2020

OPENCLASSROOMS, en partenariat avec CentraleSupélec

Parcours Data-Scientist – Projet 8

Mohamed Bouzid

DÉPLOYEZ UN MODÈLE DANS LE CLOUD

SOMMAIRE

- Problématique
- Quelles briques logicielles ?
- Stratégie de l'architecture big data
- Dataset
- Pre-processing
- Extraction des features
- PCA & résultats
- Conclusion et perspectives

PROBLEMATIQUE

PROBLEMATIQUE

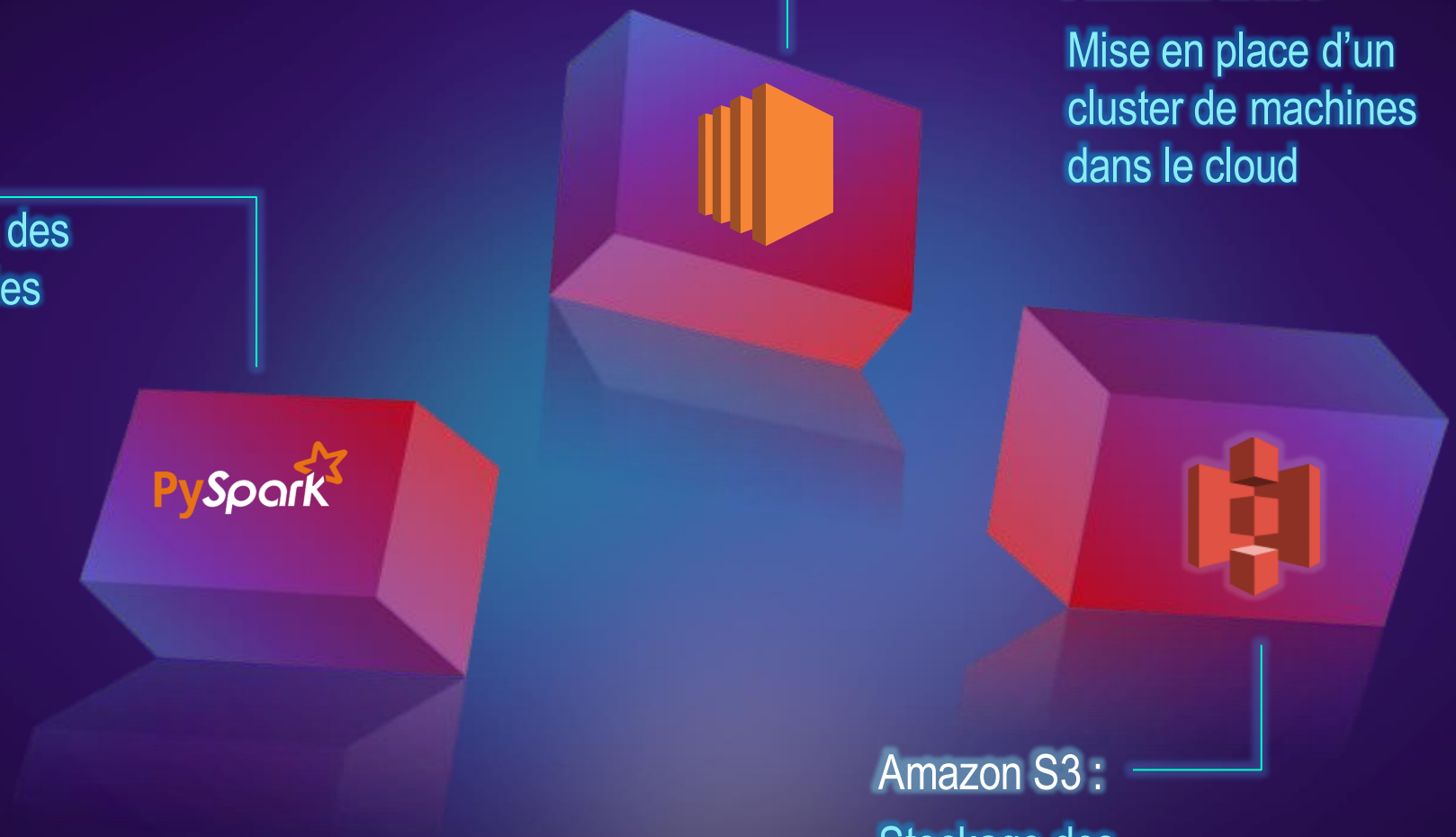
- **Historique :** « Fruits! » est une startup dont le projet à long terme est de vendre des robots cueilleurs. La première étape de leur stratégie consiste à mettre en place une app mobile de reconnaissance de fruits par l'image.

- **Problème :** le volume de données risque d'augmenter très rapidement.

- **Objectifs :**
  - construire une première version de l'architecture Big Data.
  - traiter des données : preprocessing et réduction de dimension

QUELLES BRIQUES LOGICIELLES ?

QUELLES BRIQUES LOGICIELLES ?

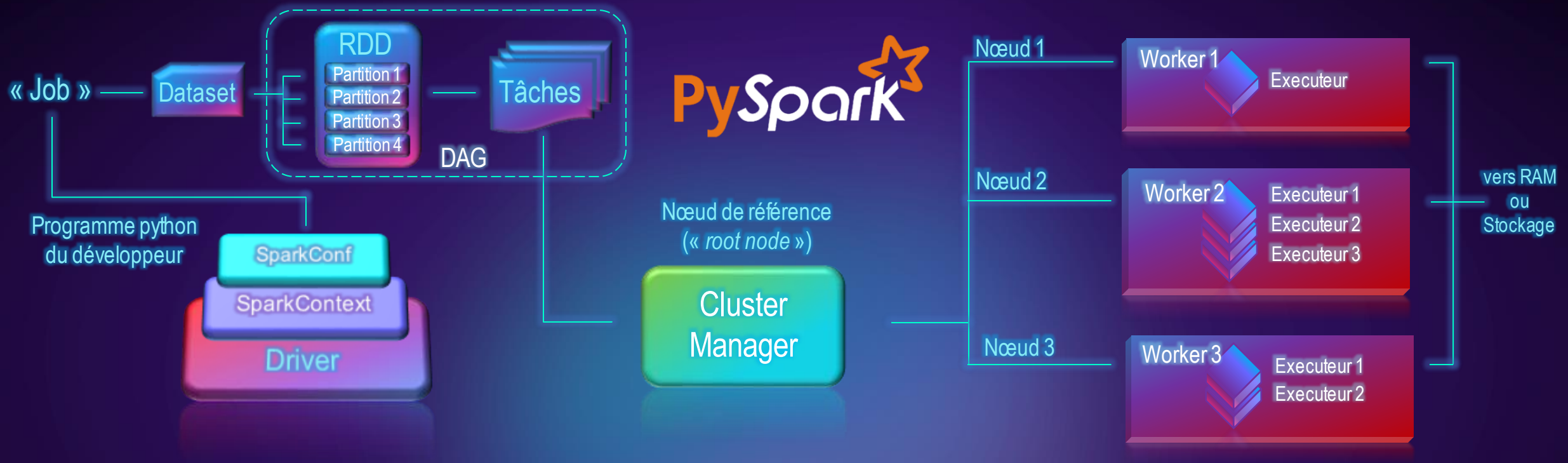


PySpark :
Parallélisation des calculs dans des clusters

Amazon EC2 :
Mise en place d'un cluster de machines dans le cloud

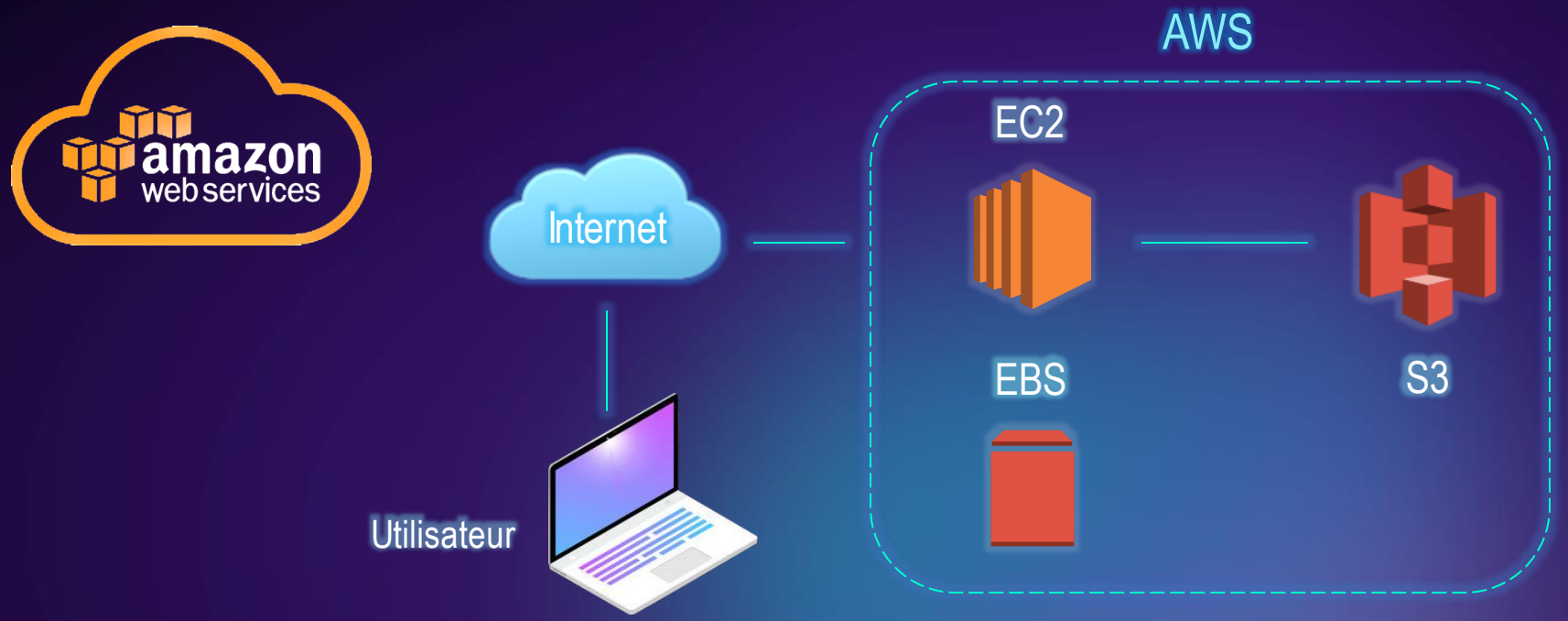
Amazon S3 :
Stockage des données dans le cloud

QUELLES BRIQUES LOGICIELLES ?



- Le **Driver** (ex-terme « *master* ») demande au Cluster manager des nœuds, et des Workers & exécuteurs disponibles pour le cluster :
  - **SparkConf** : dans le SparkContext, paramètres de configuration (du cluster, parallélisme, ressources allouées aux jobs, shuffle)
  - **SparkContext** : utilisé par le Driver pour soumettre au Cluster manager le « *job* » (calcul parallèle via des tâches (« *tasks* »))
  - **RDD (*Resilient Distributed Dataset*) :** partitionnement du dataset en nœuds stockés en mémoire (données disponibles pour réutilisation, évite de réitérer les calculs sur les mêmes données), et tolérance aux pannes
  - **DAG (*Directed Acyclic Graph*) :** convertit le dataset en tâches compréhensibles par le Cluster manager
- Le **Cluster manager** alloue les ressources des Workers et leur ordonne d'exécuter les tâches
- Le **Worker** (ex-terme « *slave* ») exécute les tâches au travers d'un ou plusieurs exécuteurs
- **L'exécuteur :** exécute une tâche (une tâche par exécuteur) puis la fait garder en mémoire ou en stockage

QUELLES BRIQUES LOGICIELLES ?



- **EC2 (Elastic Compute Cloud) :**
  - fournit un serveur virtuel appelé instance où l'on peut mettre en place un OS de départ (AMI), des applications, et des permissions (IAM).
  - **EBS (Elastic Block Storage) :** fournit des volumes de stockage de blocs persistants à EC2 (objectifs : protection contre les défaillances, performance à faible latence, évolution rapide des ressources)
  - La connexion à EC2 se fait en SSH avec une clé privée
  - L'offre gratuite d'EC2 est limitée (besoin de plus de ressources ?)
- **S3 (Simple Storage Service) :** permet aux utilisateurs de stocker dans un « *bucket* » et d'y récupérer différents types de données en utilisant des appels API, depuis n'importe quel emplacement sur Internet. Offre une durabilité, une disponibilité, des performances, une sécurité, et une scalabilité à échelle industrielle.

STRATÉGIE DE L'ARCHITECTURE BIG DATA

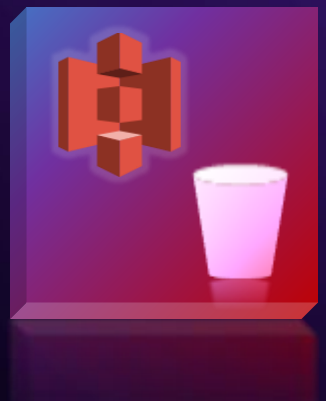
STRATÉGIE DE L'ARCHITECTURE BIG DATA

Utilisateur

SSH

AWS ID
AWS Key

AWS ID
AWS Key



Jupyter notebook:

- Ouverture des ports 8888
- Accès via mot de passe AWS
- Code python contenant le job Spark

AWS EC2 :

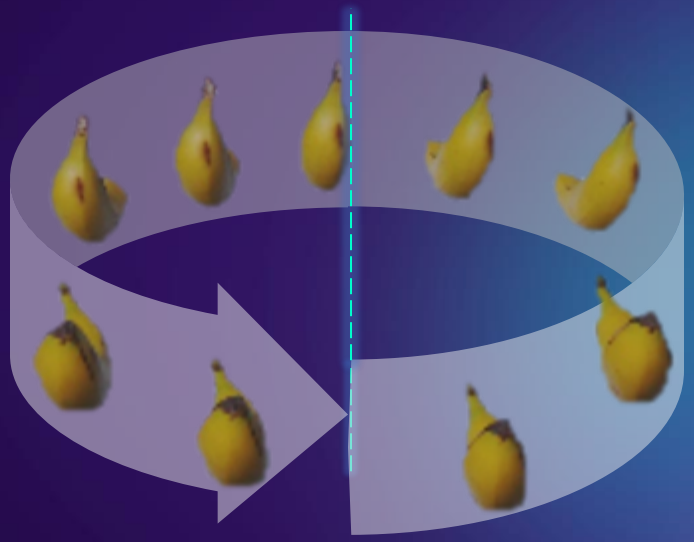
- Groupe de sécurité :
  - SSH ouverture du port 22 avec paire de clés
  - TCP : ouverture du port 8888 pour le notebook en « local EC2 »
  - Rôle IAM (Identity and Access Management) pour l'accès à S3
- Installation des packages pour le projet (Python 3, PySpark, OpenCV...)
- **Type d'instance EC2** : « t2large » (CPU : 2 cœurs, RAM : 8GB)
- **AMI** (Amazon Machine Image) : Amazon Linux 2 AMI (HVM, 64 bits x86)

AWS S3 :

- Téléversement des images (via l'interface graphique)
- Téléversement des features (format parquet, via script python)
- **Lien bucket** : s3://irelandprojet8/images/

DATASET

DATASET



Jeu de données :

- 90 483 images de fruits et légumes
- 131 variétés
- Training set (313 Mo) : ~ 500 images par variété
- Test set (103 Mo) : ~ 175 images par variété
- Taille des images : 100 x 100 pixels
- Une représentation 360° sur fond blanc
- Labélisation de chaque image

PRE-PROCESSING

PRE-PROCESSING

Image originale

Resizing

Réduction de bruits

Auto-contraste & égalisation

Flou Gaussien

Keras pre-processing

Extraction de features & réduction dimensionnelle

```python
# Resizing
resizing = cv2.resize(rgb_img, (224, 224))
```

```python
# Réduction de bruits
f_denoising = cv2.fastNlMeansDenoisingColored(resizing,None,10,10,7,21)
```

```python
#Auto-contraste & Egalisation
auto_C_E = np.array(ImageOps.equalize(ImageOps.autocontrast(Image.from
```

```python
#Flou Gaussien
gauss = cv2.GaussianBlur(auto_C_E,(5, 5),0)
```

```python
# Fonction de preprocessing du Resnet-50
arr = img_to_array(gauss)
arr_2 = preprocess_input(arr)
```

EXTRACTION DES FEATURES

EXTRACTION DES FEATURES

Rectified Linear Unit (ReLU) :

- Dans une image, la linéarité n'est pas très présente : ReLU remplace les pixels négatifs par des zéros (casse la linéarité des convolutions)
- Accélère les calculs

Pooling :

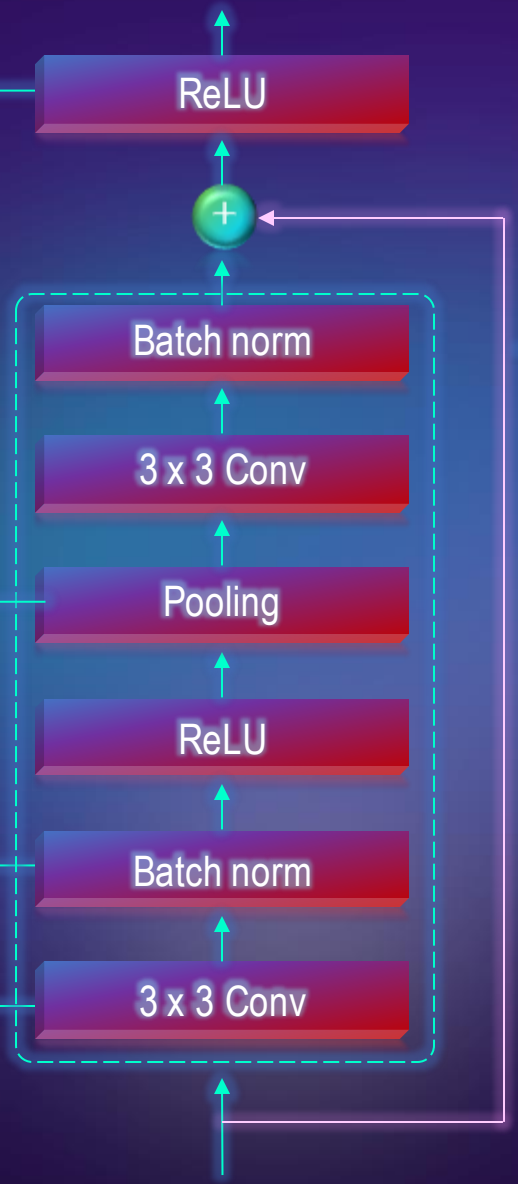
- Réduit une image en préservant les informations les plus importantes qu'elle contient

Batch normalization :

- Normalise les données des couches antérieures

Couche de Convolution :

- Filtre de feature recherchée dans une image
- Balaye pixel par pixel et applique le produit scalaire des pixels du filtre et des pixels balayés => opérations d'additions/multiplications => linéarité des valeurs en sortie par rapport à celles d'entrée



Architecture du réseau ResNet

Raccourci :

- Sauts de connexion caractéristique des réseaux de neurones résiduels (ResNet)
- Évite la disparition du gradient

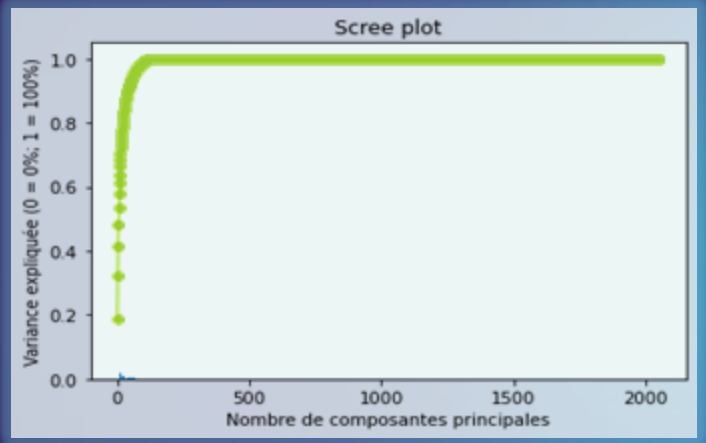
- Extraction des features via **ResNet50** (réseau de neurones convolutif « *résiduel* », i.e. avec raccourcis, constitué de 50 couches)
- « *Transfer-learning* » : modèle déjà pré-entraîné sur la base de données *ImageNet* contenant 14 millions d'images
- Requière en entrée une image en 224x224 pixels
- Aboutit en sortie à un vecteur de 2048 dimensions
- Modèle utilisé pour extraire des features seulement => on supprime la dernière couche (de classification)
- Implémenté dans la librairie *TensorFlow* (Google)

PCA & RÉSULTATS

PCA & RÉSULTATS

PCA :

- Principe : transformation des variables corrélées en «composantes principales» (dimensions)
- Avantages : facilité de calcul d'un grand dataset, visualisation possible
- Inconvénients : perte d'information
- Réussite : on observe une bonne séparation des fruits, avec d'une part, les fruits ronds et d'autre part, la banane
- PC1 = forme, PC2 = couleur ?

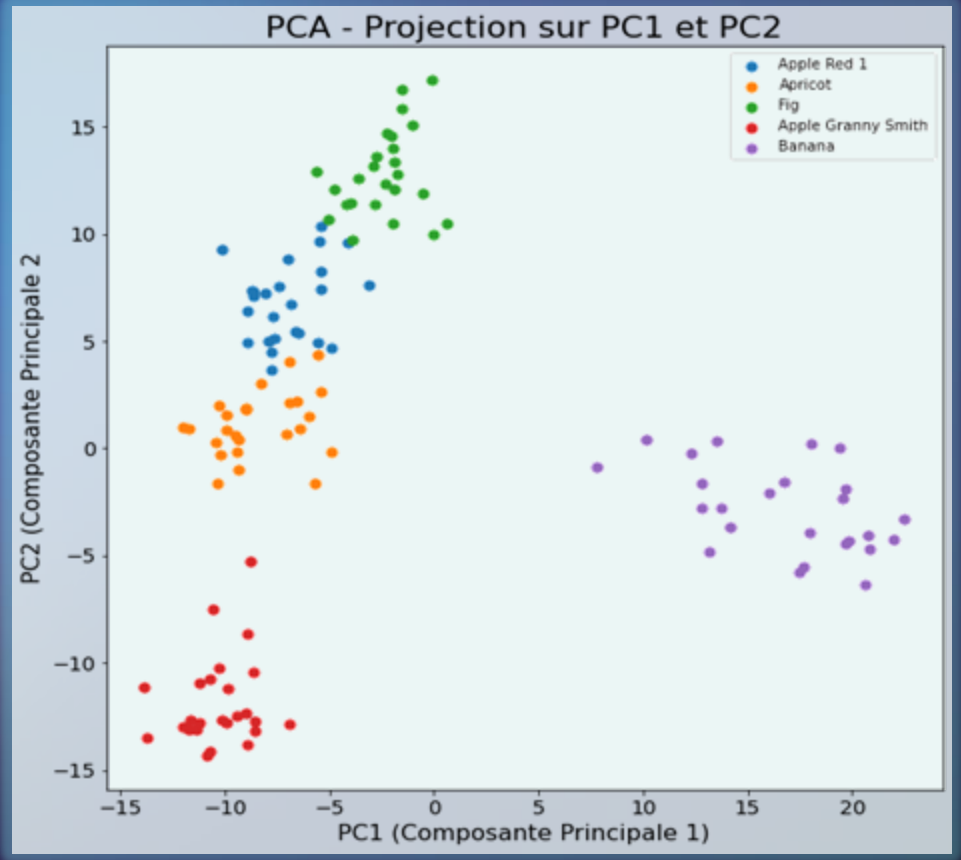



```
# Script pour trouver le nombre de dimensions optimales pour un seuil définit (ici 80%)

for i in range(50):
    a = scree.cumsum()[i]
    if a >= 0.8:
        print("{} composantes principales expliquent au moins 80% de la variance totale".format(i))
        break
```

```
19 composantes principales expliquent au moins 80% de la variance totale
```

Nom	Type	Dernière modification	Taille	Classe de stockage
features.parquet/	Dossier	-	-	-
images/	Dossier	-	-	-
P8_01_Notebook_AWS.ipynb	ipynb	27 Jan y 08:24:41 PM CET	466.6 Ko	Standard



CONCLUSION ET PERSPECTIVES

CONCLUSION ET PERSPECTIVES

Il a été effectué :

- Mise en place d'une architecture big data dans le cloud (AWS) :
  - Instance EC2 pour fournir un serveur virtuel
  - PySpark pour diriger les opérations d'extractions de features
  - Bucket S3 pour stocker les fichiers source/destination
- Utilisation d'un modèle de type réseau de neurones pour extraire les features
- Réduction dimensionelle avec le PCA (et indirectement Resnet–50)

Perspectives pour la société « Fruits! » :

- Réitérer l'expérience avec un volume plus conséquent, meilleure machine
- Reconsidérer une meilleur architecture intégrée pour s'affranchir des problèmes de compatibilité (EMR, SageMaker, …)