

PROJET 9

**Réalisez un traitement
dans un environnement
Big Data sur le Cloud**

Marion Dedieu - 03/2024

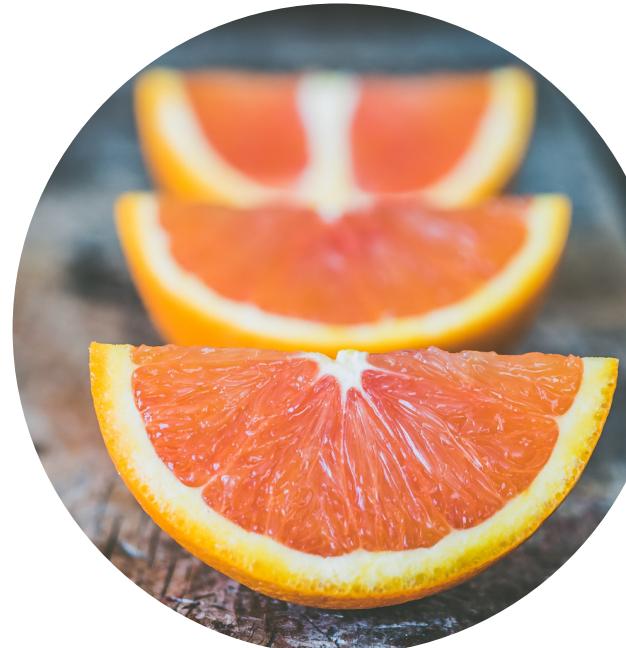


Sommaire

01. Problématique & Jeu de données
02. Environnement Big Data
03. Chaîne de traitement
04. Exécution du script PySpark
05. Conclusion



Problématique & Jeu de données



Problématique

🚜 Contexte :

- Jeune start-up AgriTech "Fruits!"
- Objectif : Révolutionner la récolte des fruits tout en préservant la biodiversité.

🍇 Volonté de l'entreprise :

- Développer des robots cueilleurs intelligents adaptés à chaque espèce de fruit.

📱 Objectif : Application Mobile

- Fonctionnalités :
 - Prendre en photo un fruit.
 - Obtenir des informations détaillées.
- Importance :
 - Sensibiliser à la biodiversité des fruits.
 - Début du moteur de classification d'images.



Mission

- Mettre en place un environnement Big Data sur le Cloud
- Réaliser une chaîne de traitement des données :
 - Preprocessing
 - Réduction de dimension

Contraintes

- Augmentation importante du volume des données à prendre en compte
- Respecter les contraintes du RGPD (serveurs situés en Europe)
- Gestion des coûts liés à l'architecture Big Data



Jeu de données

- Dataset Kaggle : **Fruits-360**
- 90 380 images contenues dans 131 dossiers de fruits et légumes
- Taille du jeu de test utilisé ici :
22 688 images
- Taille des images : 100x100 pixels
Format : jpg



Environnement Big Data



Qu'est-ce que le Big Data ?

Définition

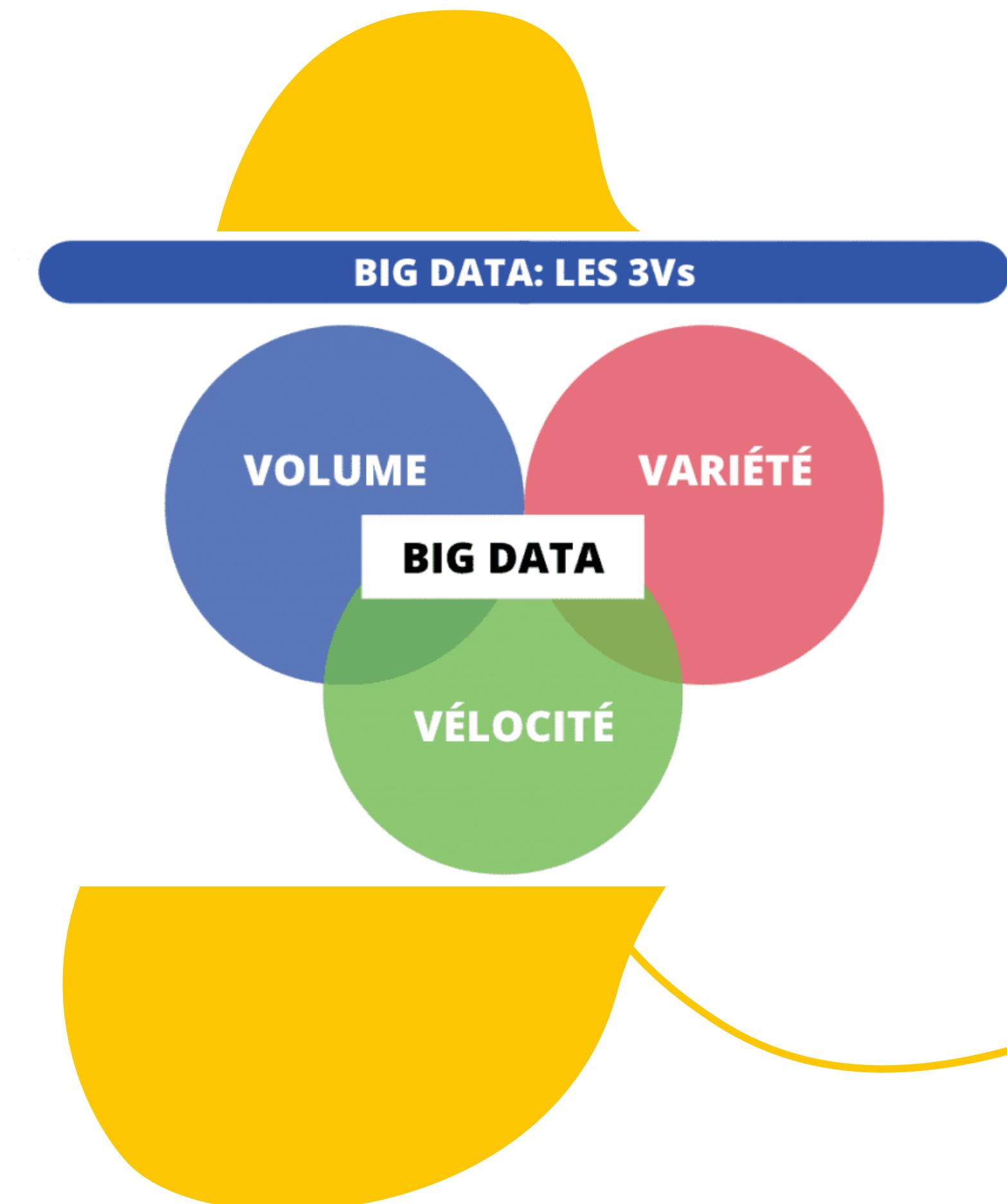
Le Big Data se réfère à de **vastes ensembles de données** complexes qui **dépassent les capacités** de traitement des outils traditionnels de gestion des données.

Les 3 V

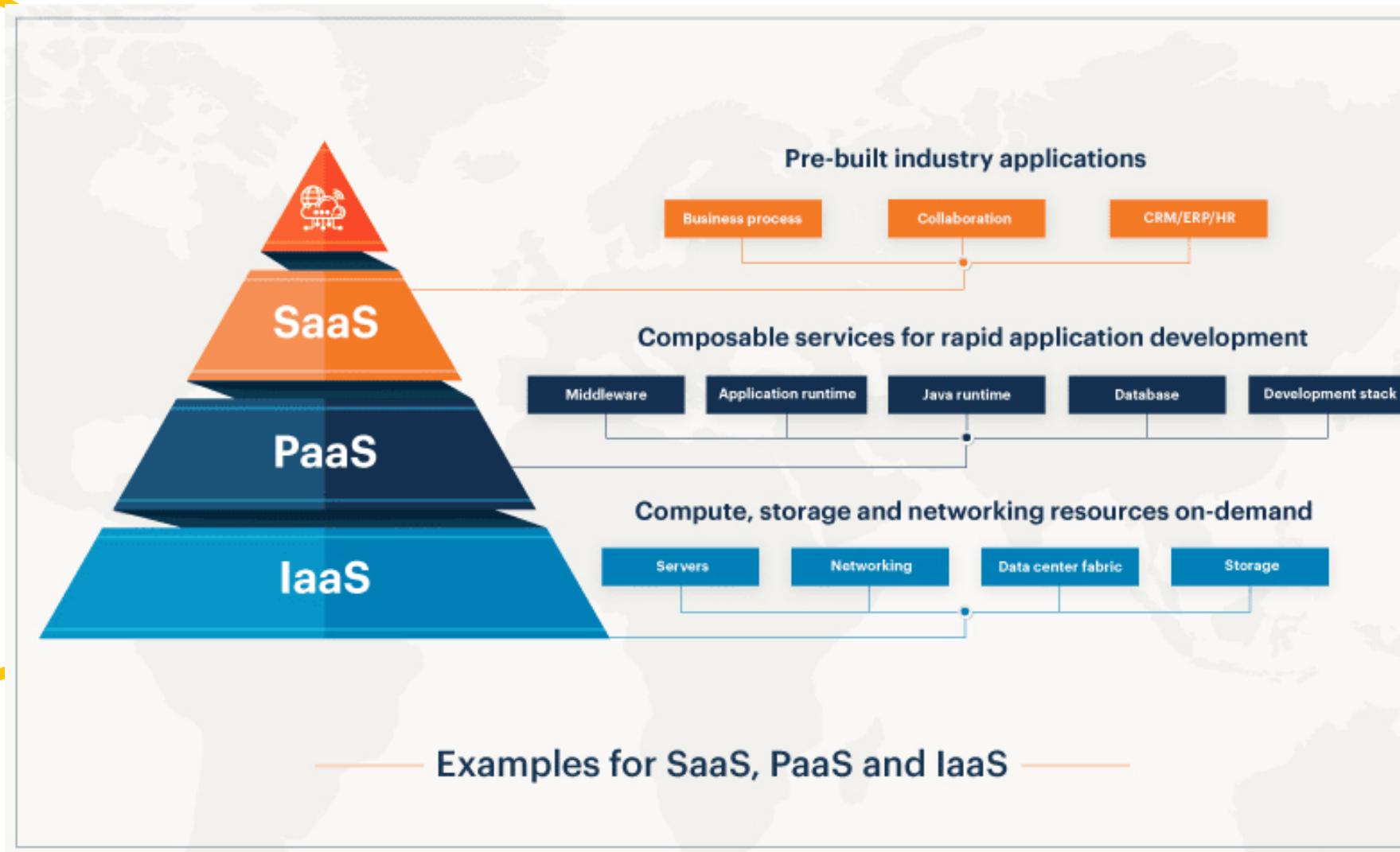
- **Volume** : Le Big Data implique des quantités massives de données qui peuvent être structurées, semi-structurées ou non structurées.
- **Variété** : Les données peuvent provenir de diverses sources telles que les réseaux sociaux, les capteurs, les transactions, les textes, les images, etc.
- **Vélocité** : Les données sont générées à un rythme élevé et doivent être traitées rapidement pour obtenir des informations en temps réel.

Pourquoi le Big Data ?

- Volume des données en augmentation
- Besoin d'utiliser des calculs distribués



Choix du prestataire



Services AWS utilisés

01.

Instance EC2

Serveur virtuel flexible et évolutif dans le cloud, permettant aux utilisateurs de déployer et de gérer des applications et des charges de travail variées.

- Format t2.micro
- Région : eu-west-3c (Paris)

02.

Stockage S3

Service de stockage objet dans le cloud offrant une scalabilité illimitée et une disponibilité élevée pour stocker et récupérer des données de manière sécurisée.

03.

Gestion accès IAM

Service de gestion des identités et des accès permettant de sécuriser et de contrôler l'accès aux ressources AWS en définissant des politiques et des permissions pour les utilisateurs, les groupes et les rôles.

04.

Cluster EMR (Elastic Map Reduce)

Service de gestion de clusters Hadoop/Spark dans le cloud, permettant le traitement et l'analyse de grands ensembles de données de manière efficace et évolutive grâce à des instances EC2 configurées automatiquement.

- emr-7.0.0 + Applications Hadoop 3.3.6, JupyterHub 1.5.0, Spark 3.5.0
- Instances m5.xlarge

Instance EC2

EC2 > Instances > i-0f2fc7053c6b16034

Résumé de l'instance pour i-0f2fc7053c6b16034 Informations Mis à jour il y a 1 minute

ID d'instance i-0f2fc7053c6b16034	Adresse IPv4 publique 13.39.5.6 [adresse ouverte]	Adresses IPv4 privées 172.31.41.161
Adresse IPv6 -	État de l'instance En cours d'exécution	DNS IPv4 public ec2-13-39-5-6.eu-west-3.compute.amazonaws.com [adresse ouverte]
Type de nom d'hôte Nom de l'adresse IP: ip-172-31-41-161.eu-west-3.compute.internal	Nom DNS de l'IP privé (IPv4 uniquement) ip-172-31-41-161.eu-west-3.compute.internal	Adresses IP élastiques 13.39.5.6 [IP publique]
Réponse à un nom DNS de ressource privée IPv4 (A)	Type d'instance t2.micro	Recherche d'AWS Compute Optimizer Inscrivez-vous à AWS Compute Optimizer pour obtenir des recommandations.
Adresse IP attribuée automatiquement -	ID de VPC vpc-0e74f6c4e70f0fa41	En savoir plus
Rôle IAM -	ID de sous-réseau subnet-04b7c5c0a4ecf043d	Nom du groupe Auto Scaling -
IMDSv2 Required		

Stockage S3

p8marionflore [Info](#)

Objets [Propriétés](#) [Autorisations](#) [Métriques](#) [Gestion](#) [Points d'accès](#)

Objets (5) [Info](#)

[Ouvrir](#) [Actions ▾](#)

[Créer un dossier](#) [Charger](#)

Les objets sont les entités fondamentales stockées dans Amazon S3. Vous pouvez utiliser l'[inventaire Amazon S3](#) pour obtenir une liste de tous les objets de votre compartiment. Pour que d'autres personnes puissent accéder à vos objets, vous devez leur accorder explicitement des autorisations. [En savoir plus](#)

Rechercher des objets en fonction du

<input type="checkbox"/>	Nom	Type	Dernière modification	Taille	Classe de stockage
<input type="checkbox"/>	bootstrap-emr.sh	sh	11 Mar 2024 02:36:17 PM CET	384.0 o	Standard
<input type="checkbox"/>	Final_results.csv	csv	11 Mar 2024 07:26:12 PM CET	2.3 Mo	Standard
<input type="checkbox"/>	jupyter/	Dossier	-	-	-
<input type="checkbox"/>	Results/	Dossier	-	-	-
<input type="checkbox"/>	Test/	Dossier	-	-	-

Test/

Objets [Propriétés](#)

Objets (130) [Info](#)

[Ouvrir](#)

[Créer un dossier](#) [Charger](#)

Les objets sont les entités fondamentales stockées dans Amazon S3. Vous pouvez utiliser l'[inventaire Amazon S3](#) pour obtenir une liste de tous les objets de votre compartiment. Pour que d'autres personnes puissent accéder à vos objets, vous devez leur accorder explicitement des autorisations

Rechercher des objets en fonction du

<input type="checkbox"/>	Nom	Type	Dernière modification
<input type="checkbox"/>	Apple Braeburn/	Dossier	-
<input type="checkbox"/>	Apple Crimson Snow/	Dossier	-
<input type="checkbox"/>	Apple Golden 1/	Dossier	-
<input type="checkbox"/>	Apple Golden 2/	Dossier	-
<input type="checkbox"/>	Apple Golden 3/	Dossier	-
<input type="checkbox"/>	Apple Granny Smith/	Dossier	-

Cluster EMR

Amazon EMR > EMR sur EC2: Clusters > Projet8Fruits

Projet8Fruits

Mise à jour il y a moins d'une minute [C](#) [Résilier](#) [Cloner dans AWS CLI](#) [Cloner](#)

Récapitulatif			
Informations sur le cluster	Applications	Gestion des clusters	Statut et heure
ID de cluster j-TUZM35TNTV6W	Version d'Amazon EMR emr-7.0.0	Destination des journaux dans Amazon S3 aws-logs-360166801129-eu-west-3/elasticmapreduce	Statut En attente
Configuration de cluster Groupes d'instances	Applications installées Hadoop 3.3.6, JupyterHub 1.5.0, Spark 3.5.0	Interfaces utilisateur d'application persistantes Serveur d'historique Spark Serveur de chronologie YARN	Heure de création 11 mars 2024 14:37 (UTC+01:00)
Capacité 1 primaire(s) 1 unité(s) principale(s) 2 tâche(s)		DNS public du nœud primaire ec2-35-180-71-215.eu-west-3.compute.amazonaws.com	Temps écoulé 1 jour, 18 heures
		Connexion au nœud primaire à l'aide de SSH Connexion au nœud primaire à l'aide de SSM	

Interfaces utilisateur d'application sur le nœud primaire

Celles-ci nécessitent l'activation du tunneling SSH.

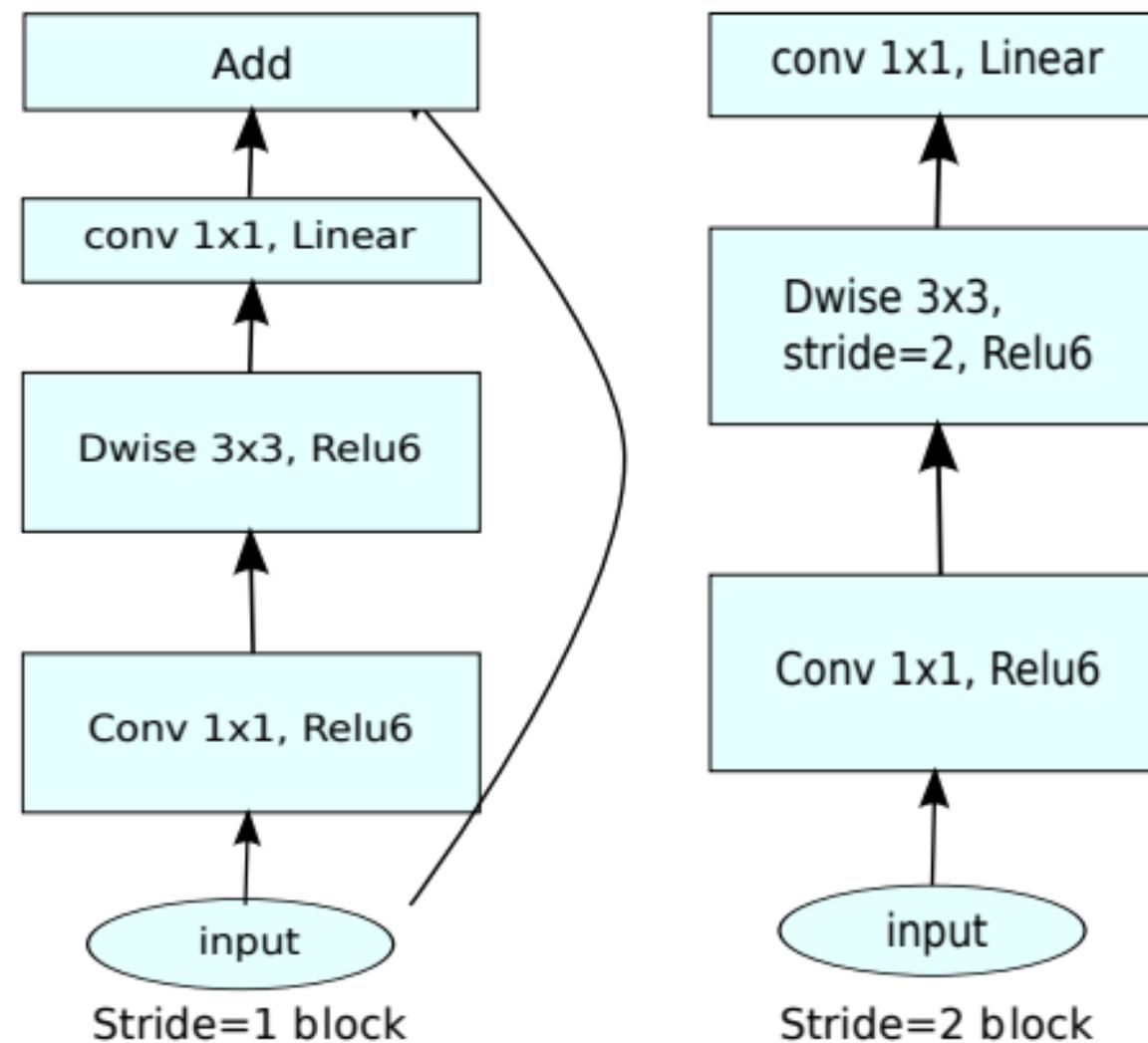
[Activer une connexion SSH](#)

Application	URL de l'interface utilisateur
Gestionnaire de ressources	http://ec2-35-180-71-215.eu-west-3.compute.amazonaws.com:8088/
JupyterHub	https://ec2-35-180-71-215.eu-west-3.compute.amazonaws.com:9443/
Nom du nœud HDFS	http://ec2-35-180-71-215.eu-west-3.compute.amazonaws.com:9870/
Serveur d'historique Spark	http://ec2-35-180-71-215.eu-west-3.compute.amazonaws.com:18080/

Chaîne de traitement



Etape 1 : Test du notebook en local



(d) Mobilenet V2

Exécution du notebook laissé par l'alternant sur Jupyter :

- Définition des chemins en local
- Crédit d'une session Spark
- Crédit du modèle MobileNetV2
- Extraction des features des images
- Enregistrement du résultat au format parquet

Etape 2 : Création de l'architecture sur le Cloud

- Lancement d'une instance EC2 :
 - Type t2.micro
 - Région eu-west-3C (Paris) : respect du RGPD
 - Génération d'une paire de clé au format .pem
- Crédit du stockage sur S3 :
 - Bucket "p8marionflore"
 - Upload du dossier "Test" du dataset



Etape 3 : Instanciation du cluster EMR

```
$ bootstrap-emr.sh
1  #!/bin/bash
2  sudo python3 -m pip install -U setuptools
3  sudo python3 -m pip install -U pip
4  sudo python3 -m pip install wheel
5  sudo python3 -m pip install pillow
6  sudo python3 -m pip install pandas==1.4.1
7  sudo python3 -m pip install pyarrow
8  sudo python3 -m pip install boto3
9  sudo python3 -m pip install s3fs
10 sudo python3 -m pip install fsspec
11 sudo python3 -m pip install tensorflow==2.14.0
```

Paramétrage :

- Version : emr-7.0.0
- Applications : Hadoop 3.3.6, JupyterHub 1.5.0, Spark 3.5.0
- Configuration au format JSON : enregistrement sur S3
- Action d'amorçage : script Shell uploadé sur S3
- Sécurité : paire de clé liée à l'instance EC2
- Création de rôles IAM pour les accès à EC2, EMR et S3

Statut

 En attente

```
{} config.json > {} 0 > {} properties > s3.persistence.bucket
1  [{"classification":"jupyter-s3-conf","properties":{"s3.persistence.bucket":"p8marionflore","s3.persistence.enabled":"true"}}]
```

Etape 4 : Connexion à JupyterHub



Règles entrantes (9)

Version IP	Type	Protocole	Plage de ports	Source	Description
-	Tous les UDP	UDP	0 - 65535	sg-0bbc9d1fca00d86a...	-
-	Tous les UDP	UDP	0 - 65535	sg-0b15d55ccaa79cb1...	-
-	Tous les TCP	TCP	0 - 65535	sg-0bbc9d1fca00d86a...	-
IPv6	SSH	TCP	22	::/0	-
-	TCP personnalisé	TCP	8443	pl-4aa04523 ↗	-
-	Tous les TCP	TCP	0 - 65535	sg-0b15d55ccaa79cb1...	-
-	Tous les ICMP - IPv4	ICMP	Tous	sg-0bbc9d1fca00d86a...	-
-	Tous les ICMP - IPv4	ICMP	Tous	sg-0b15d55ccaa79cb1...	-
IPv4	SSH	TCP	22	0.0.0.0/0	-

Proxy SOCKS

Serveur: localhost

Port: 5555

Mot de passe requis:

Vos identifiants peuvent être envoyés sans être chiffrés.

- Crédit du tunnel SSH à l'instance EC2 (Maître)
 - Modification des groupes de sécurité pour autoriser les connexions entrantes
- Connexion au noeud primaire via SSH + configuration d'un Proxy

```
● ○ ● marion — hadoop@ip-172-31-10-14:~ — ssh -i ~/Documents/Scolarité et apprentissage/Master Dat..
(base) marion@macbook-air-de-marion-2 ~ % ssh -i "/Users/marion/Documents/Scolarité et apprentissage/
Master Data Scientist/Projet 9 - Réalisez un traitement dans un environnement Big Data sur le Cloud/c
letest.pem" -D 5555 hadoop@ec2-35-180-71-215.eu-west-3.compute.amazonaws.com
```

```
'          #_
~\_\_ #####_          Amazon Linux 2023
~~ \_\#\#\#\`_
~~   \#\#\#
~~   \#/ _-- https://aws.amazon.com/linux/amazon-linux-2023
~~   V~' '---'
~~   / '
~~.._-_-_
~~ /_/
~~/_m/'
```

Last login: Wed Mar 13 09:16:19 2024

```
EEEEEEEEEEEEEEEEE MBBBBBBB      MBBBBBBB RRRRRRRRRRRRRR
E:::::::EEEEE:::E M:::::M      M:::::M R:::::::::::R
EE:::::EEEEE:::E M:::::M      M:::::M R:::::RRRRR:::::R
  E:::E   EEEEE M:::::M      M:::::M RR:::R R:::::R
  E:::E   M:::::M:::M M:::M:::M R:::R R:::::R
  E:::::EEEEE:::E M:::::M M:::::M M:::::M R:::::RRRRR:::::R
  E:::::::::::E M:::::M M:::::M M:::::M R:::::::::::RR
  E:::::EEEEE:::E M:::::M M:::::M M:::::M R:::::RRRRR:::::R
  E:::E   M:::::M M:::::M M:::::M R:::::R R:::::R
  E:::E   EEEEE M:::::M MBBB M:::::M R:::::R R:::::R
EE:::::EEEEE:::E M:::::M M:::::M R:::::R R:::::R
E:::::::::::E M:::::M M:::::M RR:::R R:::::R
EEEEEEEEEEEEEEEEE MBBBBBBB      MBBBBBBB RRRRRRRRR
```

[hadoop@ip-172-31-10-14 ~]\$ █

Exécution du script PySpark



Démarrage de la session Spark, Import des librairies

Import des librairies

Entrée [1]:

```
import pandas as pd
import numpy as np
import io
import os
import tensorflow as tf
from PIL import Image
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras import Model
from pyspark.sql.functions import col, pandas_udf, PandasUDFType, element_at, split
from pyspark.ml.linalg import Vectors, VectorUDT
from pyspark.ml.feature import PCA
from pyspark.sql import functions as F
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	User	Current session?
1	application_1710164625461_0002	pyspark	idle	Link	Link	None	✓

```
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%'),...
```

Chargement des images

Entrée [3]:

```
PATH = 's3://p8marionflore'
PATH_Data = PATH+'/Test'
PATH_Results = PATH+'/Results'
print('PATH:      '+\
      PATH+'\nPATH_Data:  '+\
      PATH_Data+'\nPATH_Results: '+PATH_Results)
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...
PATH:      s3://p8marionflore
PATH_Data:  s3://p8marionflore/Test
PATH_Results: s3://p8marionflore/Results
```

Chargement des données

Entrée [4]:

```
images = spark.read.format("binaryFile") \
    .option("pathGlobFilter", "*.jpg") \
    .option("recursiveFileLookup", "true") \
    .load(PATH_Data)
```

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

Entrée [5]:

```
images.show(5)
```

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

path	modificationTime	length	content
s3://p8marionflor...	2024-03-08 20:02:01	7353	[FF D8 FF E0 00 1...]
s3://p8marionflor...	2024-03-08 20:00:29	7350	[FF D8 FF E0 00 1...]
s3://p8marionflor...	2024-03-08 19:59:56	7349	[FF D8 FF E0 00 1...]
s3://p8marionflor...	2024-03-08 20:02:49	7348	[FF D8 FF E0 00 1...]

Ajout du label des images et sélection de colonnes

```
Entrée [7]: images = images.withColumn('label', element_at(split(images['path'], '/'), -2))
print(images.printSchema())
print(images.select('path','label').show(5, False))
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'), ...
root
 |-- path: string (nullable = true)
 |-- modificationTime: timestamp (nullable = true)
 |-- length: long (nullable = true)
 |-- content: binary (nullable = true)
 |-- label: string (nullable = true)

None
+-----+-----+
|path |label |
+-----+
|s3://p8marionflore/Test/Watermelon/r_106_100.jpg|Watermelon|
|s3://p8marionflore/Test/Watermelon/r_109_100.jpg|Watermelon|
|s3://p8marionflore/Test/Watermelon/r_108_100.jpg|Watermelon|
|s3://p8marionflore/Test/Watermelon/r_107_100.jpg|Watermelon|
|s3://p8marionflore/Test/Watermelon/r_95_100.jpg|Watermelon|
+-----+
only showing top 5 rows
```

```
None
```

Création du modèle MobileNetV2 (Transfer Learning)

```
Entrée [8]: model = MobileNetV2(weights='imagenet',
                                include_top=True,
                                input_shape=(224, 224, 3))

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weight
s_tf_dim_ordering_tf_kernels_1.0_224.h5
14536120/14536120 [=====] - 1s 0us/step
```

```
Entrée [9]: new_model = Model(inputs=model.input,
                           outputs=model.layers[-2].output)

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

```
Entrée [10]: broadcast_weights = sc.broadcast(new_model.get_weights())

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

```
Entrée [11]: new_model.summary()

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

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	[]
Conv1 (Conv2D)	(None, 112, 112, 32)	864	['input_1[0][0]']
bn_Conv1 (BatchNormalizati on)	(None, 112, 112, 32)	128	['Conv1[0][0]']
Conv1_relu (ReLU)	(None, 112, 112, 32)	0	['bn_Conv1[0][0]']

Extraction des features des images et enregistrement au format .parquet

```
Entrée [14]: features_df = images.repartition(24).select(col("path"),
                                                     col("label"),
                                                     featurize_udf("content").alias("features")
                                                   )
```

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

```
Entrée [15]: print(PATH_Results)
```

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

```
s3://p8marionflore/Results
```

```
Entrée [17]: features_df.write.mode("overwrite").parquet(PATH_Results)
```

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

Ouverture fichier parquet, vectorisation, PCA, export CSV

```
Entrée [19]: # Ouverture du fichier au format parquet
df = pd.read_parquet(PATH_Results, engine='pyarrow')
```

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

```
Entrée [23]: # Vectorisation de la colonne features
to_vector = F.udf(lambda x: Vectors.dense(x), VectorUDT())
sparkDF = features_df.select('path', 'label','features', to_vector("features").alias("features_vec"))
```

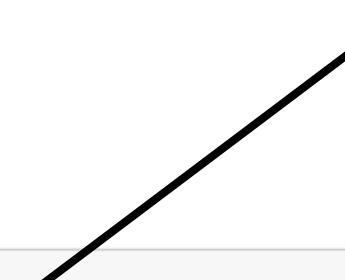
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...
```

```
Entrée [24]: # Aperçu du fichier après transformation
sparkDF.show(5)
```

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

	path	label	features	features_vec
s3://p8marionflor...	Pineapple Mini	[0.0020795355, 4....	[0.00207953550852...	
s3://p8marionflor...	Watermelon	[0.7753587, 0.245....	[0.77535867691040...	
s3://p8marionflor...	Watermelon	[0.64753145, 0.34....	[0.64753144979476...	
s3://p8marionflor...	Watermelon	[0.05196944, 0.11....	[0.05196943879127...	
s3://p8marionflor...	Raspberry	[0.010020801, 0.3....	[0.01002080086618...	

only showing top 5 rows



	path	label	pca_Features
s3://p8marionflor...	Watermelon	[-1.9308853812663...	
s3://p8marionflor...	Watermelon	[-2.4230607062742...	
s3://p8marionflor...	Pineapple Mini	[-6.7759678423393...	
s3://p8marionflor...	Watermelon	[-4.4761678492822...	
s3://p8marionflor...	Raspberry	[0.38324143415740...	

only showing top 5 rows

```
Entrée [25]: # Utilisation du PCA (k=2)
pcaSparkEstimator = PCA(inputCol="features_vec", outputCol="pca_Features", k=2)
pca = pcaSparkEstimator.fit(sparkDF)
pca_matrix=pca.transform(sparkDF)
```

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

Export au format csv

```
Entrée [29]: pca_matrix_final.toPandas().to_csv('s3://p8marionflore/Final_results.csv')
```

Suivi des jobs Spark



Conclusion



Avantages de la solution

- Stockage quasiment illimité sur S3
- Script PySpark prêt pour des ensembles de données plus grands
- Instances modifiables selon les besoins

Précautions à considérer

- Coordination des versions logicielles pour assurer la compatibilité et la stabilité
- Surveillance des coûts liés à la solution AWS



Merci !