



# Déployer un modèle dans le Cloud

---

Guilhem Berthou - Pierre-Antoine Ganaye (mentor)

# Introduction

---

**Contexte** : La start-up *Fruits!* recherche des solutions innovantes pour la récolte de fruits, notamment en développant des robots cueilleurs intelligents qui réserveraient à chaque espèce un traitement spécifique.

L'entreprise souhaite dans un premier temps mettre en œuvre une application mobile de **reconnaissance d'image de fruits** pour se faire connaître.



Traitement des images de  
fruits



Réaliser une **classification**  
des images de fruits



Mettre en place une  
architecture **Big Data**

## ⇒ Objectif du projet :

- Mise en place de la **chaîne de pré-traitement** des images dans un **environnement Big Data** afin de préparer le passage à l'échelle
- La *classification* des images est pour l'instant volontairement *mise de côté*.

# Sommaire

---

## 1. Choix techniques retenus

- a. Architecture Big Data et passage à l'échelle
  - i. Briques d'architectures nécessaires pour mettre en place un environnement big Data
  - ii. Identification des opérations critiques lors d'un passage à l'échelle
- b. Solution de transfer learning
- c. Performance du calcul distribué (*pyspark*)

## 2. Mise en oeuvre

- a. Test de la chaîne de traitement en local
- b. Déploiement d'une architecture Big Data dans le Cloud *AWS*
  - i. Conformité au RGPD (stockage et traitement sur des serveurs européens)
  - ii. Téléchargement des données sur S3
  - iii. Infrastructure Hadoop (*Cluster EMR*) - Gestion de machine virtuels *EC2*
  - iv. Lancement du notebook de pré-traitement via *JupyterHub*

## 3. Détail de la chaîne de pré-traitement

- a. Architecture *MobileNet v2*
- b. Étapes de traitement
  - i. Conversion des images en matrice (*img\_to\_array* from Keras)
  - ii. Prédiction et aplatissement des outputs
  - iii. Pandas UDF (Spark Dataframe & *yield* keyword)
  - iv. Stockage des outputs au format Apache parquet

## 4. Conclusion (points d'attention et retour d'expérience)

# Sommaire

---

## 1. Choix techniques retenus

- a. Architecture Big Data et passage à l'échelle
  - i. Briques d'architectures nécessaires pour mettre en place un environnement big Data
  - ii. Identification des opérations critiques lors d'un passage à l'échelle
- b. Solution de transfer learning
- c. Performance du calcul distribué (*pyspark*)

## 2. Mise en oeuvre

- a. Test de la chaîne de traitement en local
- b. Déploiement d'une architecture Big Data dans le Cloud *AWS*
  - i. Conformité au RGPD (stockage et traitement sur des serveurs européens)
  - ii. Téléchargement des données sur *S3*
  - iii. Infrastructure Hadoop (*Cluster EMR*) - Gestion de machine virtuels *EC2*
  - iv. Lancement du notebook de pré-traitement via *JupyterHub*

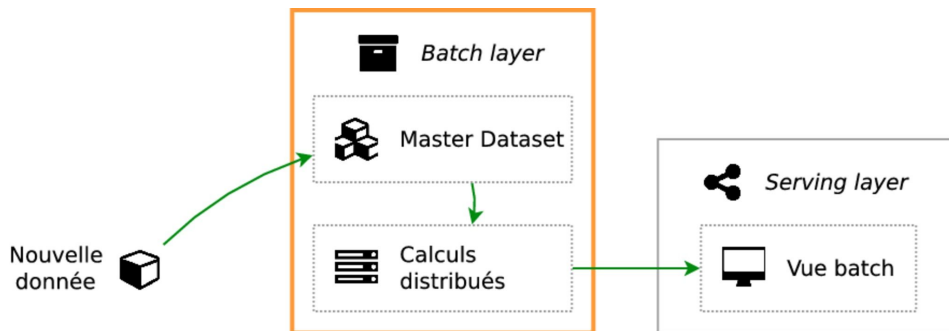
## 3. Détail de la chaîne de pré-traitement

- a. Architecture *MobileNet v2*
- b. Étapes de traitement
  - i. Conversion des images en matrice (*img\_to\_array* from Keras)
  - ii. Prédiction et aplatissement des outputs
  - iii. Pandas UDF (Spark Dataframe & *yield* keyword)
  - iv. Stockage des outputs au format Apache parquet

## 4. Conclusion (points d'attention et retour d'expérience)

- a. Architecture Big Data et passage à l'échelle
- b. Solution de transfer learning
- c. Performance du calcul distribué (pyspark)

➤ **Briques d'architectures** nécessaires pour mettre en place un **environnement Big Data**



➤ **Identification des opérations critiques** lors d'un **passage à l'échelle**

- **Sauvegarde** des données dans le *Cloud* (indépendance vis-à-vis du cluster de calcul)
- Choix du **modèle** d'extraction de *features* (transfer learning - cf 1.b)
- Pipeline de traitement à **paralléliser** (*pyspark*)

# Choix techniques retenus

---

1. Choix techniques retenus
  - a. Architecture Big Data et passage à l'échelle
  - b. **Solution de transfer learning**
  - c. Performance du calcul distribué (pyspark)

## ➤ Solution de *transfer learning* retenue : **MobileNetV2**

- **Entraînement** gourmand en ressource (GPU - CNN à 53 *couches*)
- Réutilisation de modèle dont la **performance** est a été éprouvée ([\*ImageNet\*](#))

## ➤ **Transfer Learning**

= **emprunter l'architecture de ses paramètres** à d'autres modèles pré-entraînés - *i.e* utiliser nos propres données et atteindre facilement une **précision élevée**.

- **Chargement** du modèle MobileNetV2 pré-entraîné (*keras - Google*)
- *Freeze* des **poids**/paramètres du modèle
- **Broadcast** des poids à tous les **noeux** du modèle (*cf 3.a*)
- **Choix des couches** appropriées à la tâche souhaitée
- **Entraînement** de nos données sur les couches sélectionnées

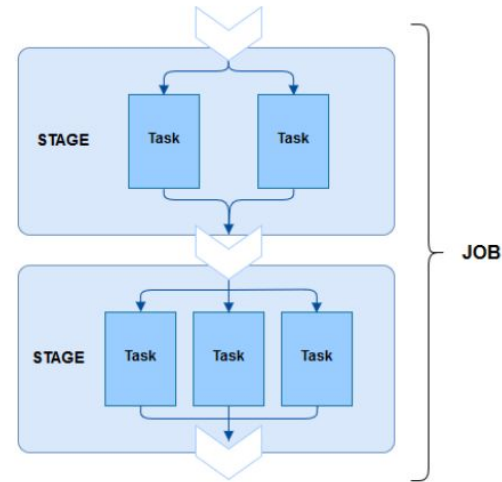
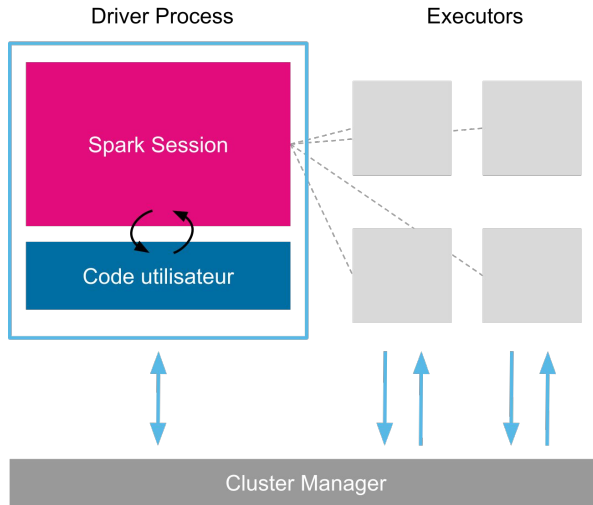
# Choix techniques retenus

## 1. Choix techniques retenus

- Architecture Big Data et passage à l'échelle
- Solution de transfer learning
- Performance du calcul distribué (pyspark)**

### ➤ pyspark

- Spark Framework *open source* de calcul **distribué** in-memory
- Destiné au traitement et l'analyse de **données massives**
- Construit en *Scala* et implémentable via l'API *python*

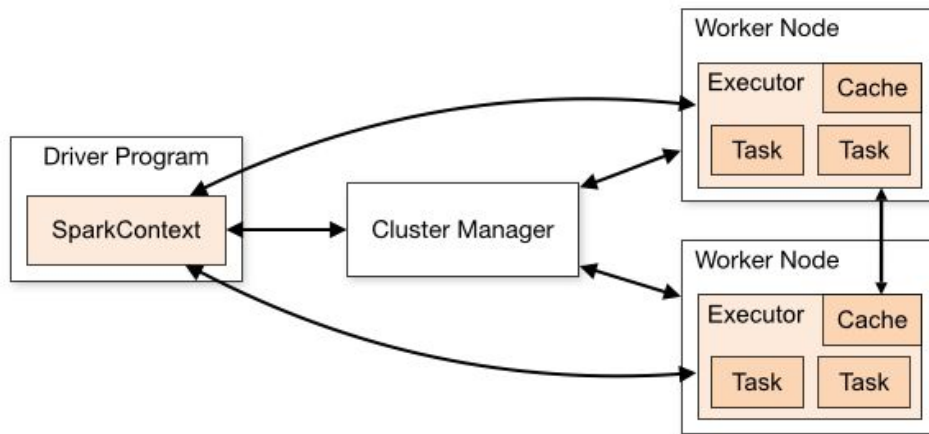


# Choix techniques retenus

1. Choix techniques retenus
  - a. Architecture Big Data et passage à l'échelle
  - b. Solution de transfer learning
  - c. **Performance du calcul distribué (pyspark)**

## ➤ pyspark

- Spark emploie un gestionnaire de groupe (**cluster manager**) qui assure le **suivi des ressources disponibles**
- Le processus de pilotage (**driver process**) est responsable de l'**exécution** du programme à travers les **exécuteurs** pour accomplir une tâche donnée





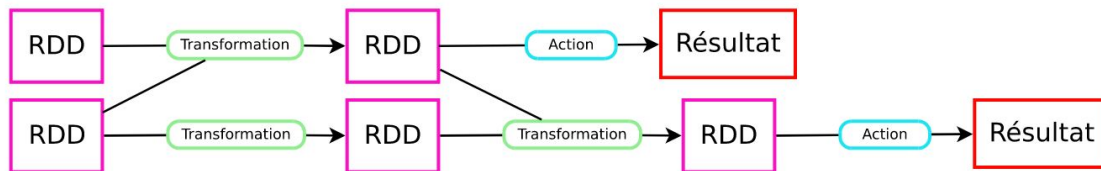
# Choix techniques retenus

## 1. Choix techniques retenus

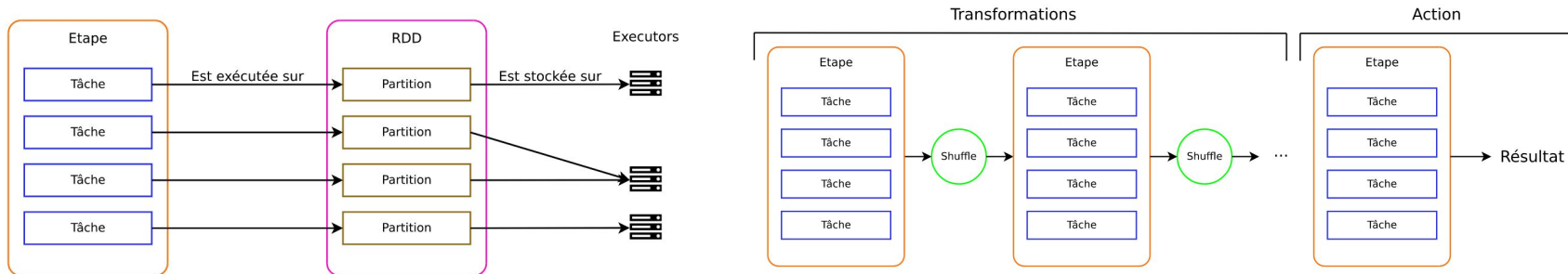
- Architecture Big Data et passage à l'échelle
- Solution de transfer learning
- Performance du calcul distribué (pyspark)**

### ➤ Resilient Distributed Datasets (RDD)

- Base de la structure des données *Spark*



### ➤ Répartition des données entre les *executors* (partitions)



# Sommaire

---

## 1. Choix techniques retenus

- a. Architecture Big Data et passage à l'échelle
  - i. Briques d'architectures nécessaires pour mettre en place un environnement big Data
  - ii. Identification des opérations critiques lors d'un passage à l'échelle
- b. Solution de transfer learning
- c. Performance du calcul distribué (*pyspark*)

## 2. Mise en oeuvre

- a. Test de la chaîne de traitement en local
- b. Déploiement d'une architecture Big Data dans le Cloud *AWS*
  - i. Conformité au RGPD (stockage et traitement sur des serveurs européens)
  - ii. Téléchargement des données sur S3
  - iii. Infrastructure Hadoop (*Cluster EMR*) - Gestion de machine virtuels *EC2*
  - iv. Lancement du notebook de pré-traitement via *JupyterHub*

## 3. Détail de la chaîne de pré-traitement

- a. Architecture *MobileNet v2*
- b. Étapes de traitement
  - i. Conversion des images en matrice (*img\_to\_array* from Keras)
  - ii. Prédiction et aplatissement des outputs
  - iii. Pandas UDF (Spark Dataframe & *yield* keyword)
  - iv. Stockage des outputs au format Apache parquet

## 4. Conclusion (points d'attention et retour d'expérience)

# Mise en oeuvre

## ➤ Installation de l'environnement de travail

- Machine Virtuelle : *VirtualBox VM*
- Système d'exploitation : *Linux Ubuntu*
- Installation de *Spark*

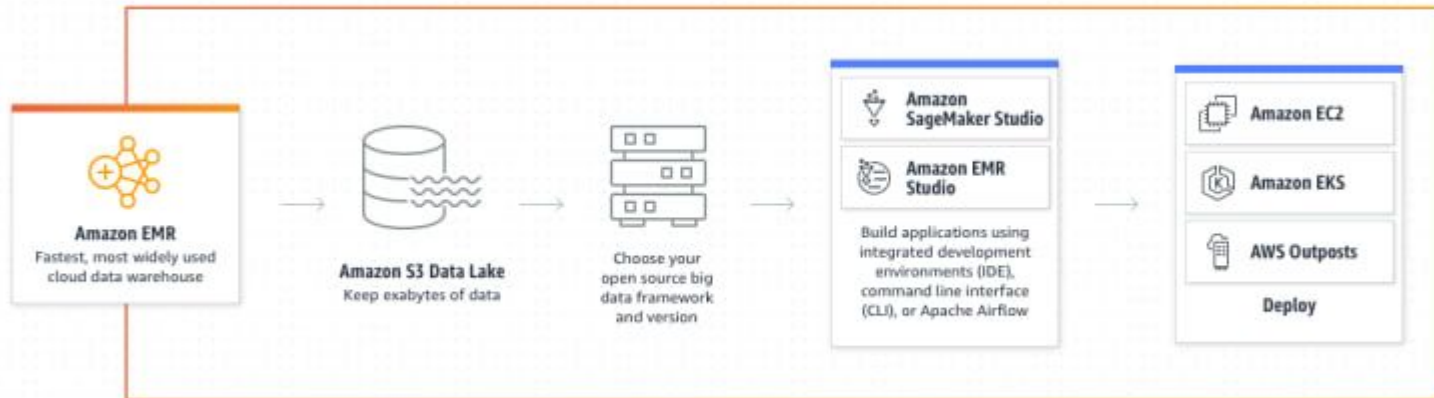
## ➤ Capitalisation sur le notebook Guide proposé

1. Test de la chaîne de traitement en local
2. Déploiement d'une architecture Big Data
  - a. Conformité au RGPD
  - b. Téléchargement des données sur S3
  - c. Infrastructure Hadoop
  - d. Lancement du notebook de pré-traitement



**amazon**  
**EMR**

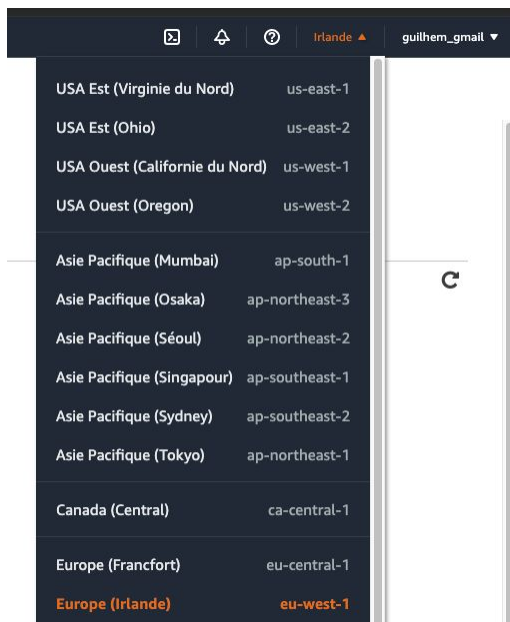
- **Cost-effective**
- **Scalability**
- **Flexibility**
- **Intégration** environnement AWS



# Mise en oeuvre

## ➤ Conformité au RGPD

- = **stockage et traitement** sur des serveurs européens



The screenshot shows the AWS Region selection interface. At the top, there are icons for a map, a refresh button, and a help icon, followed by the selected region 'Irlande' with an upward arrow and the user 'guilhem\_gmail' with a dropdown arrow. Below this is a scrollable list of regions. The 'Europe (Irlande)' region is highlighted in orange at the bottom of the list. To the right of the list, there is a vertical scrollbar and a circular arrow icon.

USA Est (Virginie du Nord)	us-east-1
USA Est (Ohio)	us-east-2
USA Ouest (Californie du Nord)	us-west-1
USA Ouest (Oregon)	us-west-2
Asie Pacifique (Mumbai)	ap-south-1
Asie Pacifique (Osaka)	ap-northeast-3
Asie Pacifique (Séoul)	ap-northeast-2
Asie Pacifique (Singapour)	ap-southeast-1
Asie Pacifique (Sydney)	ap-southeast-2
Asie Pacifique (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Francfort)	eu-central-1
<b>Europe (Irlande)</b>	<b>eu-west-1</b>

1. Test de la chaîne de traitement en local
2. **Déploiement d'une architecture Big Data**
  - a. **Conformité au RGPD**
  - b. Téléchargement des données sur S3
  - c. Infrastructure Hadoop
  - d. Lancement du notebook de pré-traitement

# Mise en oeuvre

## ➤ Téléchargement des données sur S3

1. Test de la chaîne de traitement en local
2. Déploiement d'une architecture Big Data
  - a. Conformité au RGPD
  - b. **Téléchargement des données sur S3**
  - c. Infrastructure Hadoop
  - d. Lancement du notebook de pré-traitement

The screenshot shows the Amazon S3 console interface. The left sidebar contains navigation options for Amazon S3, including Compartiments, Points d'accès, and Storage Lens. The main content area displays the 'oc-p8-bucket' with tabs for Objets, Propriétés, Autorisations, Métriques, Gestion, and Points d'accès. The 'Objets' tab is active, showing a list of 5 objects. Below the list, there are buttons for actions like 'Copier l'URI S3', 'Copier l'URL', 'Télécharger', 'Ouvrir', 'Supprimer', 'Actions', 'Créer un dossier', and 'Charger'. A search bar is also present above the object list.

	Nom	Type	Dernière modification	Taille	Classe de stockage
<input type="checkbox"/>	<a href="#">bootstrap-emr.sh</a>	sh	06 Feb 2023 05:59:58 PM CET	331.0 o	Standard
<input type="checkbox"/>	<a href="#">config_noeud.json</a>	json	23 Dec 2022 10:51:05 PM CET	122.0 o	Standard
<input type="checkbox"/>	<a href="#">jupyter/</a>	Dossier	-	-	-
<input type="checkbox"/>	<a href="#">Results/</a>	Dossier	-	-	-
<input type="checkbox"/>	<a href="#">Test/</a>	Dossier	-	-	-

1. Test de la chaîne de traitement en local
2. Déploiement d'une architecture Big Data
  - a. Conformité au RGPD
  - b. Téléchargement des données sur S3
  - c. Infrastructure Hadoop
  - d. Lancement du notebook de pré-traitement

## ➤ Infrastructure Hadoop

### ➤ (Cluster EMR) - Gestion de machines virtuelles EC2

The screenshot shows the AWS EMR console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user information. Below the navigation bar, there's a sidebar on the left with a menu for Amazon EMR, including options like EMR Studio, EMR Serverless, and EMR on EC2. The main content area displays a list of EMR clusters. A notification banner at the top of the main area states: "The new EMR console will become the default console on Feb 28, 2023. Switch to the new console. If you want, you can still switch back. Learn more." Below this, there are buttons for "Créer un cluster", "Afficher les détails", "Cloner", and "Résilier". The cluster list table has columns for "Nom", "ID", "Statut", "Heure de création (UTC+1)", "Temps écoulé", and "Heures d'instances normalisées". The first cluster, "p8-cluster", is in the "En attente" (Pending) state and is "Cluster prêt" (Cluster ready). The other six clusters are in the "Résilié" (Failed) state with the reason "Arrêt automatique" (Automatic stop).

	Nom	ID	Statut	Heure de création (UTC+1)	Temps écoulé	Heures d'instances normalisées
<input type="checkbox"/>	<a href="#">p8-cluster</a>	j-1JWMGJYAXQDG5	En attente Cluster prêt	09-02-2023 10:24 (UTC+1)	1 heure, 23 minutes	24
<input type="checkbox"/>	<a href="#">p8-cluster</a>	j-2EK402CDQ34F1	Résilié Arrêt automatique	07-02-2023 18:17 (UTC+1)	1 heure, 12 minutes	48
<input type="checkbox"/>	<a href="#">p8-cluster</a>	j-1ITNBHBDLFAXY	Résilié Arrêt automatique	07-02-2023 17:11 (UTC+1)	1 heure, 12 minutes	48
<input type="checkbox"/>	<a href="#">p8-cluster</a>	j-3UMWB40N57EKY	Résilié Arrêt automatique	07-02-2023 15:56 (UTC+1)	1 heure, 12 minutes	48
<input type="checkbox"/>	<a href="#">p8-cluster</a>	j-350JV2XQNMAW7	Résilié Arrêt automatique	07-02-2023 13:32 (UTC+1)	1 heure, 11 minutes	48
<input type="checkbox"/>	<a href="#">p8-cluster</a>	j-3D8KPU88SPWDV	Résilié Arrêt automatique	07-02-2023 12:19 (UTC+1)	1 heure, 12 minutes	48
<input type="checkbox"/>	<a href="#">p8-cluster</a>	j-3EU0009191VRR	Résilié Arrêt automatique	07-02-2023 10:41 (UTC+1)	1 heure, 13 minutes	48

# Mise en oeuvre

## ➤ Lancement du notebook de pré-traitement via JupyterHub

1. Test de la chaîne de traitement en local
2. **Déploiement d'une architecture Big Data**
  - a. Conformité au RGPD
  - b. Téléchargement des données sur S3
  - c. Infrastructure Hadoop
  - d. **Lancement du notebook de pré-traitement**

The screenshot shows the AWS S3 console interface. The left sidebar contains navigation options for Amazon S3, including Compartiments, Points d'accès, and Storage Lens. The main content area displays the 'jovyan/' bucket. Under the 'Objets' tab, there are two objects listed in a table:

	Nom	Type	Dernière modification	Taille	Classe de stockage
<input type="checkbox"/>	<a href="#">.s3keep</a>	s3keep	09 Feb 2023 10:50:43 AM CET	0 o	Standard
<input type="checkbox"/>	<a href="#">P8_01_notebook_Linux_EMR.ipynb</a>	ipynb	09 Feb 2023 11:41:56 AM CET	184.7 Ko	Standard

Below the table, there is a search bar and pagination controls showing 1 item.

# Sommaire

---

## 1. Choix techniques retenus

- a. Architecture Big Data et passage à l'échelle
  - i. Briques d'architectures nécessaires pour mettre en place un environnement big Data
  - ii. Identification des opérations critiques lors d'un passage à l'échelle
- b. Solution de transfer learning
- c. Performance du calcul distribué (*pyspark*)

## 2. Mise en oeuvre

- a. Test de la chaîne de traitement en local
- b. Déploiement d'une architecture Big Data dans le Cloud *AWS*
  - i. Conformité au RGPD (stockage et traitement sur des serveurs européens)
  - ii. Téléchargement des données sur *S3*
  - iii. Infrastructure Hadoop (*Cluster EMR*) - Gestion de machine virtuels *EC2*
  - iv. Lancement du notebook de pré-traitement via *JupyterHub*

## 3. Détail de la chaîne de pré-traitement

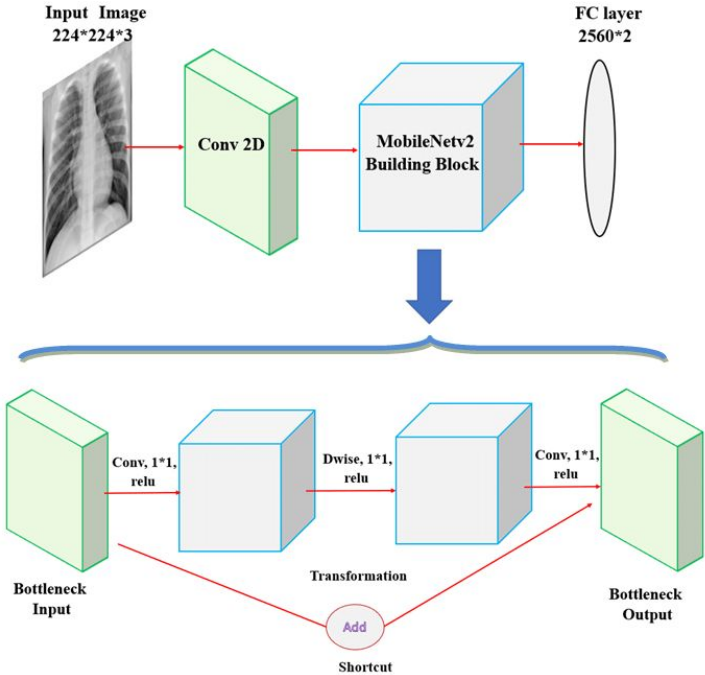
- a. Architecture *MobileNet v2*
- b. Étapes de traitement
  - i. Conversion des images en matrice (*img\_to\_array* from *Keras*)
  - ii. Prédiction et aplatissement des outputs
  - iii. *Pandas* UDF (*Spark Dataframe* & *yield* keyword)
  - iv. Stockage des outputs au format *Apache parquet*

## 4. Conclusion (points d'attention et retour d'expérience)

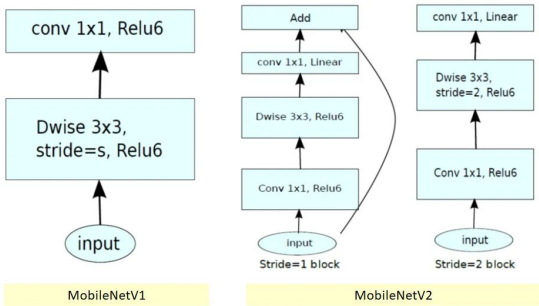


# Détail de la chaîne de pré-traitement

- Architecture du modèle (sélection des couches → *pooling average*)
- Distributed Featurization (*broadcast weights*)



1. Architecture MobileNet v2
2. Étapes de traitement
  - a. Conversion des images en matrice
  - b. Prédiction et aplatissement des outputs
  - c. Pandas UDF
  - d. Stockage des outputs au format parquet



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	-	-

- **Conversion des images** en matrice (img\_to\_array from Keras)
- **Features Extraction** - Pandas UDF

```
def preprocess(content):  
    """  
    Preprocesses raw image bytes for prediction.  
    """  
    img = Image.open(io.BytesIO(content)).resize([224, 224])  
    arr = img_to_array(img)  
    return preprocess_input(arr)  
  
def featurize_series(model, content_series):  
    """  
    Featurize a pd.Series of raw images using the input model.  
    :return: a pd.Series of image features  
    """  
    input = np.stack(content_series.map(preprocess))  
    preds = model.predict(input)  
    # For some layers, output features will be multi-dimensional tensors.  
    # We flatten the feature tensors to vectors for easier storage in Spark DataFrames.  
    output = [p.flatten() for p in preds]  
    return pd.Series(output)  
  
@pandas_udf('array<float>', PandasUDFType.SCALAR_ITER)  
def featurize_udf(content_series_iter):  
    """  
    This method is a Scalar Iterator pandas UDF wrapping our featurization function.  
    The decorator specifies that this returns a Spark DataFrame column of type ArrayType(FloatType).  
  
    :param content_series_iter: This argument is an iterator over batches of data, where each batch  
                                is a pandas Series of image data.  
    """  
    # With Scalar Iterator pandas UDFs, we can load the model once and then re-use it  
    # for multiple data batches. This amortizes the overhead of loading big models.  
    model = model_fn()  
    for content_series in content_series_iter:  
        yield featurize_series(model, content_series)
```

# Détail de la chaîne de pré-traitement

1. Architecture MobileNet v2
2. Étapes de traitement

- **Prédiction et aplatissement** des outputs (format Apache Parquet)
- **PCA** - Pipeline pyspark.ml

```
def update_schema(df):  
    # Define a UDF to convert the features column to a list  
    to_list_udf = udf(lambda row: [float(i) for i in row], ArrayType(FloatType()))  
  
    # Convert the features column to a list and create a new column "features_list"  
    features_df = features_df.withColumn("features_list", to_list_udf(features_df["features"]))  
  
    # Define UDF to convert list to DenseVector  
    to_dense_vector_udf = udf(lambda x: Vectors.dense(x), VectorUDT())  
  
    # Convert the "features_list" column to DenseVector and create a new column "features_vec"  
    features_df = features_df.withColumn("features_vec", to_dense_vector_udf(features_df["features_list"]))  
  
    # Drop the "features_list" column  
    features_df = features_df.drop("features_list")  
  
    return features_df  
  
def scaler(df):  
    scaler = StandardScaler(  
        inputCol = 'features_vec',  
        outputCol = 'scaledFeatures',  
        withMean = True,  
        withStd = True  
    ).fit(features_df)  
  
    # when we transform the dataframe, the old  
    # feature will still remain in it  
    features_df = scaler.transform(features_df)  
  
    return features_df  
  
def PCA(df):  
    n_components = 2  
    pca = PCA(  
        k = n_components,  
        inputCol = 'scaledFeatures',  
        outputCol = 'pcaFeatures'  
    ).fit(df_scaled)  
  
    df_pca = pca.transform(df_scaled).select('path', 'label', 'pcaFeatures')  
  
    return df_pca  
  
def export_Parquet(df, export_path):  
    df_pca.write.mode("overwrite").parquet(PATH_Result)  
    df_pca = pd.read_parquet(PATH_Result, engine='pyarrow')  
  
    return df_pca
```

# Sommaire

---

## 1. Choix techniques retenus

- a. Architecture Big Data et passage à l'échelle
  - i. Briques d'architectures nécessaires pour mettre en place un environnement big Data
  - ii. Identification des opérations critiques lors d'un passage à l'échelle
- b. Solution de transfer learning
- c. Performance du calcul distribué (*pyspark*)

## 2. Mise en oeuvre

- a. Test de la chaîne de traitement en local
- b. Déploiement d'une architecture Big Data dans le Cloud *AWS*
  - i. Conformité au RGPD (stockage et traitement sur des serveurs européens)
  - ii. Téléchargement des données sur *S3*
  - iii. Infrastructure Hadoop (*Cluster EMR*) - Gestion de machine virtuels *EC2*
  - iv. Lancement du notebook de pré-traitement via *JupyterHub*

## 3. Détail de la chaîne de pré-traitement

- a. Architecture *MobileNet v2*
- b. Étapes de traitement
  - i. Conversion des images en matrice (*img\_to\_array* from *Keras*)
  - ii. Prédiction et aplatissement des outputs
  - iii. *Pandas* UDF (*Spark Dataframe* & *yield* keyword)
  - iv. Stockage des outputs au format *Apache parquet*

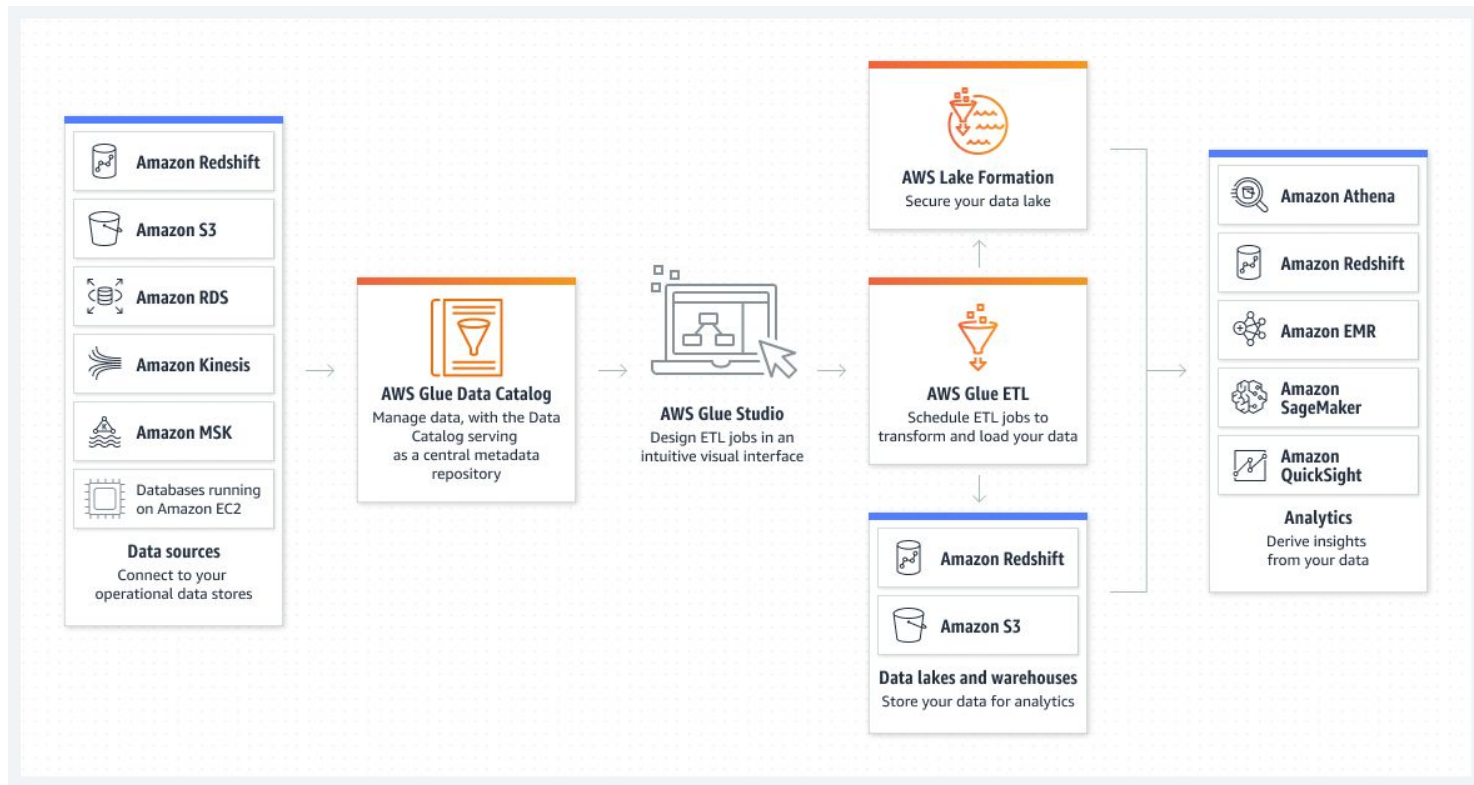
## 4. Conclusion (points d'attention et retour d'expérience)

## ➤ Introduction aux enjeux de *Cloud Computing*

- **Coûts**
- **Large gamme** de services (e.g 175 Services chez AWS)
  - **AWS Sage Maker** : Plateforme Machine Learning
  - **AWS Well-Architeched Framework** : checklist de bonnes pratiques cloud, autour de 5 piliers :
    - Excellence opérationnelle (execution / supervision des systèmes)
    - Sécurité (confidentialité / intégrité des données)
    - Fiabilité (tolérance aux pannes : systèmes distribués, planification des récupérations)
    - Efficacité des performances (optimisation de l'allocation des puissances de calcul)
    - Optimisation des coûts (analyse des dépenses, et contrôle de l'allocation des fonds)
  - **AWS GovCloud (US)** : exigence stricte de conformité des données sensibles hébergées aux US, ou de citoyens US (Ministère de la défense etc.)

## ➤ Solutions No-code de Data Pipeline ([Databricks](#))

## ➤ AWS Glue



➤ Résultats - Features extraction

aws

Services

Rechercher

[Alt+S]

Amazon S3

Compartiments

Points d'accès

Points d'accès de l'objet Lambda

Points d'accès multi-région

Opérations par lot

IAM Access Analyzer pour S3

Paramètres de blocage de l'accès public pour ce compte

Storage Lens

Tableaux de bord

Paramètres AWS Organizations

Fonctionnalité spot

AWS Marketplace pour S3

Amazon S3 > Compartiments > oc-p8-bucket > Results/

Results/

Copier l'URI S3

ObjetsPropriétés

Objets (25)  
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

↻

Copier l'URI S3

Copier l'URL

Télécharger

Ouvrir

Supprimer

Actions

Créer un dossier

Charger

Rechercher des objets en fonction du préfixe

	Nom	Type	Dernière modification	Taille	Classe de stockage
<input type="checkbox"/>	_SUCCESS	-	12 Mar 2023 08:19:13 PM CET	0 o	Standard
<input type="checkbox"/>	part-00000-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:14:00 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00001-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:14:00 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00002-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:14:28 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00003-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:14:28 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00004-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:14:57 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00005-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:14:57 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00006-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:15:25 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00007-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:15:24 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00008-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:15:53 PM CET	3.4 Mo	Standard
<input type="checkbox"/>	part-00009-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:15:53 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00010-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:16:21 PM CET	3.5 Mo	Standard
<input type="checkbox"/>	part-00011-a6d16c28-e62b-4064-8b8b-52aa80bf3404-c000.snappy.parquet	parquet	12 Mar 2023 08:16:21 PM CET	3.5 Mo	Standard