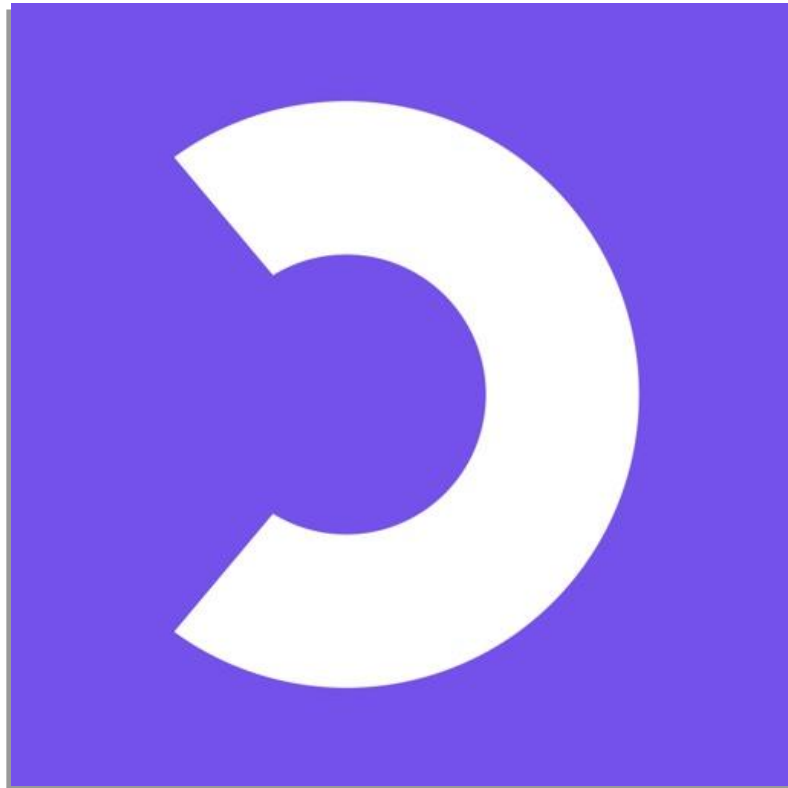
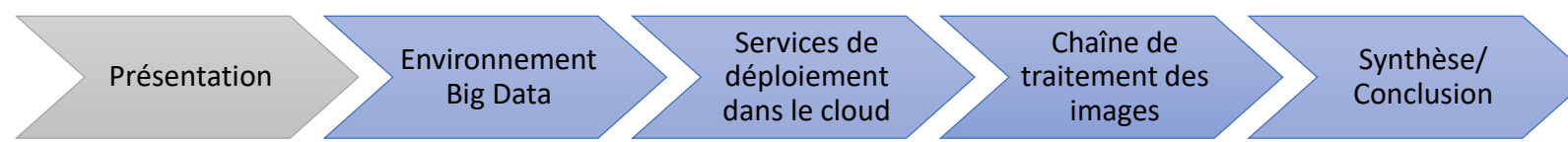


# Projet 8 : Déployez un modèle dans le cloud

---

Eva Rondeau





# Présentation

## 1. Contexte

- Start-up de l'AgriTech : propose des solutions innovantes pour la récolte de fruits
- Application : informations sur le fruit en question pour l'utilisateur

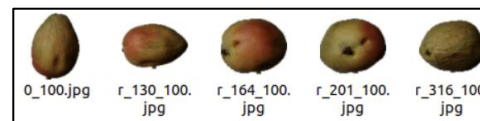


## 2. Objectifs

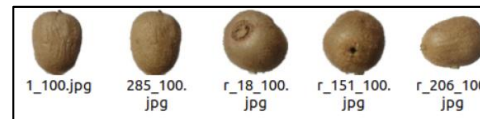
- Première version du moteur de classification des images des fruits
- Première version de l'architecture Big Data nécessaire

## 3. Données

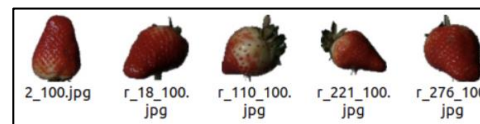
Total : **90 483** images (format .jpg)  
131 catégories différentes  
67 692 images (entraînement)  
22 688 images (test)  
Format : 100 x 100 pixels  
Rotation à 360° sur 3 axes



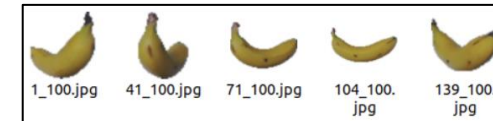
Papaya



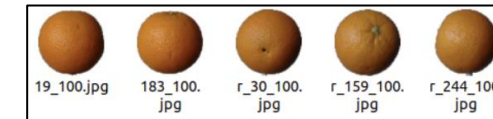
Kiwi



Strawberry

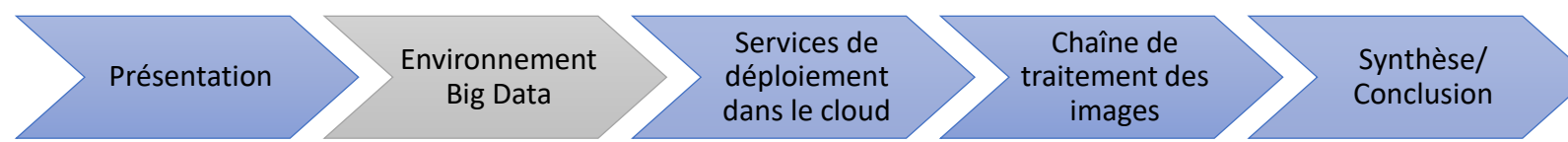


Banana



Orange

**25 images à traiter** (5 images pour 5 catégories de fruits)



## Environnement Big Data

**Volume de données amené à augmenter très rapidement après livraison.**







### Utilisation du cloud AWS



- AWS : Amazon Web Service
- Plateforme cloud
- Services divers : calcul, stockage, base de données, analyse de données, ...

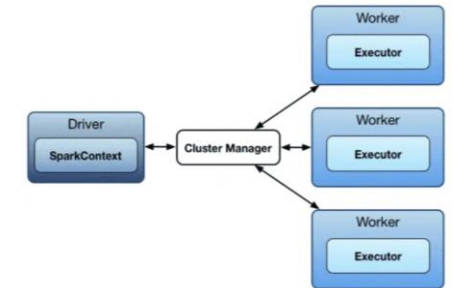
#### Services utilisés :

-  IAM (Identity Access Management) : gestion des utilisateurs et leur rôle avec des autorisations spécifiques
-  Amazon S3 (Simple Storage Service) : stockage de données
-  EC2 (Elastic Compute Cloud) : lancer instances de machines virtuelles dans le cloud et exécuter des tâches de calcul
-  EMR (Elastic MapReduce) : traitement distribué de données massives

### Développement des scripts en PySpark



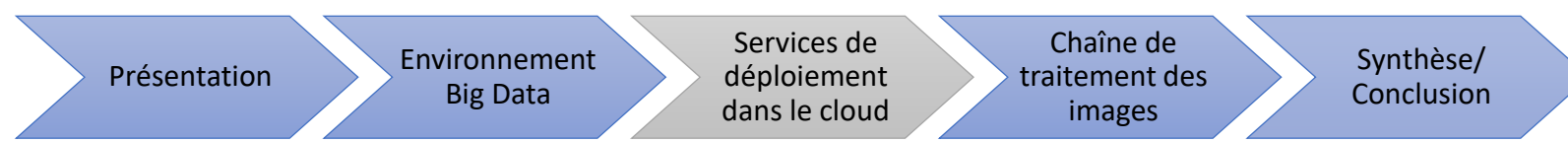
- Framework : traitement et analyse de grands volumes de données
- Traitement de données sur un cluster de machines (plusieurs ordinateurs)



[L'architecture du Framework Spark \(meritis.fr\)](http://meritis.fr)



- API python de Spark
- Utilise les fonctionnalités de Spark à partir de programmes Python
- Langage Spark : scala



# Services de déploiement de Spark dans le cloud

## 1. AWS Identity Access Management (IAM)

**Utilisateurs (1)** [Infos](#)

Un utilisateur IAM est une identité avec des informations d'identification à long terme utilisées pour interagir avec AWS dans un compte.

Rechercher des utilisateurs par nom d'utilisateur ou clé d'accès

[Ajouter des utilisateurs](#)

|                          | Nom d'utilisateur | Groupes | Dernière ac...  | MFA   | Âge du mot de p... | Âge des clés activ... |
|--------------------------|-------------------|---------|-----------------|-------|--------------------|-----------------------|
| <input type="checkbox"/> | userproject       | Aucun   | Il y a 13 jours | Aucun | Il y a 13 jours    | Il y a 13 jours       |

→

| <input type="checkbox"/> | Nom de la politique | Type           | Attaché via |
|--------------------------|---------------------|----------------|-------------|
| <input type="checkbox"/> | AmazonS3FullAccess  | Gérées par AWS | Directement |

## 2. Création paire de clés

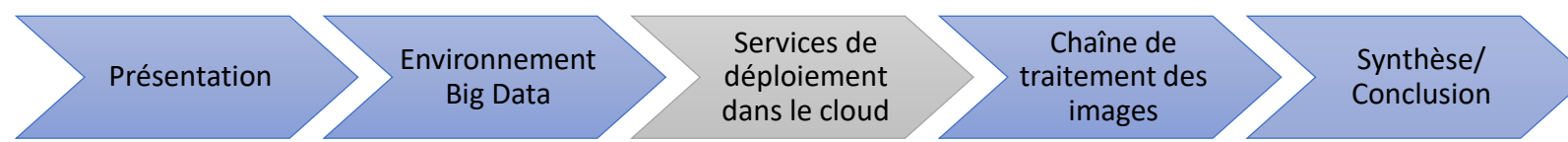
**Paires de clés (1)** [Informations](#)

Recherche

[Créer une paire de clés](#)

|                          | Nom            | Type | Créé                   | Empreinte digitale                          | ID                    |
|--------------------------|----------------|------|------------------------|---|-----------------------|
| <input type="checkbox"/> | EC2-p8-keypair | rsa  | 2023/07/21 17:50 GMT+2 | a9:3d:31:fa:6e:5c:0b:6f:fc:ac:d1:25:b4:a... | key-0a03e5f26f2ad49dd |

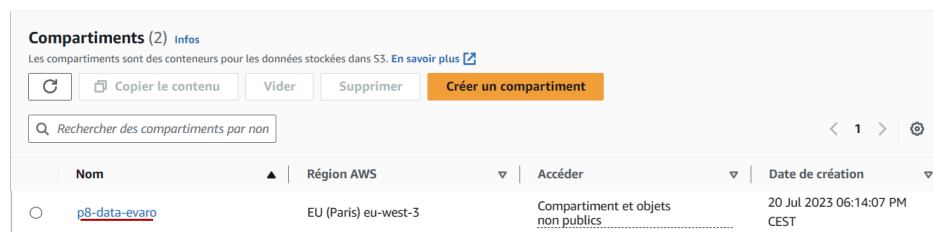
Accès SSH aux futurs serveurs EC2



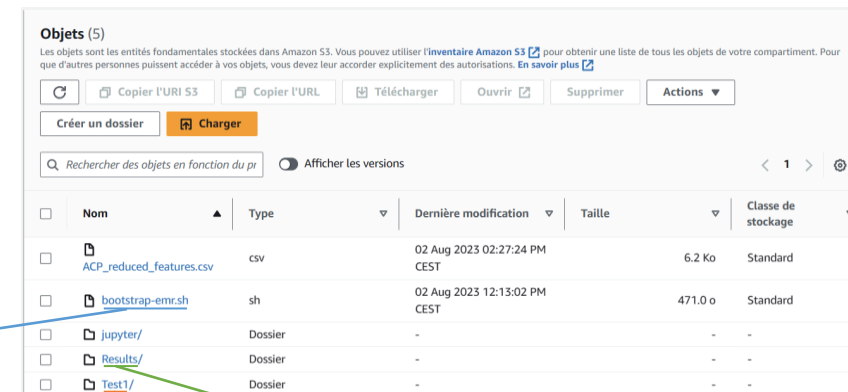
## Services de déploiement de Spark dans le cloud

### 3. AWS Simple Storage Service (S3)

#### 1) Création d'un bucket sur S3



#### 2) Stockage des données dans le bucket S3



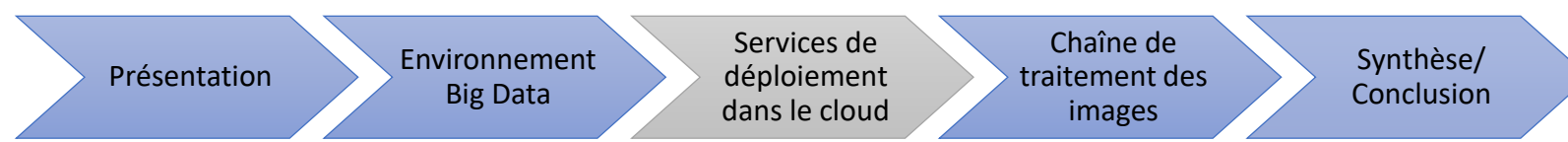
```
#!/bin/bash
sudo python3 -m pip install -U setuptools
sudo python3 -m pip install -U pip
sudo python3 -m pip install wheel
sudo python3 -m pip install pillow
sudo python3 -m pip install pandas==1.2.5
sudo python3 -m pip install pyarrow
sudo python3 -m pip install boto3
sudo python3 -m pip install s3fs
sudo python3 -m pip install fsspec
sudo python3 -m pip install tensorflow
sudo python3 -m pip install matplotlib
```

Installation des  
packages et  
bibliothèques Python

25 images de test

| <input type="checkbox"/> | Nom         | Type    |
|--------------------------|-------------|---------|
| <input type="checkbox"/> | Banana/     | Dossier |
| <input type="checkbox"/> | Kiwi/       | Dossier |
| <input type="checkbox"/> | Orange/     | Dossier |
| <input type="checkbox"/> | Papaya/     | Dossier |
| <input type="checkbox"/> | Strawberry/ | Dossier |

Features obtenues après exécution  
du modèle de traitement d'images  
(Parquet)



# Services de déploiement de Spark dans le cloud

## 4. AWS EMR sur EC2

1) Lancement cluster EMR : exécution tâches de traitement de données massives

Cluster : p8-data-fruits-evaro-2 En attente Cluster ready to run steps.

Récapitulatif

Historique de l'application

Surveillance

Matériel

Configurations

Événements

Étapes

Actions d'amorçage

Récapitulatif

ID : j-tit3rGEHWN8R

Date de création : 03-08-2023 16:37 (UTC+2)

Temps écoulé : 7 minutes

Résiliation automatique : Cluster waits

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

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

DNS public principal : ec2-13-39-105-206.eu-west-3.compute.amazonaws.com [Connect to the Master Node Using SSH](#)

Service d'historique : [Spark history server, YARN timeline server](#)

Connexions : [Not Enabled](#) [Activer la connexion Web](#)

Détails de configuration

Étiquette de version : emr-6.9.0

Distribution Hadoop : Amazon 3.3.3

Applications : JupyterHub 1.4.1, Spark 3.3.0

URI de connexion : s3://aws-logs-424586539406-eu-west-3/elasticmapreduce/ [📄](#)

Vue cohérente EMRFS : Désactivé

ID d'AMI personnalisée : --

Version d'Amazon Linux : 2.0.20230628.0 [En savoir plus](#) [🔗](#)

Application user interfaces

Zone de disponibilité : eu-west-3c

ID de sous-réseau (subnet) : [subnet-020af9cdbc458a2ea](#) [🔗](#)

Maître : [Action d'amorçage](#) 1 m5.xlarge

Principal : [Action d'amorçage](#) 2 m5.xlarge

Tâche : --

Cluster scaling : Not enabled

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

Sécurité et accès

Nom de clé : EC2-p8-keypair

Profil d'instance EC2 : EMR\_EC2\_DefaultRole

Rôle EMR : EMR\_DefaultRole

Auto Scaling role: EMR\_AutoScaling\_DefaultRole

Visible pour tous les utilisateurs : Tous [Modification](#)

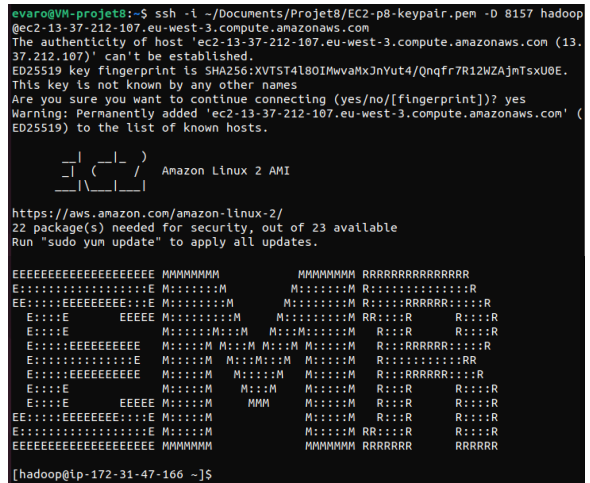
Groupes de sécurité pour le principal : [sg-0220b19bba9391847](#) [🔗](#) (ElasticMapReduce-principal : master)

Groupes de sécurité pour la base et les tâches : [sg-0a9322ec28d05f2c1](#) [🔗](#) (ElasticMapReduce-slave)

2) Activation de la connexion web

Connexion à l'instance EC2 maître du cluster en utilisant le tunnel SSH

```
ssh -i ~/EC2-p8-keypair.pem -ND 8157 hadoop@ec2-13-37-212-107.eu-west-3.compute.amazonaws.com
```



- Accès à l'environnement du cluster
- Lancement de l'application JupyterHub et exécution du script PySpark



Application user interfaces

Persistent user interfaces [🔗](#) [Spark history server, YARN timeline server](#)

On-cluster user [HDFS Name Node, Spark History Server, interfaces](#) [🔗](#) [JupyterHub, Resource Manager](#)

# Chaîne de traitement des images

## 1. Définition des chemins (PATH)



```
PATH = 's3://p8-data-evaro'
PATH_Data = PATH+'/Test1'
PATH_Result = PATH+'/Results'
print('PATH:'+\\
      PATH+'\\nPATH_Data:'+\\
      PATH_Data+'\\nPATH_Result:'+PATH_Result)
```

```
PATH:s3://p8-data-evaro
PATH_Data:s3://p8-data-evaro/Test1
PATH_Result:s3://p8-data-evaro/Results
```

Compartiments (2) [Infos](#)

Les compartiments sont des conteneurs pour les données stockées dans S3. [En savoir plus](#)

[Copier le contenu](#) [Vider](#) [Supprimer](#) [Créer un compartiment](#)

Rechercher des compartiments par nom

| Nom           | Région AWS           | Accéder                            | Date de création             |
|---------------|----------------------|------------------------------------|------------------------------|
| p8-data-evaro | EU (Paris) eu-west-3 | Compartiment et objets non publics | 20 Jul 2023 06:14:07 PM CEST |



Objets (5)

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 pr ☐ Afficher les versions

| Nom                      | Type    | Dernière modification        | Taille  | Classe de stockage |
|--------------------------|---------|------------------------------|---------|--------------------|
| ACP_reduced_features.csv | csv     | 02 Aug 2023 02:27:24 PM CEST | 6.2 Ko  | Standard           |
| bootstrap-emr.sh         | sh      | 02 Aug 2023 12:13:02 PM CEST | 471.0 o | Standard           |
| jupyter/                 | Dossier | -                            | -       | -                  |
| Results/                 | Dossier | -                            | -       | -                  |
| Test1/                   | Dossier | -                            | -       | -                  |

## 2. Chargement des données images

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

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

- Lecture au format binaire (extension .jpg)
- Lecture des fichiers compris dans les sous-dossiers de « Test1 »

```
path|modificationTime|length|conte
nt|
+---+
[s3://p8-data-evar...|2023-07-21 11:29:21| 5608|[FF D8 FF E0 00
1...|
[s3://p8-data-evar...|2023-07-21 11:29:21| 5484|[FF D8 FF E0 00
1...|
[s3://p8-data-evar...|2023-07-21 11:29:21| 5420|[FF D8 FF E0 00
1...|
[s3://p8-data-evar...|2023-07-21 11:29:21| 5397|[FF D8 FF E0 00
1...|
[s3://p8-data-evar...|2023-07-21 11:29:21| 5375|[FF D8 FF E0 00
1...|
+---+
only showing top 5 rows
```

## Chaîne de traitement des images

### 3. Création d'une colonne « label »



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

- Division de la chaîne de caractère de la colonne « path »
- Récupération de l'élément à la position -2

```
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://p8-data-evaro/Test1/Strawberry/2_100.jpg|Strawberry|
|s3://p8-data-evaro/Test1/Orange/r_159_100.jpg|Orange|
|s3://p8-data-evaro/Test1/Orange/19_100.jpg|Orange|
|s3://p8-data-evaro/Test1/Orange/r_30_100.jpg|Orange|
|s3://p8-data-evaro/Test1/Orange/183_100.jpg|Orange|
+-----+-----+
only showing top 5 rows
```

### 4. Préparation du modèle

```
def model_fn():
    """
    Returns a MobileNetV2 model with top layer removed
    and broadcasted pretrained weights.
    """
    model = MobileNetV2(weights='imagenet',
                        include_top=True,
                        input_shape=(224, 224, 3))
    for layer in model.layers:
        layer.trainable = False
    new_model = Model(inputs=model.input,
                      outputs=model.layers[-2].output)
    new_model.set_weights(broadcast_weights.value)
    return new_model
```

- Modèle MobileNetV2 pré-entraîné sur le dataset ImageNet
- Images au format 224 x 224 pixels
- Dernière couche de classification retirée
- Diffusion des poids à l'ensemble des nœuds du cluster



# Chaîne de traitement des images

## 5. Modèle MobileNetV2



[Submitted on 13 Jan 2018 (v1), last revised 21 Mar 2019 (this version, v4)]

### MobileNetV2: Inverted Residuals and Linear Bottlenecks

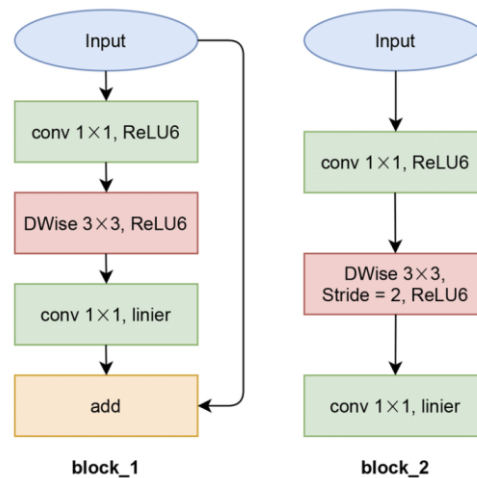
Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen

In this paper we describe a new mobile architecture, MobileNetV2, that improves the state of the art performance of mobile models on multiple tasks and benchmarks as well as across a spectrum of different model sizes. We also describe efficient ways of applying these mobile models to object detection in a novel framework we call SSDLite. Additionally, we demonstrate how to build mobile semantic segmentation models through a reduced form of DeepLabv3 which we call Mobile DeepLabv3.

The MobileNetV2 architecture is based on an inverted residual structure where the input and output of the residual block are thin bottleneck layers opposite to traditional residual models which use expanded representations in the input. MobileNetV2 uses lightweight depthwise convolutions to filter features in the intermediate expansion layer. Additionally, we find that it is important to remove non-linearities in the narrow layers in order to maintain representational power. We demonstrate that this improves performance and provide an intuition that led to this design.

Finally, our approach allows decoupling of the input/output domains from the expressiveness of the transformation, which provides a convenient framework for further analysis. We measure our performance on Imagenet classification, COCO object detection, VOC image segmentation. We evaluate the trade-offs between accuracy, and number of operations measured by multiply-adds (MAdd), as well as the number of parameters

- Modèle d'apprentissage pour la classification d'images
- Récent : publié en 2018
- Rapidité d'exécution (gros volumes d'images)
- Entraîné sur plus d'1 million d'images
- Base de données : ImageNet



- 2 types de blocs : bloc résiduel et bloc de réduction
- Bloc résiduel :
  - 1<sup>ère</sup> couche de convolution 1x1 avec ReLU6
  - 2<sup>ème</sup> couche de convolution en profondeur
  - 3<sup>ème</sup> couche de convolution 1x1 linéaire
- Bloc de réduction:
  - Réduction des dimensions spatiales
- Activation utilisées pour une couche dense (fully connected) avec 1280 neurones (classification finale)

# Chaîne de traitement des images

## 6. Redimensionnement images et extraction des features



```
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)

features_df = images.repartition(24).select(col("path"),
                                           col("label"),
                                           featurize_udf("content").alias("features")
                                           )
```

- Redimensionnement image de taille 224x224 pixels
- Conversion en tableau numpy et normalisation des valeurs de pixels de l'image

- Appel de la fonction précédente pour le pré-traitement
- Prédiction sur les images pré-traitées en utilisant le modèle pour extraire les features

- Colonne retournée de type ArrayType(FloatType)
- Chargement du modèle MobileNetV2
- Pour chaque lot (série de plusieurs images) : fonction « featurize\_series » appelée pour extraire les features

- Création dataframe Spark : sélection « path », « label » et « features » (featurize\_udf)

# Chaîne de traitement des images

## 7. Chargement données



```
features_df.write.mode("overwrite").parquet(PATH_Result)

df = pd.read_parquet(PATH_Result, engine='pyarrow')

df.loc[0, 'features'].shape

(1280,)
```

- Données enregistrées dans répertoire « Results » au format Parquet
- Chargement des résultats dans un dataframe Pandas « df »
- 1280 features extraites

## 8. Réduction de dimensions (ACP en PySpark)

```
from pyspark.ml.linalg import Vectors
df["features"] = df["features"].apply(lambda x: Vectors.dense(x))

# Conversion du dataframe Pandas en un dataframe Spark
spark_df = spark.createDataFrame(df)

from pyspark.ml.feature import PCA
from pyspark.ml.feature import StandardScaler

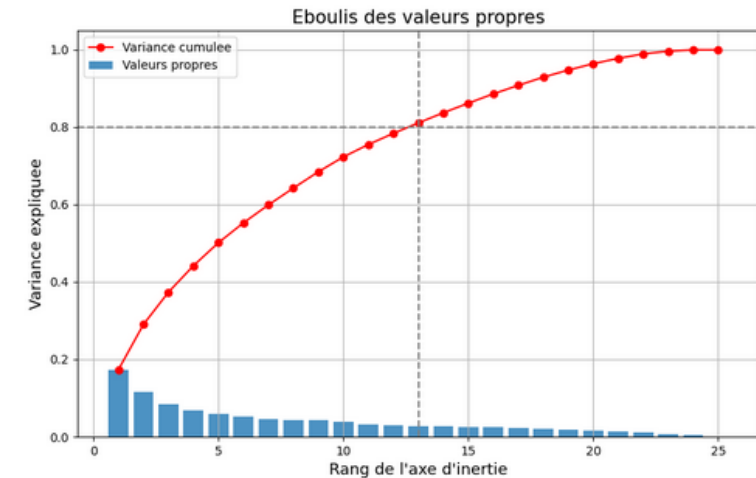
# Création de l'instance StandardScaler
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures")

# Application de l'instance et du transformeur résultant
scalerModel = scaler.fit(spark_df)
scaled_df = scalerModel.transform(spark_df)

# Création de l'instance PCA
pca = PCA(k=25, inputCol="scaledFeatures", outputCol="ACP_features")
model_pca = pca.fit(scaled_df)

# Valeurs propres
explained_var = model_pca.explainedVariance
cumulative_var = np.cumsum(explained_var)
```

- Conversion de la liste de « floats » en vecteurs denses.
- Conversion dataframe Pandas en dataframe Spark
- Normalisation des données avec StandardScaler()
- Création de l'ACP (k=25)
- Calcul de la variance expliquée cumulée



Nombre de composantes pour 80% de la variance expliquée: 13

# Chaîne de traitement des images



## 9. Sauvegarde des résultats en format .csv

```
# Nouvelle instance ACP
pca_new = PCA(k=position, inputCol="scaledFeatures", outputCol="ACP_reduced_features")
model_pca_new = pca_new.fit(scaled_df) # Appliquer le fit avec le nouveau modèle
result_pca_new = model_pca_new.transform(scaled_df)
```

```
results = result_pca_new.select("ACP_reduced_features")
results_pd = results.toPandas()
results_pd.to_csv(PATH + "/ACP_reduced_features.csv", index=False)
```

```
# Affichage du nombre de features
results_pd.loc[0, 'ACP_reduced_features'].shape
```

(13,)

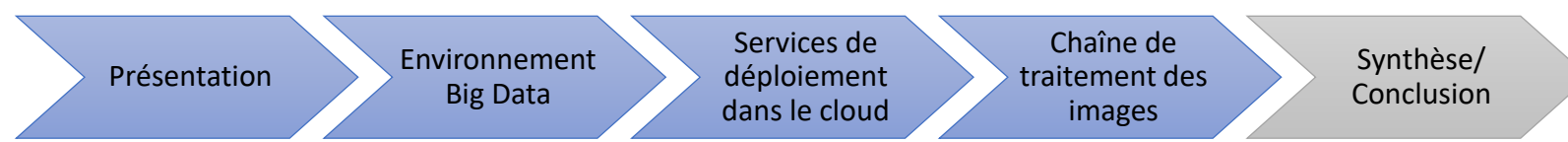
- Nouvelle instance en sélectionnant les composantes principales expliquant 80% de la variance.
- Enregistrement sur notre bucket S3 (format .csv)

Objets (5)

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](#)

☐ Afficher les versions

| <input type="checkbox"/> | Nom                      | Type    | Dernière modification        | Taille  | Classe de stockage |
|--------------------------|--------------------------|---------|------------------------------|---------|--------------------|
| <input type="checkbox"/> | ACP_reduced_features.csv | csv     | 02 Aug 2023 02:27:24 PM CEST | 6.2 Ko  | Standard           |
| <input type="checkbox"/> | bootstrap-emr.sh         | sh      | 02 Aug 2023 12:13:02 PM CEST | 471.0 o | Standard           |
| <input type="checkbox"/> | jupyter/                 | Dossier | -                            | -       | -                  |
| <input type="checkbox"/> | Results/                 | Dossier | -                            | -       | -                  |
| <input type="checkbox"/> | Test1/                   | Dossier | -                            | -       | -                  |



## Synthèse

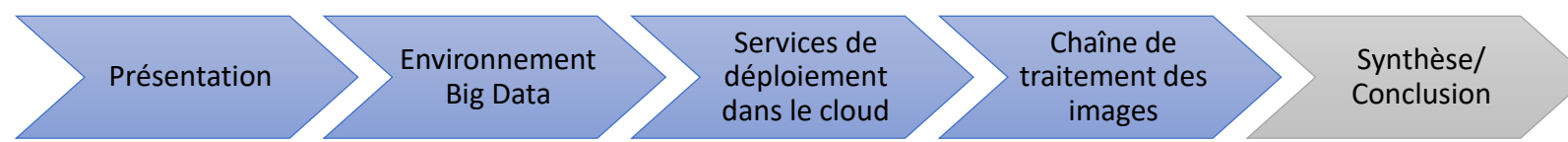
### Mise en place d'un environnement Big Data + déploiement modèle Machine Learning :

- Services proposés par AWS
- Respect conformité du RGPD : stockage des données et traitement sur des serveurs situés sur le territoire européen



- Spark et PySpark pour les opérations de calcul
- AWS IAM pour la gestion des utilisateurs et autorisations
- AWS S3 pour le stockage des données (25 images pour limiter les coûts)
- AWS EMR sur EC2 pour l'exécution d'application de traitement de données distribuées





# Conclusion

## Difficultés rencontrées



- Prise en main de PySpark
  - Découverte de l'environnement AWS
- Développement dans un environnement Linux Ubuntu
  - Installation d'une machine virtuelle

## Compétences acquises

- ✓ Identifier les outils du cloud permettant la mise en place d'un environnement Big Data
- ✓ Utilisation des outils du cloud afin de manipuler des données dans un environnement Big Data
  - ✓ Paralléliser des opérations de calcul avec PySpark

