

Présentation du projet 8 : Déployez un modèle dans le cloud



Préparé par :
MAGHLAZI Hanane

Projet 8 : Déployez un modèle dans le cloud

- **Contexte**
- **Mission**
- **Contraintes**
- **Présentation des données**
- **Le Big Data**
- **Calcul distribué**
- **Spark**
- **Pyspark**
- **Solutions Cloud**
- **AWS**
- **Choix de la solution de stockage des données : Amazon S3**
- **Choix de la solution technique : EMR**
- **Le service IAM**
- **Traitements des images**
- **Conclusion**
- **Difficultés**
- **Aller plus loin**

Contexte :

- La jeune start-up de l'AgriTech, nommée "**Fruits!**", cherche à proposer des **solutions innovantes** pour la récolte des fruits.
- La **volonté** de l'entreprise est de **préserver** la **biodiversité** des fruits en permettant des traitements spécifiques pour chaque espèce de fruits en développant des **robots cueilleurs intelligents**.
- Elle souhaite dans un **premier** temps se faire connaître en mettant à disposition du grand public une application mobile qui permettrait aux utilisateurs de prendre en **photo** un **fruit** et d'obtenir des **informations** sur ce fruit.
- Pour l'entreprise, cette application permettrait de **sensibiliser** le grand public à la **biodiversité** des fruits et de mettre en place une **première version** du moteur de **classification** des **images** de fruits.
- De plus, le **développement** de l'application mobile permettra de construire une première version de l'architecture **Big Data** nécessaire.



Fruits!

Mission :

- Développer une première chaîne de traitement des données qui comprendra le **preprocessing** et une étape de **réduction de dimension**
- Tenir compte du fait que le volume de données va **augmenter** très rapidement après la livraison de ce projet, ce qui **implique** de:
 - Déployer le traitement des données dans un environnement **Big Data**
 - Développer les scripts en **Pyspark** pour effectuer du **calcul distribué**



Fruits!

Contraintes :

- Développer des scripts en **Pyspark**.
- Utiliser le **cloud** AWS pour profiter d'une architecture **Big Data (EMR, S3)**



Fruits!

Déroulement

- Le projet va être réalisé en **2** temps, dans deux **environnements** différents :

Dans un premier temps développer et exécuter le code **en local**, en travaillant sur un nombre limité d'images à traiter.

Une fois les choix techniques validés, nous déployerons la solution dans un environnement **Big Data** en **mode distribué**.

Présentation des données

Le nombre total d'images : **90483.**

Taille du set de l'entraînement: 67692 images

Taille du set du test: 22688 images

Le nombre de classes: 131 fruits

Taille d'image: 100x100 pixels.



Présentation des données

- Exemple d'un fruit :



0_100.jpg



1_100.jpg



2_100.jpg



3_100.jpg



10_100.jpg



11_100.jpg



r_30_100.jpg



r_31_100.jpg



r_100_100.jpg



r_101_100.jpg



r_102_100.jpg



r_103_100.jpg



r_104_100.jpg



r_105_100.jpg



r_106_100.jpg



r_107_100.jpg



r_108_100.jpg



r_109_100.jpg



r_110_100.jpg



r_111_100.jpg



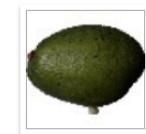
r_112_100.jpg



r_113_100.jpg



r_114_100.jpg



r_115_100.jpg

- Vue 360° de chaque fruit
- Sur fond blanc
- Sous format JPG

Le Big Data

- Le **Big Data** : des données plus variées, arrivant dans des **volumes croissants** et à une **vitesse** plus **élevée**. C'est ce que l'on appelle les trois « **V** ».
- Les **trois « V » du Big Data** :

Volume :

Avec le Big Data, on traite de gros volumes de données non structurées et à faible densité. Il peut s'agir de données de valeur inconnue, comme des flux de données Twitter, des flux de clics sur une page Internet ou une application mobile

Vitesse:

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.

Variété :

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).

En d'autres termes, le Big Data est composé de jeux de données complexes, provenant essentiellement de nouvelles sources. Ces ensembles de données sont si volumineux qu'un logiciel de traitement de données traditionnel ne peut tout simplement pas les gérer.

Calcul distribué :

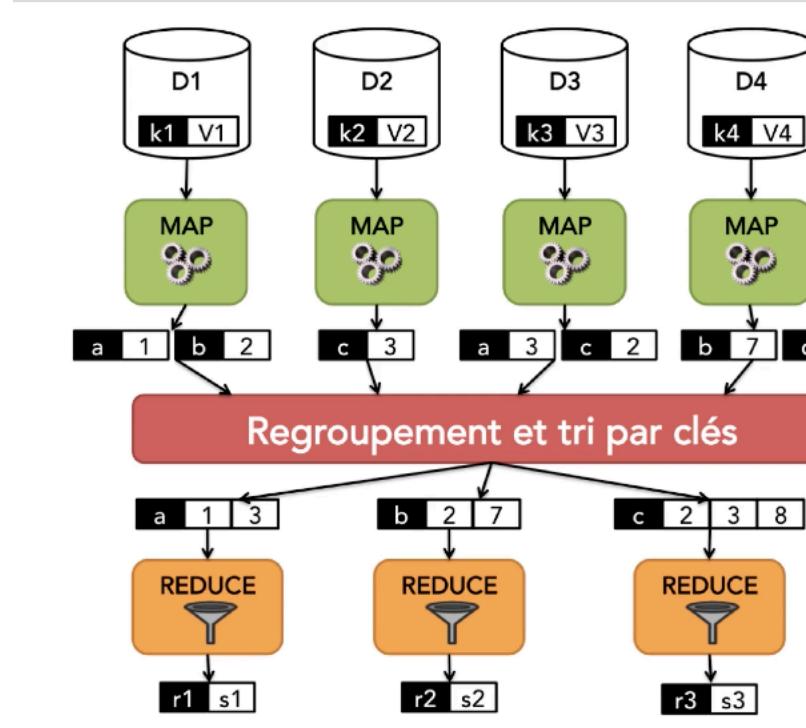
Lors d'un calcul réalisé en **parallèle** : différents *threads* d'exécutions sont exécutés en même temps et partagent une mémoire commune. On passe à l'échelle de manière **verticale**, en augmentant la puissance des processeurs

Dans le calcul **distribué**, les nœuds sur lesquels les calculs sont exécutés sont **distantes, autonomes** et ne partagent pas de ressources ; la communication entre les **nœuds** s'effectue grâce à l'envoi de messages, au sein d'un **cluster**. Le passage à l'échelle s'effectue de manière **horizontale** c'est à dire qu'il suffit d'ajouter des nœuds au cluster pour augmenter sa **capacité** de calcul.

Le modèle **distribué** permet une plus grande **tolérance** aux **pannes** : lorsqu'un nœud du cluster subit une panne, il suffit **d'affecter** la tâche qu'il était en train de traiter à un **autre nœud**, alors que dans le modèle parallèle la machine sur laquelle le calcul est exécuté constitue un point unique de défaillance.

MapReduce

Modèle de programmation qui fournit un cadre pour automatiser le **calcul parallèle sur des données massives**.
Son principe général est que toute **parallélisation** de traitement sur des données massives peut s'effectuer uniquement à l'aide de deux types **d'opérations** : une opération **map** et une opération **reduce**

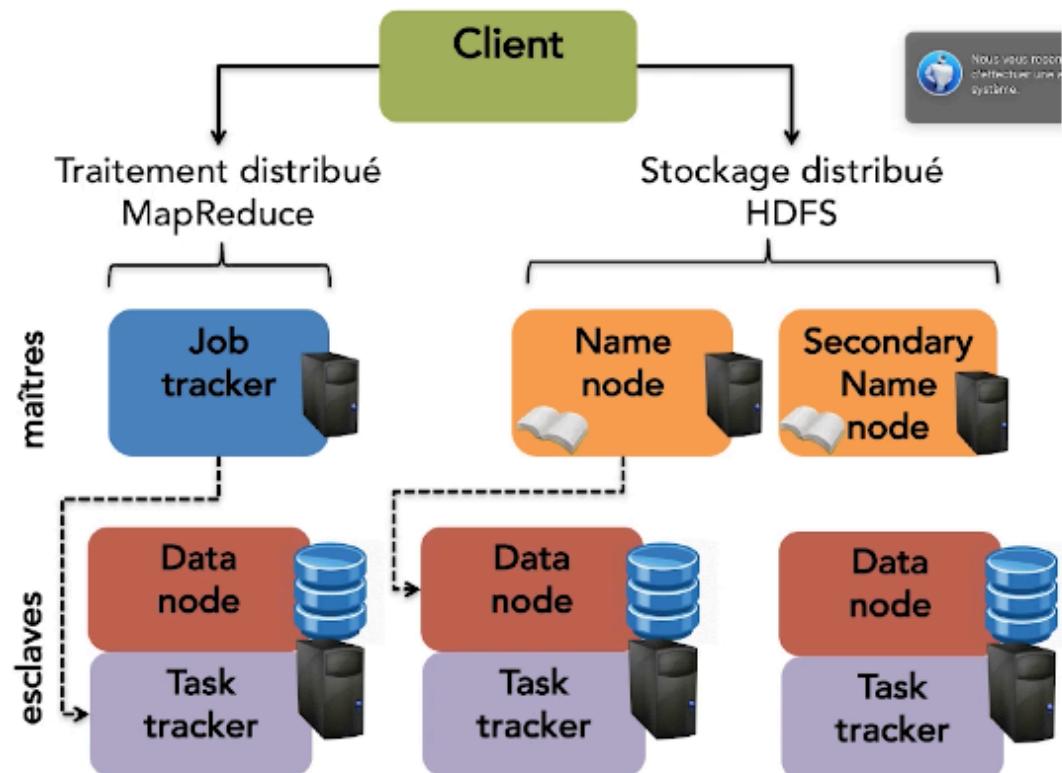
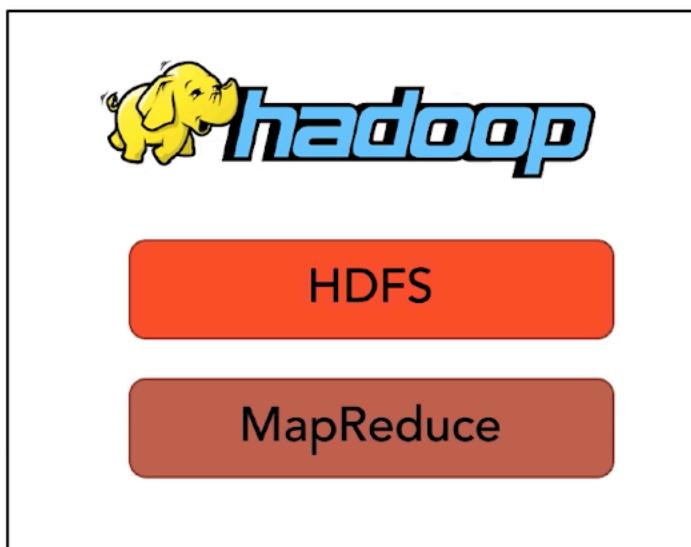


- 4 étapes distinctes :
- ✧ Découper (SPLIT)
 - ✧ Transformer (MAP)
 - ✧ Grouper (SHUFFLE)
 - ✧ Réduire (REDUCE)

Hadoop

Framework Java open source utilisé pour le **stockage et traitement des big data**. Le principe utilisé par hadoop pour cela est la **répartition** de manière **distribuée** des données afin de les traiter en parallèle. Hadoop utilise le modèle de programmation **MapReduce** pour stocker et récupérer plus rapidement les données dans ses nœuds.

Hadoop : le socle technique



Spark

- **Spark** (ou **Apache Spark**) est un **Framework** open source de **calculé** distribué. Ce produit est un cadre applicatif de traitements des **méga données** (*Big data*) pour effectuer des **analyses complexes** à **grande échelle**. Il a été développé en **Scala**

Apache Spark est une alternative à Hadoop MapReduce pour le **calcul distribué** qui vise à **résoudre** ces deux **problèmes** :

- ✓ Après une opération map ou reduce, le résultat doit être écrit sur disque. Ces écritures et lectures sont coûteuses en temps.
- ✓ Le jeu d'expressions composé exclusivement d'opérations map et reduce est très limité et peu expressif. Il est difficile d'exprimer des opérations complexes en n'utilisant que cet ensemble de deux opérations.

La différence fondamentale entre Hadoop MapReduce et Spark est que Spark écrit les données en RAM, et non sur disque. Ceci a plusieurs conséquences importantes sur la rapidité de traitement des calculs (10 fois plus élevés que Hadoop) ainsi que sur l'architecture globale de Spark.

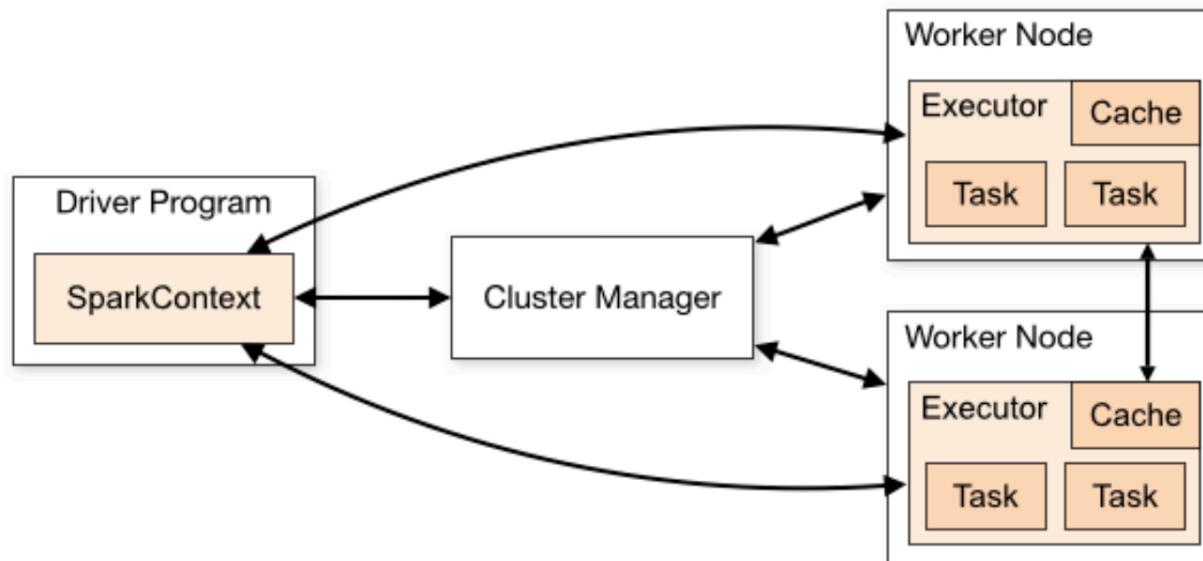


Spark

- **Fonctionnement de Spark :**

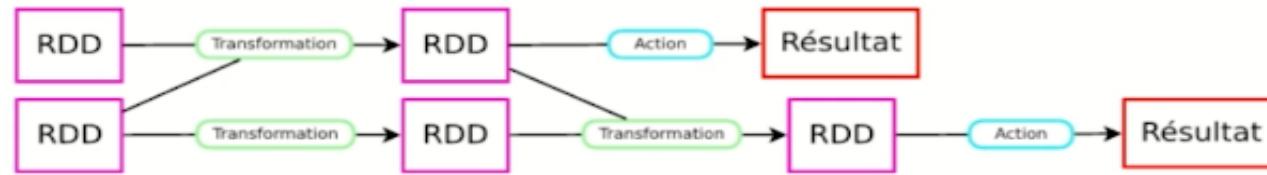
Les applications Spark se composent d'un :

- ✓ **Un ou plusieurs workers** : chaque worker instancie un executor chargé d'exécuter les différentes tâches de calculs.
- ✓ **Un driver** : chargé de répartir les tâches sur les différents executors. C'est le driver qui exécute la méthode main de nos applications.
- ✓ **Un cluster manager** : chargé d'instancier les différents workers.

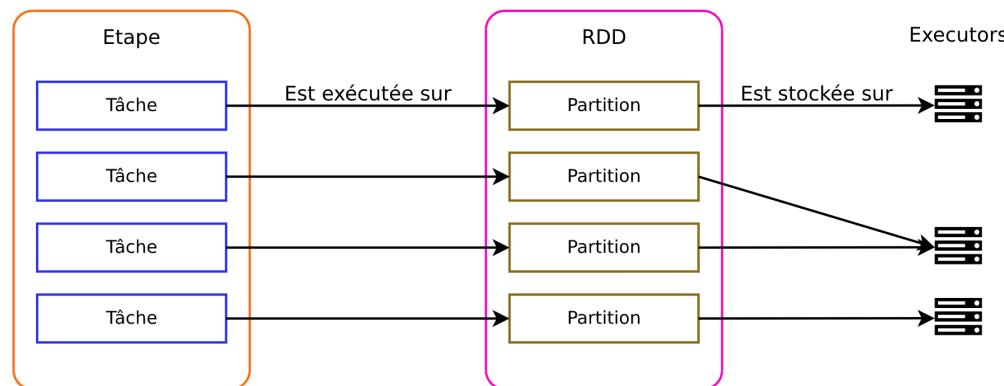


Spark

Calcul distribué:



Cycle de vie d'une application:



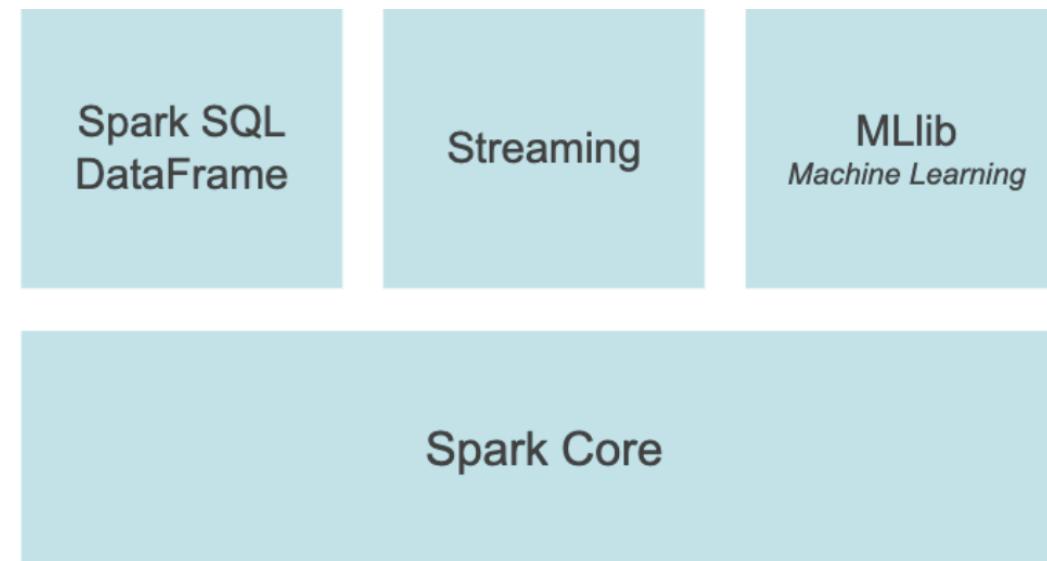
Les données sont découpées en **partitions**. Chaque partition est assignée à un des executors. Le traitement d'une partition représente une **tâche**.

Un ensemble de tâches réalisées en parallèle, une par partition d'un RDD, constitue une **étape** (stage).

Un **job** Spark est composé d'une succession d'étapes et est créé pour chaque action qu'on réalise sur un RDD.

Pyspark

- la librairie **PySpark** propose d'utiliser Spark avec le langage **Python**.
- PySpark fournit également le shell PySpark pour analyser de manière interactive les données dans un environnement distribué. Il prend en charge la plupart des fonctionnalités de Spark telles que **Spark SQL, Data Frame, Streaming, MLlib (Machine Learning) et Spark Core**.

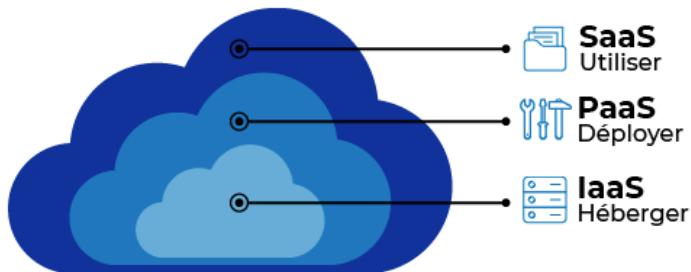


Solutions cloud :

- **Qu'est ce que le cloud ?**

Le cloud est la pratique consistant à utiliser des serveurs informatiques à distance et hébergés sur internet pour stocker, gérer et traiter des données, plutôt qu'un serveur local ou un ordinateur personnel.

- **Les différents types de cloud :**



SaaS (Software as a Service) :

On vous fournit l'accès à un logiciel sous forme de service. Avant, vous deviez installer le logiciel sur votre machine (ex. : Microsoft Office). Aujourd'hui, le logiciel se présente sous la forme d'une application. Vous devez juste vous rendre à une adresse et vous pouvez l'utiliser (ex. : Microsoft Office 365, Google Workspace...).

PaaS (Platform as a Service) :

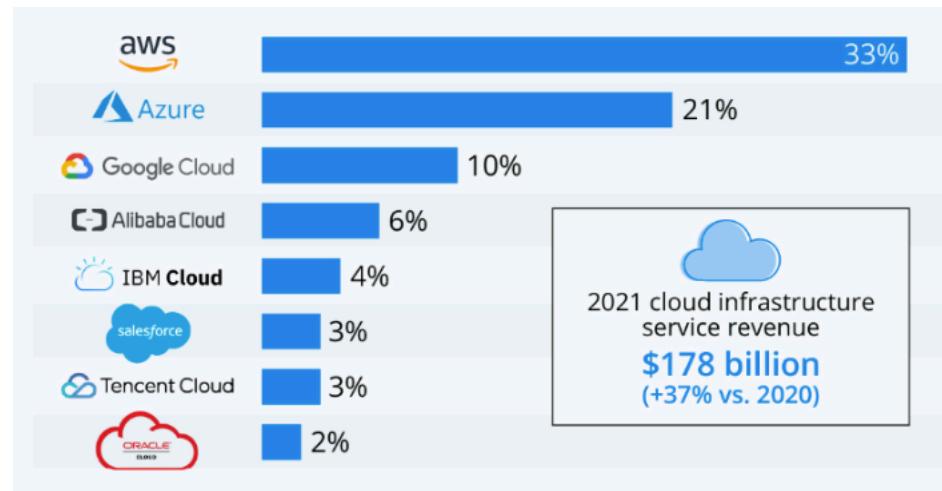
non seulement on vous fournit un accès à l'infrastructure, mais on s'occupe aussi de gérer le nombre de machines nécessaires pour que votre application fonctionne bien quelle que soit la charge de trafic. On vous donne aussi accès à des fonctionnalités comme par exemple des bases de données, des serveurs de cache, des serveurs d'e-mail...

IaaS (Infrastructure as a Service) :

un prestataire vous fournit un accès à tout ou partie de son infrastructure technique, c'est-à-dire à ses serveurs. C'est ce que faisait Amazon à ses tout débuts

AWS

Le graphique suivant, datant de 2021, nous donne une idée de la **part** de **marché** de chaque **fournisseur** :

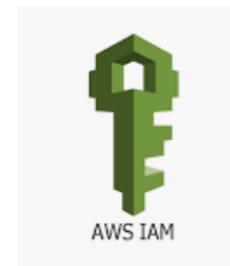
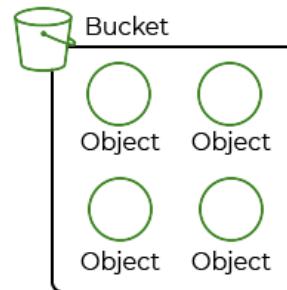


- Fort de ses **33 %** de part de marché, **Amazon** est celui qui domine l'offre
- **AWS** fournit à la fois des services de **PaaS et de IaaS**, mais aussi quelques services en **SaaS**.

AWS

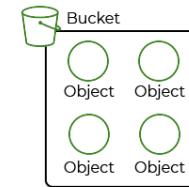
Démarche :

- Configuration du compte AWS
- Installation et configuration de awscli
- Précision de la région eu-west-1
- Choix de la solution de stockage S3
- Choix de la solution technique EMR
- Création d'un utilisateur IAM



Choix de la solution de stockage des données : Amazon S3

- **Amazon Simple Storage Service** (abrégé **S3**) est un service de **stockage** de **données**. Il s'agit d'un moyen de stocker des fichiers sur Internet.
- Amazon S3 propose de stocker des données dans des **buckets**



p8-hanane-maghazi [Info](#)

Objets Propriétés Autorisations Métriques Gestion Points d'accès

Objets (6)

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 dans votre bucket. Vous devez leur accorder explicitement des autorisations. [En savoir plus](#)

Copier l'URI S3 Copier l'URL Télécharger Ouvrir Supprimer

Rechercher des objets en fonction du préfixe

<input type="checkbox"/>	Nom	Type	Dernière modification
<input type="checkbox"/>	.DS_Store	DS_Store	16 Jan 2023 11:29:33 AM CET
<input type="checkbox"/>	bootstrap-emr.sh	sh	16 Jan 2023 03:31:31 PM CET
<input type="checkbox"/>	config.json	json	16 Jan 2023 12:20:49 PM CET
<input type="checkbox"/>	Fruits/	Dossier	-
<input type="checkbox"/>	jupyter/	Dossier	-
<input type="checkbox"/>	Results/	Dossier	-

Amazon S3 > Compartiments > p8-hanane-maghazi > Fruits/

Fruits/

Objets Propriétés

Objets (6)

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 dans votre bucket. Vous devez leur accorder explicitement des autorisations. [En savoir plus](#)

Copier l'URI S3 Copier l'URL Télécha

Rechercher des objets en fonction du préfixe

<input type="checkbox"/>	Nom	Type
<input type="checkbox"/>	.DS_Store	DS_Store
<input type="checkbox"/>	Apple Braeburn/	Dossier
<input type="checkbox"/>	Apple Pink Lady/	Dossier
<input type="checkbox"/>	Avocado/	Dossier
<input type="checkbox"/>	Kiwi/	Dossier
<input type="checkbox"/>	Orange/	Dossier

Choix de la solution technique : EMR

- **Choix de la solution technique : EMR (Elastic MapReduce)**
 - Dans le cadre de ce projet nous opterons pour une solution **PAAS** (Plateforme As A Service)
 - **AWS** fournit énormément de services différents, dans l'un de ceux-là il existe une offre qui permet de louer des **instances EC2** avec des applications **préinstallées** et configurées : il s'agit du **service EMR**.
 - **Spark** y sera déjà installé
 - Possibilité de demander l'installation de **Tensorflow** ainsi que **JupyterHub**
 - Possibilité d'indiquer des **packages complémentaires** à installer à l'initialisation du serveur **sur l'ensemble des machines du cluster**.

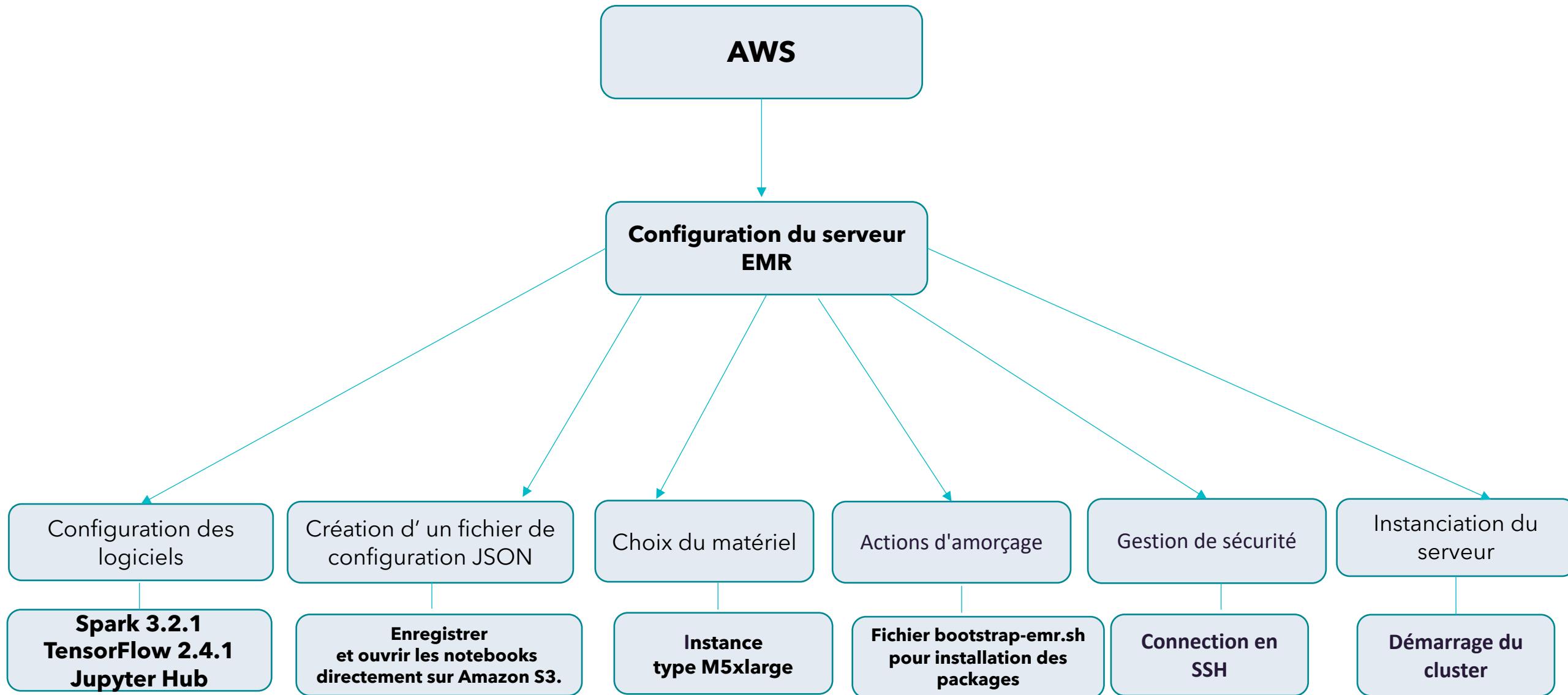
Avantages:

- Facilité de mise en œuvre
- Rapidité de mise en œuvre
- Solutions matérielles et logicielles optimisées par les ingénieurs d'AWS
- Stabilité de la solution
- Solution évolutive
- Plus sécurisé

Inconvénients :

Version des packages installés

Etapes de configuration :



- **Actions d'amorçage:**

- Afin de **choisir les packages manquants à installer**, la procédure est de créer un fichier **bootstrap** qui contient l'ensemble des instructions permettant d'installer tous les packages dont nous aurons besoin :

```
#!/bin/bash

sudo python3 -m pip install -U setuptools
sudo python3 -m pip install -U pip
sudo python3 -m pip install wheel
sudo python3 -m pip install pillow
sudo python3 -m pip install pandas==1.2.5
sudo python3 -m pip install matplotlib
sudo python3 -m pip install pyarrow
sudo python3 -m pip install s3fs
sudo python3 -m pip install fsspec
```

EMR

Cluster : P8_Fruits Démarrage en cours

Récapitulatif Historique de l'application Surveillance Matériel Configurations Événements Étapes Actions d'amorçage

Récapitulatif

ID : j-3AOA5ELFFBH6A

Date de création : 19-01-2023 17:18 (UTC+1)

Temps écoulé : 0 secondes

Résiliation automatique : Cluster waits

Protection de la résiliation : Désactivé [Modification](#)

Balises : -- [Afficher tout/Modifier](#)

DNS public principal : --

Application user interfaces

Service d'historique : [--](#)

Connexions : [--](#)

Détails de configuration

Étiquette de version : emr-6.7.0

Distribution Hadoop : Amazon 3.2.1

Applications : JupyterHub 1.4.1, TensorFlow 2.4.1, Spark 3.2.1

URI de connexion : s3://aws-logs-153437555282-eu-west-1
/elasticmapreduce/ [\[dossier\]](#)

Vue cohérente EMRFS : Désactivé

ID d'AMI personnalisée : --

Version d'Amazon Linux : 2.0.20221210.1 [En savoir plus](#) [\[info\]](#)

Réseau et matériel

Zone de disponibilité : --

ID de sous-réseau (subnet) : [subnet-0904761cc990471c1](#) [\[info\]](#)

Maître : Mise en service 1 m5.xlarge

Principal : Mise en service 2 m5.xlarge

Tâche : --

Cluster scaling: Not enabled

Résiliation automatique : Arrêter en cas d'inactivité pour 1 heure

Sécurité et accès

Nom de clé : AwsKey

Profil d'instance EC2 : EMR_EC2_DefaultRole

- Les instances sont de type **M5** qui sont des **instances de type équilibrés**
- De type **xlarge** qui est l'instance la moins onéreuse disponible
- On a **1 instance Maître** (le driver) et **2 instances Principales** (les workeurs), soit **un total de 3 instance EC2**.

Service IAM

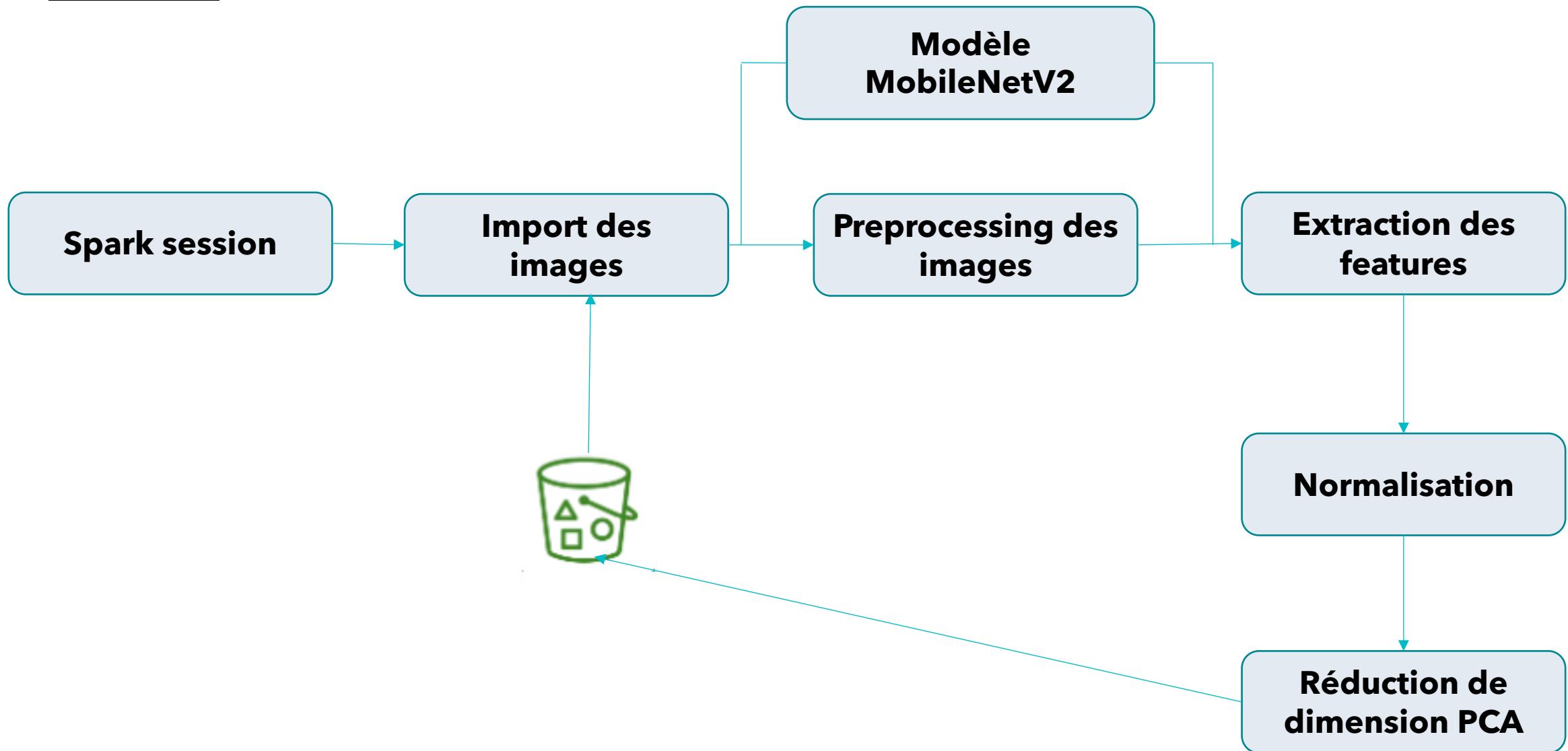
IAM est le service central dans AWS responsable de l'**identification et l'autorisation des utilisateurs, groupes et rôles**.



- Création d'un **utilisateur** et lui attribuer la stratégie "**AmazonS3FullAccess**" : créer des buckets, déposer des fichiers, les supprimer, etc.
- Création d'une **politique IAM** "**AmazonEMRFullAccessPolicy_v2**" pour l'utilisateur afin d'accéder à l'ensemble des clusters EMR.
- Pour configurer de nombreux services AWS : Transmission d' **un rôle IAM** au service. Cela permet au service d'assumer le rôle plus tard et d'effectuer des **actions** au nom de l'utilisateur.

Traitements des images :

- Processus :



Traitement des images

- Le code sera exécuté sous JupyterHub hébergé sur notre cluster EMR
- **Démarrage de la session Spark :**

```
1 # L'exécution de cette cellule démarre l'application Spark
2 # Changer le kernel en celui de spark
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	User	Current session?
0	application_1675373267878_0001	pyspark	idle	Link	Link	None	✓

```
1 %%info
```

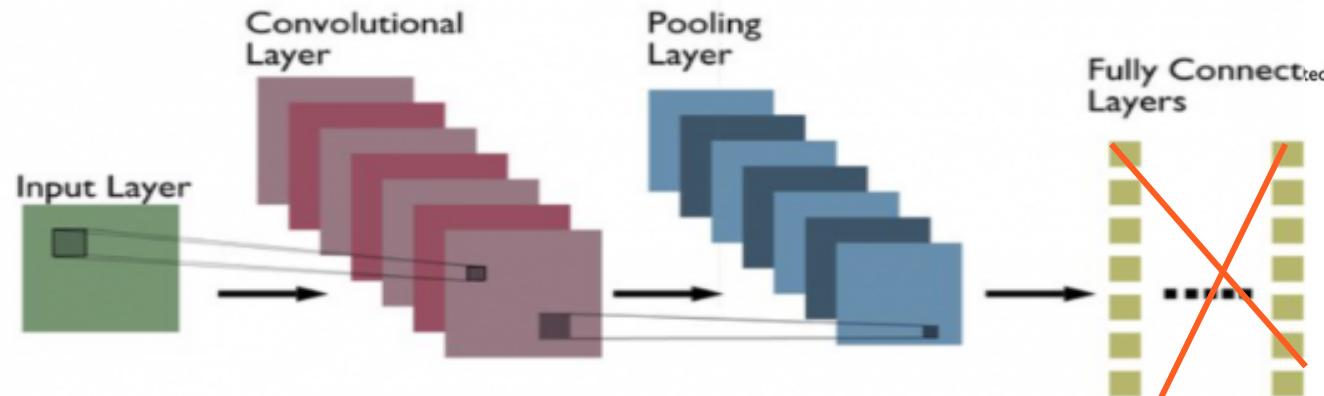
Current session configs: { 'driverMemory': '1000M', 'executorCores': 2, 'proxyUser': 'jovyan', 'kind': 'pyspark' }

ID	YARN Application ID	Kind	State	Spark UI	Driver log	User	Current session?
0	application_1675373267878_0001	pyspark	idle	Link	Link	None	✓

Traitements des images

- **Préparation du modèle :**

- ✓ CNN Transfer Learning : Les CNN sont des réseaux de neurones spécialement conçus pour traiter des images en entrée
- ✓ Utilisation du **MobileNetV2** en tant que Pre-Trained Model as **Feature Extractor Preprocessor**



Traitement des images

Architecture globale du modèle MobileNetV2

Input	Operator	<i>t</i>	<i>c</i>	<i>n</i>	<i>s</i>
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

```
model = MobileNetV2(weights='imagenet',
                     include_top=True,
                     input_shape=(224, 224, 3))
```

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

Chargement du modèle **MobileNetV2** avec les poids **précalculés** issus d'**imagenet** et en spécifiant le format des images en entrée

en entrée : l'entrée du modèle MobileNetV2

en sortie : l'avant dernière couche du modèle MobileNetV2

Traitement des images

- Le code comprend les éléments suivants:
 - ✓ La fonction de chargement des données qui renvoie un Data Frame
 - ✓ Définition d' une fonction que nous voulons faire en tant que UDF.
 - ✓ Renvoie un nouveau Data Frame avec une colonne correspondante aux features
 - ✓ Normalisation des features
 - ✓ La réduction de dimension par PCA

Traitement des images

- **Chargement des images :**

- Importer les images dans un data frame
- Les images sont chargées au format binaire

path	modificationTime	length	content
s3://p8-hanane-ma...	2023-01-17 09:54:15	5452	[FF D8 FF E0 00 1...
s3://p8-hanane-ma...	2023-01-17 09:54:15	5423	[FF D8 FF E0 00 1...
s3://p8-hanane-ma...	2023-01-17 09:54:14	5409	[FF D8 FF E0 00 1...
s3://p8-hanane-ma...	2023-01-17 09:54:14	5397	[FF D8 FF E0 00 1...
s3://p8-hanane-ma...	2023-01-17 09:54:14	5386	[FF D8 FF E0 00 1...

Traitement des images

- **Preprocessing et extraction des features:**

Redimensionnement de l'image : de 100* 100 à [224, 224]

- **Traitement de diffusion des poids du modèle Tensorflow sur les clusters :**

```
broadcast_weights = sc.broadcast(new_model.get_weights())
```

- **Application de la featurisation des images à travers l'utilisation de pandas UDF**

path	label	features
s3://p8-hanane-ma...	Orange	[0.033774916, 0.0...
s3://p8-hanane-ma...	Apple Pink Lady	[0.6022466, 0.032...
s3://p8-hanane-ma...	Orange	[0.29246765, 0.0,...
s3://p8-hanane-ma...	Kiwi	[1.3506836, 0.003...
s3://p8-hanane-ma...	Apple Pink Lady	[0.34955275, 0.0,...

Traitement des images

- Réduction de dimension de type PCA en PySpark

- Conversion des features en Vector dense :

```
# Transform array to vector for PCA
ud_f = udf(lambda r: Vectors.dense(r), VectorUDT())
features_vect = features_df.select('path', 'label', ud_f('features').alias('features_vec'))
```

- StandardScaler pour la normalisation des données avant PCA :

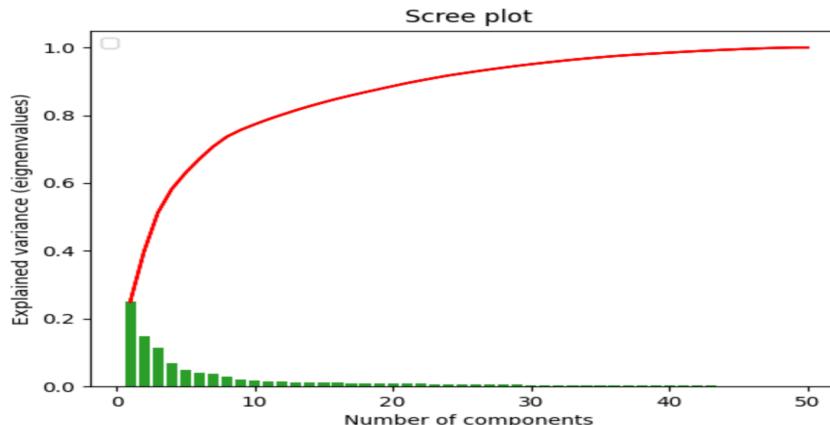
```
# Standardiser les données avant la PCA
standardizer = StandardScaler(withMean=True, withStd=True, inputCol="features_vec", outputCol="features_scaled")
model = standardizer.fit(features_vect)
result = model.transform(features_vect)
features_vect_scaled= result.select('path', 'label', 'features_scaled')
```

Traitement des images

- PCA :

```
1 # PCA
2 pca = PCAml(k=50, inputCol="features_scaled", outputCol="features_pca")
3 model = pca.fit(features_vect_scaled)
4 features_df_pca = model.transform(features_vect_scaled)
```

```
model.explainedVariance.sum()
0.9999999999999989
```



path	label	features_pca
s3://p8-hanane-ma...	Orange	[8.18036660117578...]
s3://p8-hanane-ma...	Orange	[6.90666432619188...]
s3://p8-hanane-ma...	Apple Pink Lady	[13.5566886964658...]
s3://p8-hanane-ma...	Apple Pink Lady	[15.9440812907747...]
s3://p8-hanane-ma...	Apple Pink Lady	[14.6141329175917...]

Traitement des images

- **Enregistrement des données enregistrées :**

- Sous format parquet :

```
# Enregistrement des données traitées au format "parquet" :  
features_df_pca.write.mode("overwrite").parquet("s3://p8-hanane-maghazi")
```

- Sous format CSV:

```
df_pca.to_csv("s3://p8-hanane-maghazi/result.csv", index=False)
```

Traitement des images

- Présence des fichiers au format "**parquet**" et **csv** sur le **serveur S3** :

<input type="checkbox"/>	 _SUCCESS	-	02 Feb 2023 10:47:09 PM CET	0 o	Standard
<input type="checkbox"/>	 part-00000-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:46:57 PM CET	4.7 Ko	Standard
<input type="checkbox"/>	 part-00001-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:46:53 PM CET	4.8 Ko	Standard
<input type="checkbox"/>	 part-00002-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:46:57 PM CET	4.8 Ko	Standard
<input type="checkbox"/>	 part-00003-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:47:00 PM CET	4.7 Ko	Standard
<input type="checkbox"/>	 part-00004-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:47:00 PM CET	4.7 Ko	Standard
<input type="checkbox"/>	 part-00005-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:47:03 PM CET	4.8 Ko	Standard
<input type="checkbox"/>	 part-00006-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:47:03 PM CET	4.8 Ko	Standard
<input type="checkbox"/>	 part-00007-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:47:06 PM CET	4.7 Ko	Standard
<input type="checkbox"/>	 part-00008-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:47:06 PM CET	4.8 Ko	Standard
<input type="checkbox"/>	 part-00009-48e9d95e-e973-44cd-abbc-c2019d7b1975-c000.snappy.parquet	parquet	02 Feb 2023 10:47:09 PM CET	4.8 Ko	Standard
<input type="checkbox"/>	 result.csv	csv	02 Feb 2023 10:49:06 PM CET	110.6 Ko	Standard

Suivi de l'avancement des tâches avec le Serveur d'Historique Spark

- Suivi de l'avancement des tâches avec le Serveur d'Historique Spark

Completed Jobs (20)					
Job Id (Job Group) ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
19 (26)	Job group for statement 26 toPandas at <stdin>:4	2023/02/03 17:52:45	19 s	1/1 (1 skipped)	10/10 (2 skipped)
18 (26)	Job group for statement 26 toPandas at <stdin>:4	2023/02/03 17:52:44	0.9 s	1/1	2/2
17 (23)	Job group for statement 23 parquet at NativeMethodAccesso...:0	2023/02/03 17:52:05	28 s	1/1 (1 skipped)	10/10 (2 skipped)
16 (23)	Job group for statement 23 parquet at NativeMethodAccesso...:0	2023/02/03 17:51:55	9 s	1/1	2/2
15 (18)	Job group for statement 18 showString at NativeMethodAccesso...:0	2023/02/03 17:49:02	3 s	1/1 (1 skipped)	1/1 (2 skipped)
14 (18)	Job group for statement 18 showString at NativeMethodAccesso...:0	2023/02/03 17:48:59	3 s	1/1 (1 skipped)	1/1 (2 skipped)
13 (18)	Job group for statement 18 showString at NativeMethodAccesso...:0	2023/02/03 17:48:58	1 s	1/1	2/2
12 (17)	Job group for statement 17 treeAggregate at RowMatrix.scala:156	2023/02/03 17:48:24	16 s	2/2 (1 skipped)	12/12 (2 skipped)
11 (17)	Job group for statement 17 isEmpty at RowMatrix.scala:426	2023/02/03 17:48:21	3 s	1/1 (1 skipped)	1/1 (2 skipped)
10 (17)	Job group for statement 17 treeAggregate at Statistics.scala:58	2023/02/03 17:48:03	18 s	2/2 (1 skipped)	12/12 (2 skipped)
9 (17)	Job group for statement 17 first at RowMatrix.scala:62	2023/02/03 17:48:00	3 s	1/1 (1 skipped)	1/1 (2 skipped)
8 (17)	Job group for statement 17 first at PCA.scala:44	2023/02/03 17:47:56	4 s	2/2	3/3
7 (16)	Job group for statement 16 first at StandardScaler.scala:113	2023/02/03 17:47:47	0.1 s	1/1 (2 skipped)	1/1 (12 skipped)
6 (16)	Job group for statement 16 first at StandardScaler.scala:113	2023/02/03 17:47:30	18 s	1/1 (1 skipped)	10/10 (2 skipped)
5 (16)	Job group for statement 16 first at StandardScaler.scala:113	2023/02/03 17:47:28	1 s	1/1	2/2

Conclusion :

- Configuration d'une solution BigData dans le cloud:

- ✓ Mise en place d'un bucket **S3**
- ✓ Gestion des droits sur **S3** avec le service **IAM**
- ✓ Configuration du service **EMR**



- Traitement des images :

- ✓ Chargement du bucket
- ✓ Prétraitement des images
- ✓ Utilisation des pandas UDF
- ✓ Transfer Learning à partir du modèle MobileNetV2 as Feature Extractor Preprocessor
- ✓ Enregistrement du résultat sur S3



Fruits!

- **Difficultés** :

- ✓ Mise en place de l'environnement Spark
- ✓ Plusieurs choix de versions (Spark, Hadoop, tensorflow) et packages(pandas)

- **Aller plus loin** :

- ✓ Augmenter le nombre d'images
- ✓ Faire face à une montée de la charge de travail en redimensionnant simplement notre cluster de machines
- ✓ Faire le Suivi en détail de l'avancement des tâches avec le Serveur d'Historique Spark afin de debugger, optimiser les futurs tâches à réaliser.

Annexe

- Lien GitHub pour le code en local et sur le cloud:

https://github.com/HananeMaghlazi/P8_BigData

- Lien des données sur s3:

s3://p8-hanane-maghlazi/Fruits/

<https://p8-hanane-maghlazi.s3.eu-west-1.amazonaws.com/Fruits/>

- Lien du résultat :

s3://p8-hanane-maghlazi/Results/

<https://p8-hanane-maghlazi.s3.eu-west-1.amazonaws.com/Results/>

Discussion :



FIN

• **MERCI**