en place des outils appropriés.



Distribuer de manière intelligente et efficace le stockage et les traitements de ces données



BIG DATA



La solution à notre problème consiste à paralléliser les calculs sur plusieurs machines différentes.

Système distribué

un ensemble de programmes informatiques qui utilisent des ressources informatiques sur plusieurs nœuds de calcul distincts pour atteindre un objectif commun et partagé

Les systèmes distribués :

- Sont extensibles
- Offrent une évolutivité.
- Et des facilités de partage des ressources



Stockage

Utiliser des systèmes de fichiers => Hadoop HDFS

Traitement des données

MapReduce ou Spark

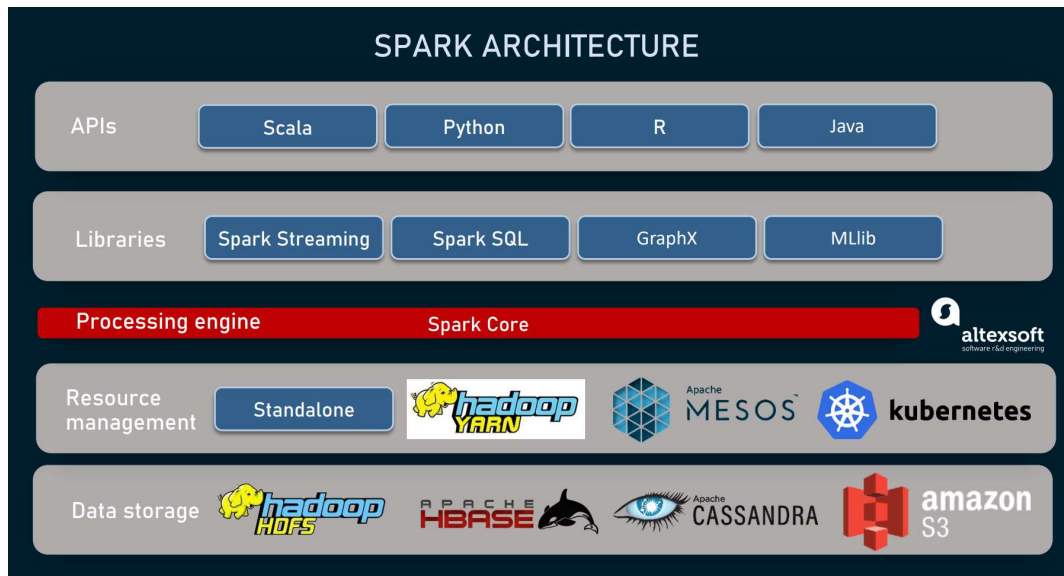
- Stratégie de distribution ? Agrégation des résultats ? Prise en compte des pannes lors de l'exécution du calcul ? Optimisation des coûts ?



est un framework open source de calcul distribué in-memory pour le traitement et l'analyse de données massives par le biais de Clusters (cluster computing).



Spark exécute des programmes jusqu'à 100 fois plus vite que Hadoop MapReduce en mémoire, ou 10 fois plus vite sur disque.



APACHE Spark + python = PySpark

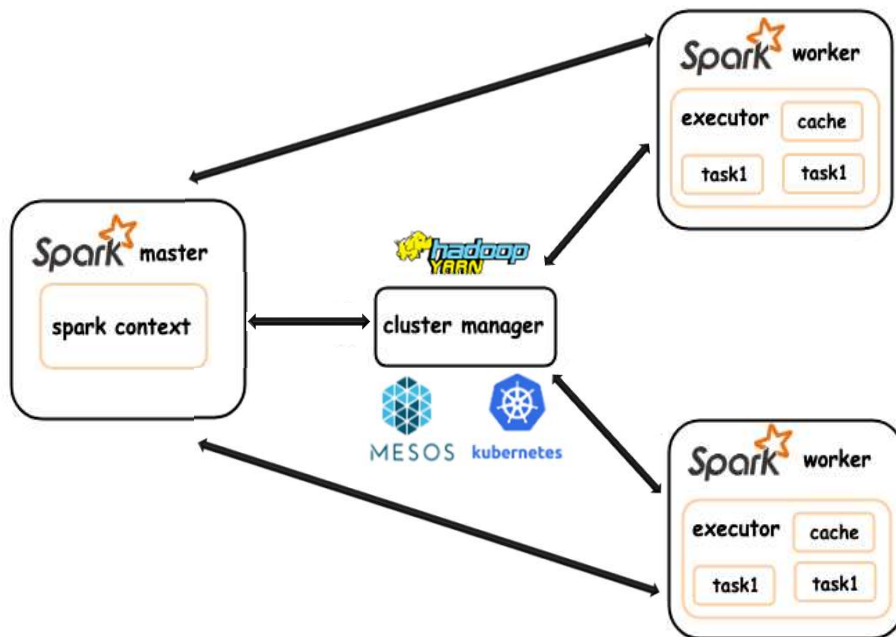


PySpark est une interface pour Apache Spark en Python.

En mode local, le parallélisme des calculs se fera en fonction du nombre de cœurs disponibles sur la machine

SPARK

Architecture Spark : maître-esclave

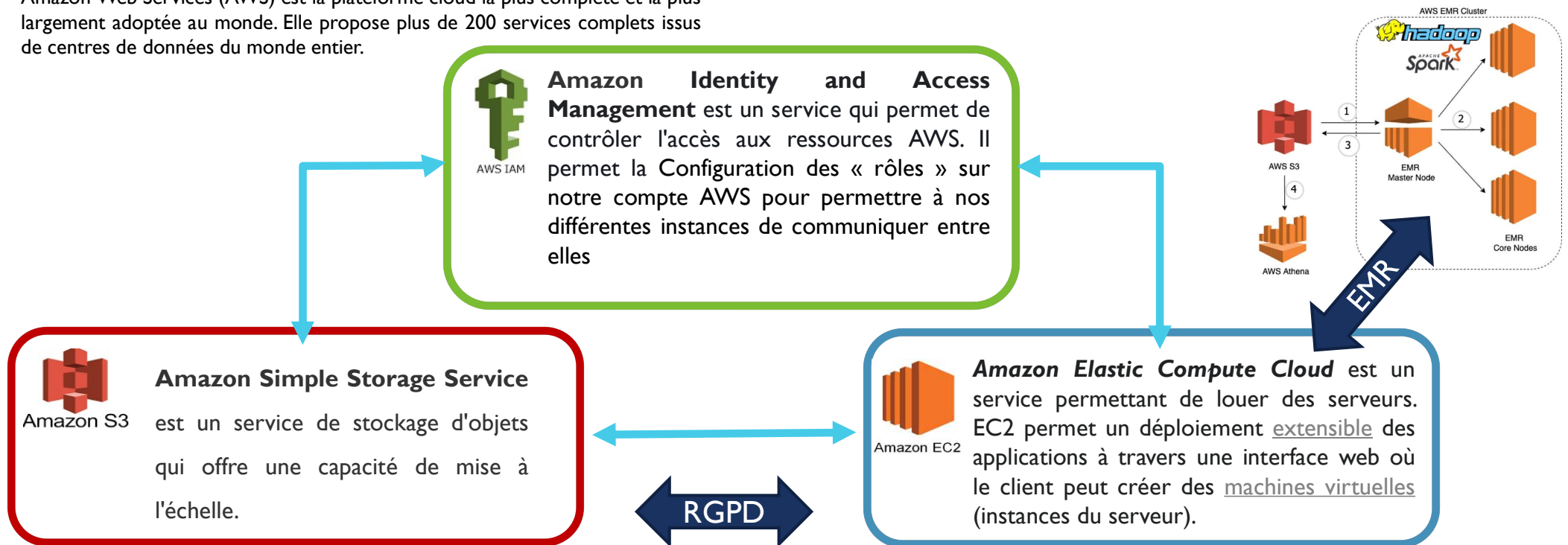


- **Driver** : L'objet Sparkcontext ou Sparksession instancié et configuré va créer une machine virtuelle (Java) responsable de la mise en œuvre de l'application et de la distribution et l'agrégation des calculs (RDD)
- **Cluster manager** : il permet de :
 - Gérer l'allocation de ressources.
 - Gérer la division des programmes.
 - Gérer l'exécution du programme.
- **Worker** : reçoit une tâche du pilote et instancie un «executor» chargé d'exécuter les différentes tâches de calcul

SPARK



Amazon Web Services (AWS) est la plateforme cloud la plus complète et la plus largement adoptée au monde. Elle propose plus de 200 services complets issus de centres de données du monde entier.



Instances et bucket S3 créés sur le territoire européen (France et Irlande)

AWS



IAM

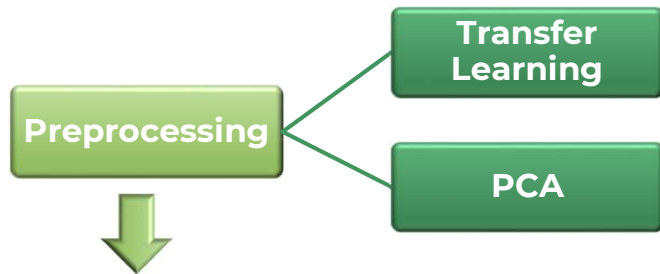
- Définition des rôles par défaut et utilisation du compte Root pour les besoins limités sur les premières étapes du projet

S3

- Création d'un bucket «oc-lolo-p8-data».
- Stockage des fichiers : images, résultats (parquet, json), notebook

EMR

- Création des instances Linux m5.xlarge avec choix des applications de base (Hadoop, spark, Jupyter, Tensorflow)
- Bootstrapping (pré installation des librairies et packages nécessaires : pandas, fsspec, etc...)
- Connection à EMR via tunnel SSH (Putty)
- Exécution du code sur Jupyter Notebook

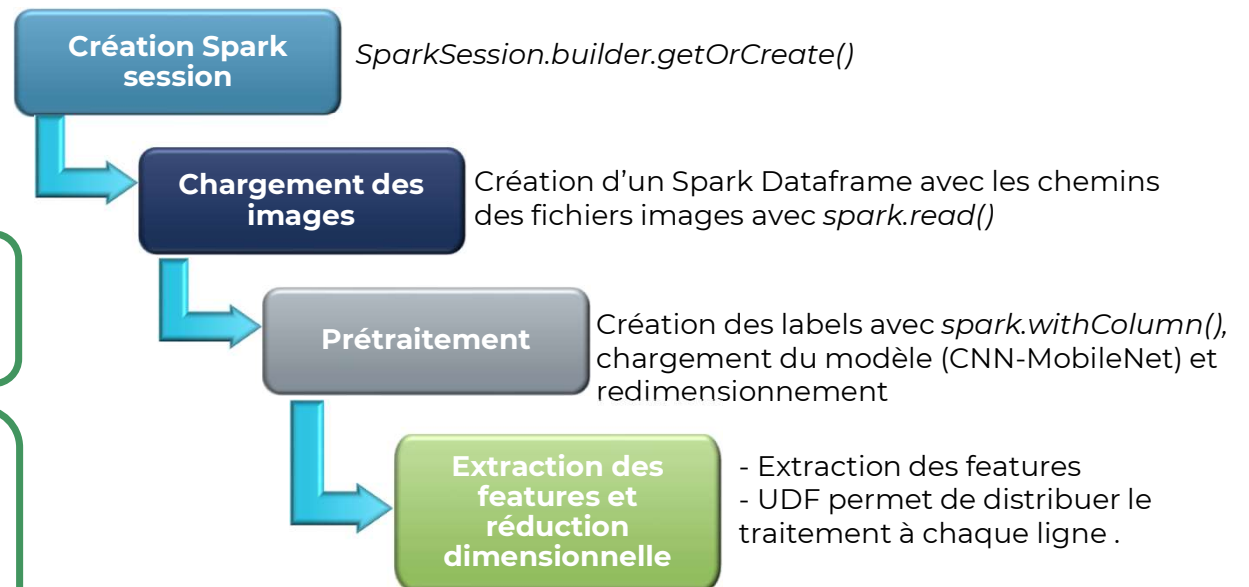


Extraire les features afin de préparer les images pour la modélisation

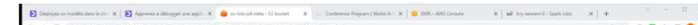
Le **Transfer Learning** permet d'utiliser les connaissances acquises par un réseau de neurones lors de la résolution d'un problème afin d'en résoudre un autre plus ou moins similaire

Analyse des Composantes Principales : Il s'agit de résumer l'information contenue dans une large base de données en un certain nombre de variables synthétiques appelées : Composantes Principales.

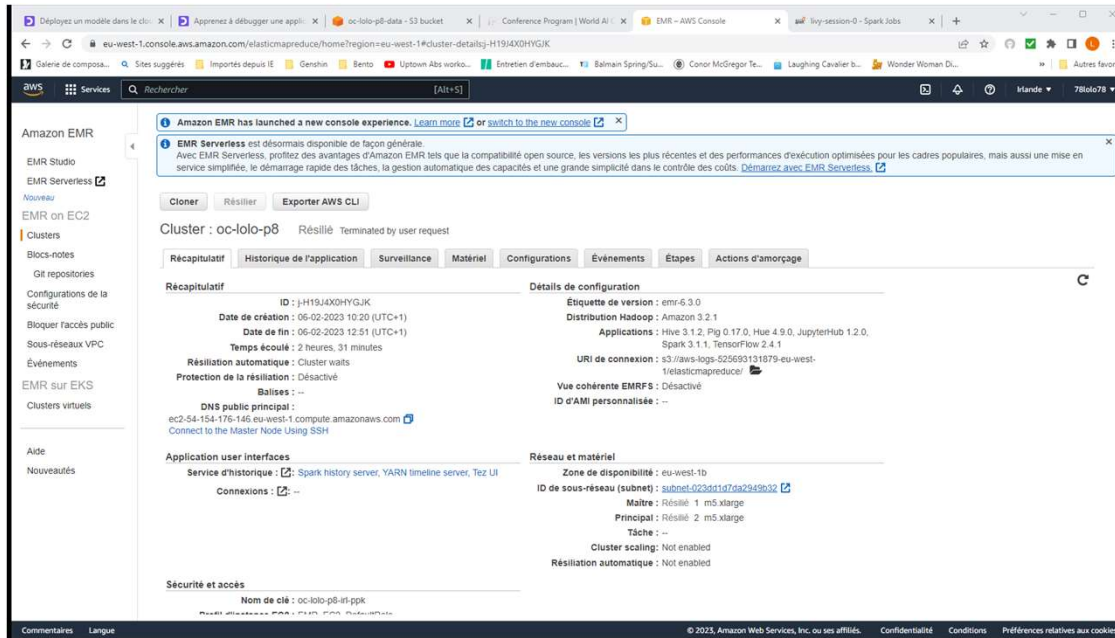
Pipeline Transfer Learning- MobileNet - PCA



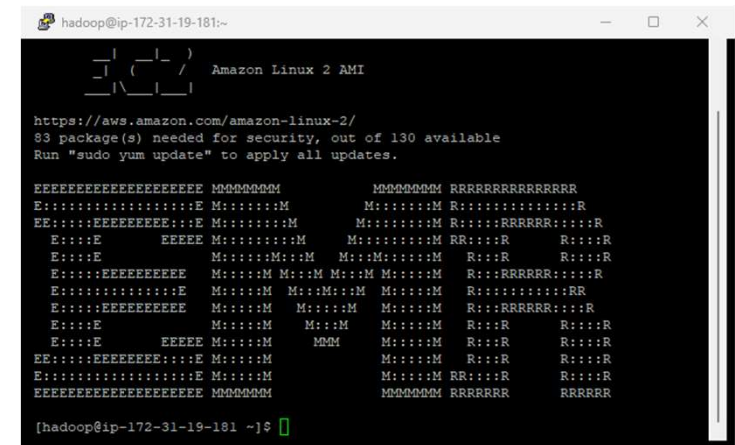
SCRIPTS PYPARK



- EMR console - cluster



- Connexion instance EMR via tunnel SSH



■ Lancement de la session Spark

Entrée [1]: # L'exécution de cette cellule démarre l'application Spark

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1675675973171_0002	pyspark	idle	Link	Link	✓

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...)

SparkSession available as 'spark'.

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...)

[Affichage des informations sur la session en cours et liens vers Spark UI](#)

4.10.2 Installation des packages

Les packages nécessaires ont été installés via l'étape de bootstrap à l'instanciation du serveur.

4.10.3 Import des librairies



■ Résultat des scripts PySpark

df_pca_transformed.write.mode('overwrite').json(PATH_RESULT)

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...)

Entrée [30]: df_result = spark.read.json(PATH_RESULT + '/*.json')

df_result.show()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...)

features	label	path	pcaFeatures	scaledFeatures
[1, 1.1428744792...	Watermelon	s3://oc-lolo-p8-d...	[1, [-14.34861957...	[1, 1.3603997785...
[1, 1.3194596767...	Watermelon	s3://oc-lolo-p8-d...	[1, [-14.62411263...	[1, 1.6876204888...
[1, 0.0401640757...	Pineapple Mini	s3://oc-lolo-p8-d...	[1, [-13.51770216...	[1, -0.682974675...
[1, 0.6248898506...	Watermelon	s3://oc-lolo-p8-d...	[1, [-10.35058049...	[1, 0.4005497993...
[1, 0.0359662100...	Watermelon	s3://oc-lolo-p8-d...	[1, [-10.87490829...	[1, -0.690753518...
[1, 0.0, 5.16979...	Pineapple Mini	s3://oc-lolo-p8-d...	[1, [-13.26912349...	[1, -0.757400604...
[1, 0.2590608094...	Raspberry	s3://oc-lolo-p8-d...	[1, [-2.170062305...	[1, -0.277348542...
[1, 0.3444652250...	Raspberry	s3://oc-lolo-p8-d...	[1, [-2.352304663...	[1, -0.119090273...
[1, 0.0, 0.83160...	Cauliflower	s3://oc-lolo-p8-d...	[1, [-4.983006192...	[1, -0.757400604...
[1, 0.0, 4.60650...	Pineapple	s3://oc-lolo-p8-d...	[1, [-14.52643875...	[1, -0.757400604...
[1, 0.0, 3.97352...	Pineapple	s3://oc-lolo-p8-d...	[1, [-13.36912718...	[1, -0.757400604...
[1, 0.0, 1.96264...	Cauliflower	s3://oc-lolo-p8-d...	[1, [-4.639903334...	[1, -0.757400604...
[1, 0.0052550141...	Cauliflower	s3://oc-lolo-p8-d...	[1, [-8.004192543...	[1, -0.747662815...
[1, 0.0, 1.33845...	Cauliflower	s3://oc-lolo-p8-d...	[1, [-9.854618014...	[1, -0.757400604...
[1, 0.0, 0.01042...	Apple Golden 1	s3://oc-lolo-p8-d...	[1, [2.4128159437...	[1, -0.757400604...
[1, 0.0140607263...	Apple Golden 1	s3://oc-lolo-p8-d...	[1, [-0.141399746...	[1, -0.731345414...
[1, 0.0, 4.06660...	Pineapple Mini	s3://oc-lolo-p8-d...	[1, [-14.74331821...	[1, -0.757400604...
[1, 0.0, 4.12750...	Pineapple Mini	s3://oc-lolo-p8-d...	[1, [-15.27769709...	[1, -0.757400604...
[1, 2.1734669208...	Cucumber Ripe	s3://oc-lolo-p8-d...	[1, [-17.44495754...	[1, 3.2701362433...
[1, 0.0, 0.05584...	Apple Golden 1	s3://oc-lolo-p8-d...	[1, [4.9605768489...	[1, -0.757400604...

only showing top 20 rows

AWS SCREENSHOTS – JUPYTER NOTEBOOK

Timeline



Spark Jobs (7)
 User: livy
 Total Uptime: 1.2 h
 Scheduling Mode: FIFO
 Completed Jobs: 12
 Active Jobs: 1

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
12 (29)	Job group for statement 25 first at PCASuccessful	2023/02/06 10:45:51	2.5 min	0/2	152/152 (0 skipped)

Completed Jobs (12)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11 (25)	Job group for statement 25 showing at NativeMethodAccessorImpl.java:0	2023/02/06 10:48:14	23 s	1/1 (1 skipped)	1/1 (1 skipped)
10 (25)	Job group for statement 25 showing at NativeMethodAccessorImpl.java:0	2023/02/06 10:33:02	13 min	1/1	766/766
9 (25)	Job group for statement 25 first at NativeMethodAccessorImpl.java:113	2023/02/06 10:35:02	0.1 s	1/1 (2 skipped)	1/1 (2 skipped)
8 (25)	Job group for statement 25 first at NativeMethodAccessorImpl.java:113	2023/02/06 10:32:06	2.8 min	1/1 (1 skipped)	16/16 (109 skipped)
7 (25)	Job group for statement 25 first at NativeMethodAccessorImpl.java:113	2023/02/06 10:18:47	13 min	1/1	766/766

Spark Jobs (7)
 User: livy
 Total Uptime: 2.1 h
 Scheduling Mode: FIFO
 Completed Jobs: 21

Completed Jobs (21)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
20 (29)	Job group for statement 29 join at NativeMethodAccessorImpl.java:0	2023/02/06 11:42:00	3.0 min	1/1 (1 skipped)	24/24 (709 skipped)
19 (29)	Job group for statement 29 join at NativeMethodAccessorImpl.java:0	2023/02/06 11:28:04	13 min	1/1	766/766
18 (29)	Job group for statement 29 showing at NativeMethodAccessorImpl.java:0	2023/02/06 11:28:16	26 s	1/1 (1 skipped)	1/1 (1 skipped)
17 (29)	Job group for statement 29 showing at NativeMethodAccessorImpl.java:0	2023/02/06 11:14:56	13 min	1/1	766/766
16 (29)	Job group for statement 29 showing at NativeMethodAccessorImpl.java:0	2023/02/06 11:03:43	2.9 min	2/2 (1 skipped)	16/16 (709 skipped)
15 (29)	Job group for statement 29 first at RowMetrics.java:442	2023/02/06 11:05:06	17 s	1/1 (1 skipped)	1/1 (1 skipped)
14 (29)	Job group for statement 29 showing at NativeMethodAccessorImpl.java:0	2023/02/06 11:03:35	2.8 min	2/2 (1 skipped)	16/16 (709 skipped)
13 (29)	Job group for statement 29 first at RowMetrics.java:442	2023/02/06 11:02:14	22 s	1/1 (1 skipped)	1/1 (1 skipped)
12 (29)	Job group for statement 29 first at PCASuccessful	2023/02/06 10:45:51	13 min	2/2	762/762
11 (25)	Job group for statement 25 showing at NativeMethodAccessorImpl.java:0	2023/02/06 10:48:14	23 s	1/1 (1 skipped)	1/1 (1 skipped)

Active job

List of all jobs

AWS SCREENSHOTS – SPARK JOBS

AXES D'AMÉLIORATION

- Infrastructure AWS et scripts Pyspark créés permettant une future mise à l'échelle (revoir le scaling des clusters selon les besoins pour optimisation des coûts)

- Améliorer la sécurisation de l'ensemble (IAM, bucket S3), minimiser la surface d'attaque

- Affiner la PCA et finaliser la modélisation (classification des images)



MERCI DE VOTRE ATTENTION

LAURENT CAGNIART

MARQUEUR ICON BY ICONS8