# Projet 8 :
# Déployez un modèle dans le cloud

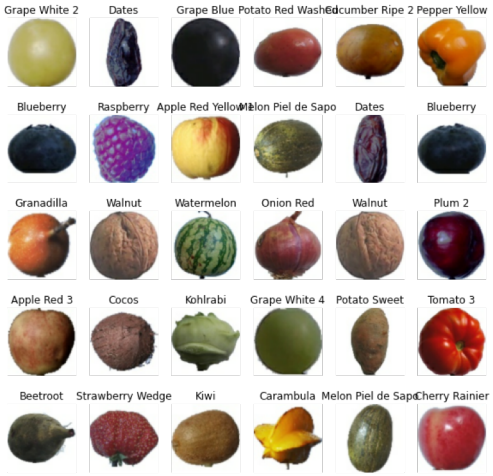
# Sommaire

Fruits!

# 01- Présentation du projet

## Mission: Déployez un modèle dans le cloud

• Une jeune start-up de l'AgriTech, nommée **"Fruits!"**, nous missionne pour le développement d'une application mobile qui permettrait aux utilisateurs de prendre en photo un fruit et d'obtenir des informations sur ce fruit.




## Objectif du projet:

• Développement d'un environnement Big Data qui comprendra le preprocessing et une étape de réduction de dimension;

• Passage à l'échelle en termes de volume de données;

# 02- Etude des données

## Sources des données sur Kaggle - MIHAI OLTEAN:

https://www.kaggle.com/datasets/moltean/fruits

**Le jeu de données est un ensemble d'images de fruits et de labels combinés:**

- On a différentes variétés d'un même fruit (pomme par exemple) ;
- 131 variétés différentes (fruits et légumes);

**Images:**

- Nombre total d'images : 90 483
- Jeu d'entraînement : 67 692 images;
- Jeu de test : 22 688 images;
- Jeu multi-fruits : 103 images;

**Format & Prise de vue:** https://youtu.be/_HFKJ144JuU

- Taille de l'image : 100 x 100 pixels;
- Photos prises avec webcam (Logitech C920) sur un axe tournant à 360° (moteur électrique);
- Et sur un fond blanc (papier blanc);

# 03- L'Architecture Big Data & La chaîne de traitement

## Les enjeux du Big data (*méga données* en français)

**Les 3V du big data :**
**Volume, Vélocité, Variété**

- **Le *Volume* des données générées** nécessite de repenser la manière dont elles sont stockées;
- **La Vélocité à laquelle nous parviennent 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).
- …

# 03- L'Architecture Big Data & La chaîne de traitement

**Comment répondre à ces enjeux? Plusieurs contraintes se posent:**

**Quel prestataire de Cloud choisir ?**

**Amazon Web Services (AWS):**
**- offre la plus large dans le cloud computing;**
**- Certaines de leurs offres sont adaptées à notre problématique;**
**→ louer de la puissance de calcul à la demande.**
**→ diminuer les coûts.**

# 03- L'Architecture Big Data & La chaîne de traitement

**Comment répondre à ces enjeux? Plusieurs contraintes se posent:**

## Où stocker nos données ?

### Serveur EC2:

C'est une solution de stockage de nos données sur l'espace alloué par le serveur **EC2**,
- Mais si nos données venaient à **saturer** l'espace disponible de nos serveurs (ralentissements, dysfonctionnements).
- Les données seront **perdues** lorsque le serveur sera résilié (on résilie le serveur lorsqu'on ne s'en sert pas pour des raisons de coût).

→ imaginer une solution pour sauvegarder les données avant la résiliation du serveur.

### Amazon S3 (Amazon Simple Storage Service ):

- L'espace disque disponible est illimité, et il est indépendant de nos serveurs EC2.
- L'accès aux données est très rapide car nous restons dans l'environnement d'AWS;
- Il est possible d'accéder aux données sur S3 de la même manière que l'on accède aux données sur un disque local. Nous utiliserons simplement un PATH au format s3://... .

### Stockage de données sur HDFS:

Dans une application Spark exécutée sur un cluster EMR, les fichiers contenant des données ne sont pas chargés à partir du système de fichier local, mais à partir d'HDFS, le système de fichier distribué d'Hadoop.

→ stocker les données directement sur votre cluster dans certains cas, par exemple pour accélérer le chargement des données.

# 03- L'Architecture Big Data & La chaîne de traitement

## Comment répondre à ces enjeux? Plusieurs contraintes se posent:

**Quelles solutions de ce prestataire adopter ?** Plusieurs solutions s'offre à nous :

### → 1/ Solution IAAS (Infrastructure AS A Service)

Dans cette configuration **AWS** met à notre disposition des serveurs vierges sur lequel nous avons un accès en administrateur, ils sont nommés **instance EC2**.
→ cette solution reproduit pratiquement à l'identique la solution mis en œuvre en local sur notre machine.

On installe nous-même l'intégralité des outils puis on exécute notre script (spark, python, jupyter, Java, …)

**Avantages :**
-Liberté totale de mise en œuvre de la solution;
-Facilité de mise en œuvre à partir d'un modèle qui s'exécute en local sur une machine Linux;

**Inconvénients :**
-Chronophage;
-Nécessité d'installer et de configurer toute la solution;
-Possible problèmes techniques à l'installation des outils (des problématiques qui n'existaient pas en local sur notre machine - peuvent apparaitre sur le serveur EC2);
-Solution non pérenne dans le temps : mise à jour des outils et éventuellement devoir réinstaller Spark, Java etc;

# 03- L'Architecture Big Data & La chaîne de traitement

## Comment répondre à ces enjeux? Plusieurs contraintes se posent:

### → 2/ Solution PAAS (Plateforme As A Service)

**AWS** 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,** et d'autres **packages complémentaires** à l'initialisation du serveur **sur l'ensemble des machines du cluster**.

**Avantages :**
- Facilité de mise en œuvre; Peu de configuration pour obtenir un environnement fonctionnel; Rapidité de mise en œuvre;
- Solution de clonage des clusters; Solutions matérielles et logicielles optimisées par les ingénieurs d'AWS; Stabilité de la solution; Solution évolutive Il est facile d'obtenir à chaque nouvelle instanciation une version à jour de chaque package, en étant garanti de leur compatibilité avec le reste de l'environnement.
Plus sécurisé; Les éventuels patchs de sécurité seront automatiquement mis à jour à chaque nouvelle instanciation du cluster EMR.

**Inconvénients :**
-Peut-être un certain manque de liberté sur la version des packages disponibles ? Même si je n'ai pas constaté ce problème.

# 03- L'Architecture Big Data & La chaîne de traitement

**Comment répondre à ces enjeux? Plusieurs contraintes se posent:**

## Solution technique retenue: PAAS

- **Je retiens la solution PAAS en choisissant d'utiliser le service EMR d'Amazon Web Services;**
- **Je la trouve plus adaptée à notre problématique et permet une mise en œuvre qui soit à la fois plus rapide et plus efficace que la solution IAAS;**

## Solution de stockage retenue : Amazon S3

- Je retiens la solution Amazon S3 car elle permet de s'affranchir de toutes les problématiques lié à EC2;
- Solution efficace pour la gestion du stockage des données;

# 03- L'Architecture Big Data & La chaîne de traitement

**Prétraitement:**

**Préparation des images pour le transfert learning**

Preprocessing

Réduction de dimension de type PCA

Modèle MobileNetV2 retenue (préentrainé)

Rapidité d'exécution

Adaptée pour le traitement d'un gros volume de données

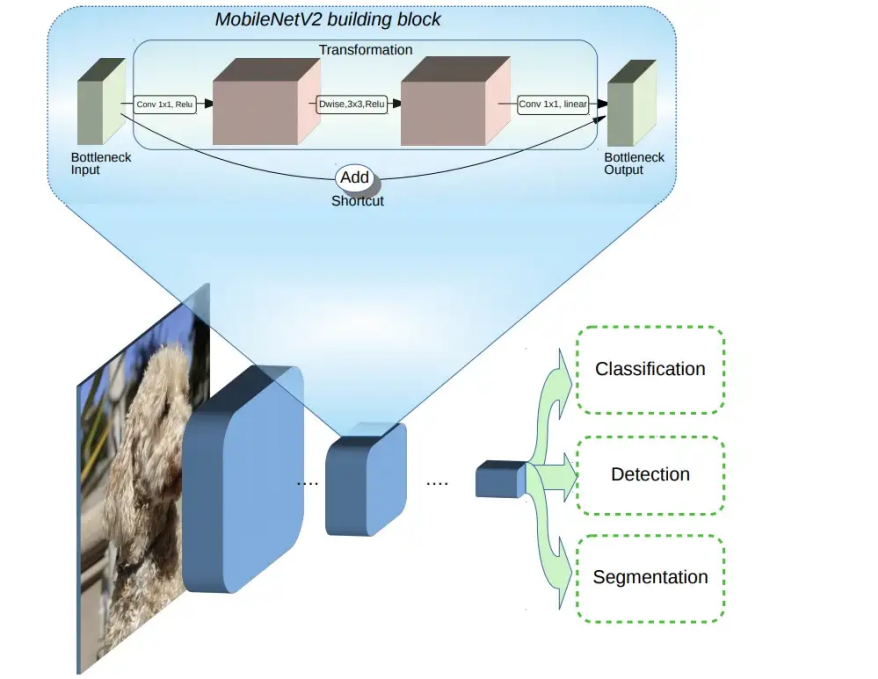
Faible dimensionnalité du vecteur de caractéristique en sortie (1,1,1280)

Solutions envisageables
→ Traitement d'image + Extraction de features (algo CNN, SIFT,…)

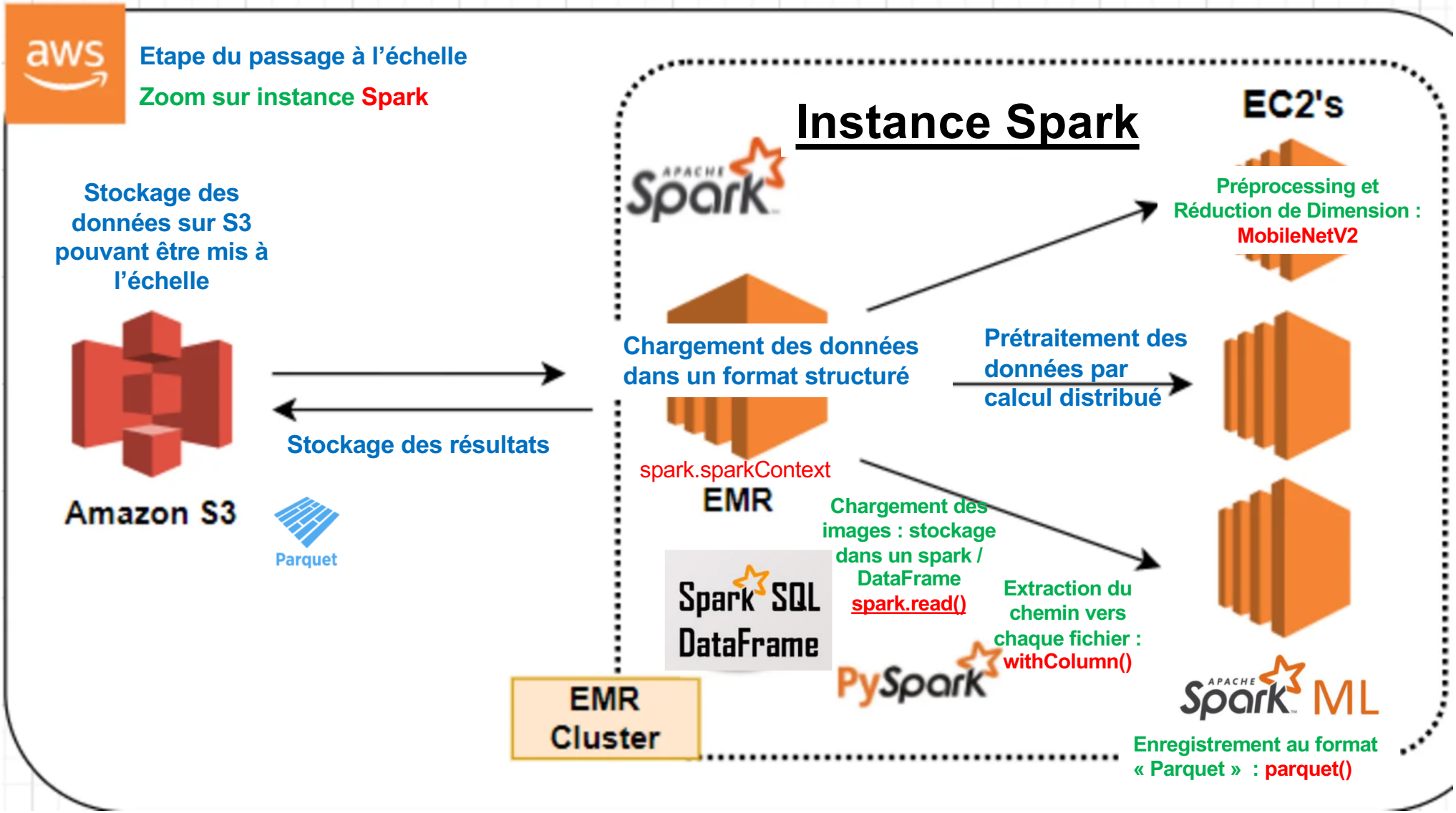
# 03- L'Architecture Big Data & La chaîne de traitement

## Modèle: MobileNetV2

Voici le schéma de son architecture globale :




| Input | Operator | $t$ | $c$ | $n$ | $s$ |
|---|---|---|---|---|---|
| $224^2 \times 3$ | 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 | - | |


Etape du passage à l'échelle
Zoom sur instance Spark
Instance Spark
EC2's
Stockage des données sur S3 pouvant être mis à l'échelle
Amazon S3
Parquet
Stockage des résultats
Chargement des données dans un format structuré
spark.sparkContext
EMR
Préprocessing et Réduction de Dimension : MobileNetV2
Prétraitement des données par calcul distribué
Chargement des images : stockage dans un spark / DataFrame spark.read()
Spark SQL DataFrame
PySpark
Extraction du chemin vers chaque fichier : withColumn()
ML
Enregistrement au format « Parquet » : parquet()
EMR Cluster

# 03- L'Architecture Big Data & La chaîne de traitement

## Passage à l'échelle

→ Stockage des données : Amazon S3 : OK
(Alternative : HDFS sur plusieurs serveurs)

→ Évolution de l'infrastructure de calcul:
Instance EC2 avec grande capacité RAM/Processeur
→ Remplacement par un cluster EMR (Elastic Map Reduce) avec plusieurs instances EC2 (1 Maître + n esclaves) : Configuration automatique;

→ Évolution sans risque de coupure : Pas de modification du code Spark/Python;

→2ème Etape: augmentation du nombre d'instances esclaves / nœuds ←

# 04- Conclusions

**Compétences développés**

- Manipulation Pyspark
- Connaissance du format distribué « parquet »
- Connaissance des offres AWS
- Prise en main d'un serveur Linux par SSH

**Difficultés rencontrées**

- Choix complexes  sur une stratégie technique (Plusieurs Possibilités)
- Beaucoup de débuggage à effectuer (erreurs peu explicites sur la configuration Spark/Java/S3)

**Propositions d'évolution:**

- Amélioration du prétraitement (recadrage, plusieurs fruits, etc.) ;
- Entraînement du modèle (approche transfert learning);
- Déployement du modèle en production (sur un cluster);

# Merci

Avez-vous des questions?