Réaliser un traitement dans un environnement Big Data sur le Cloud

14/05/2024

Introduction

- Data Scientist dans une très jeune start-up de l'AgriTech

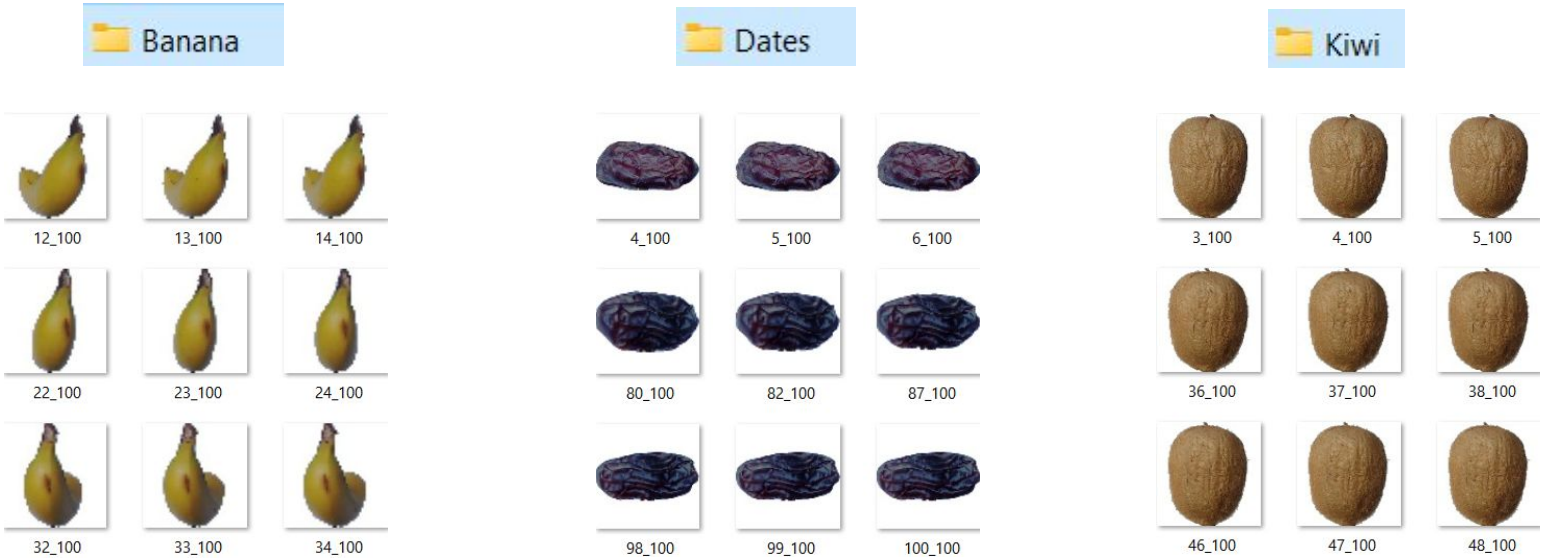
- Objectif: robot cueilleur de fruits, au préalable application mobile qui permettrait aux utilisateurs de prendre en photo un fruit et d'obtenir des informations sur ce fruit.

- Alternant a testé une première approche dans un environnement Big Data AWS EMR

- Mission:
  - Préparer la chaîne de traitement en local (mise à jour script pyspark avec ACP)
  - Mettre en place un environnement Big Data dans le Cloud (AWS)
  - Exécuter le traitement dans le Cloud

Jeu de données

- 22688 images de fruits réparties en 131 catégories

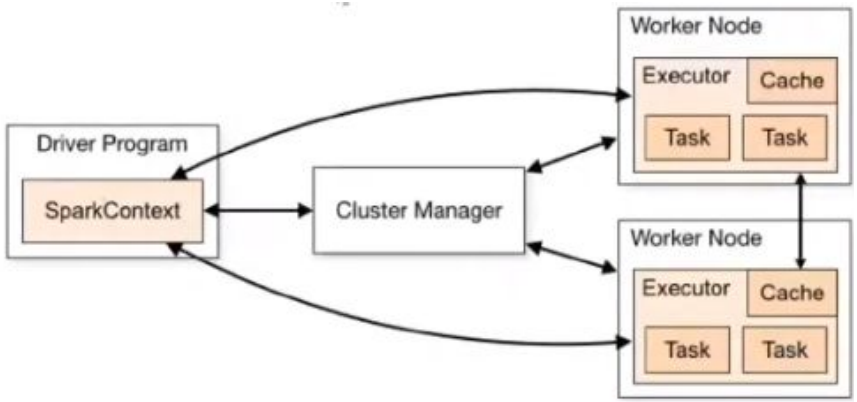


Partie 1: Processus de création de l’environnement Big Data, S3 et EMR

Environnement Big Data

- Volume de données va augmenter rapidement
  - Nécessité de déployer le traitement dans un environnement Big Data

- Spark:
  - Framework permettant de traiter des données massives en utilisant le **calcul distribué** (Pyspark permet d'utiliser Spark en Python, Pandas trop lent)
  - Corrige les limites de Hadoop (vitesse en utilisant la RAM)
  - Gestion de l'exécution des tâches



Choix du Cloud

- Cloud: AWS
  - Prestataire le plus connu, offre la plus large en cloud computing
  - **Louer de la puissance de calcul à la demande** (même si volume de données augmente fortement)
  - Baisse des coûts VS bail de location de serveurs de calcul

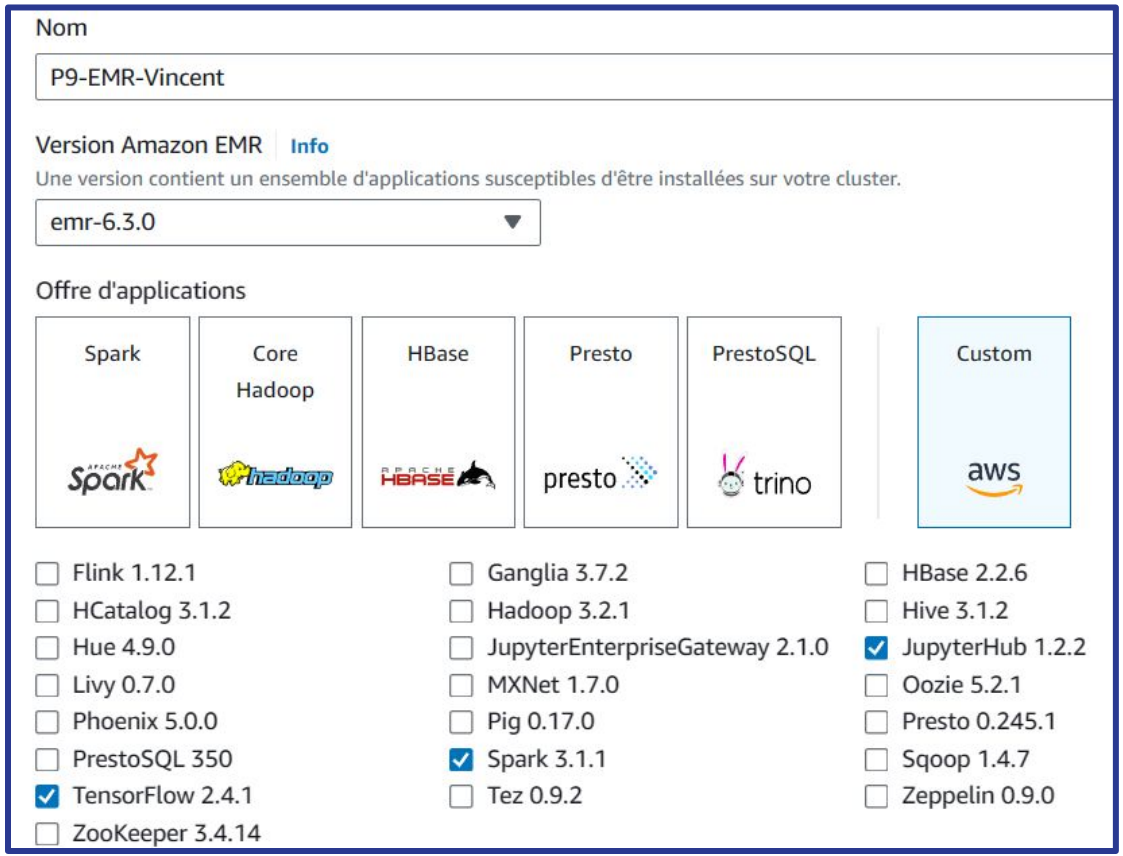
Choix du service EMR et du stockage S3

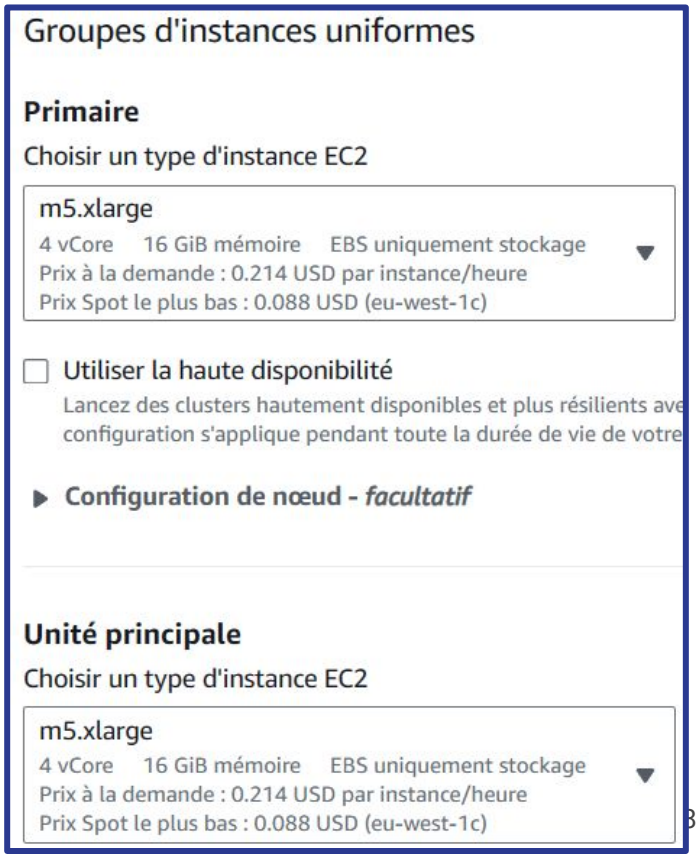
- **Service EMR**
  - Solution **PaaS** choisie vs IaaS (rapidité de mise en oeuvre)
  - Location d'instances EC2 avec applications pré-installées (**Spark**, ...)
  - Possibilité de demander installation de Tensorflow et JupyterHub

- **Stockage S3**
  - Inconvénients de stocker données sur EC2 (taille disque, sauvegarde, perte données si résiliation ...)
  - S3: espace disque illimité, et indépendant de EC2
  - Accès rapide et simple aux données (patch au format S3://...)

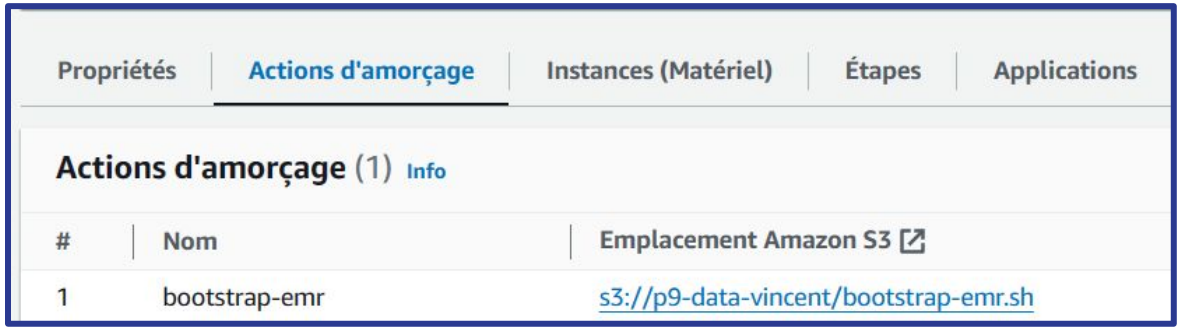
Solution	Proximité des serveurs	Persistance	Faible coût	Lecture aléatoire
EC2	✓	✗	✗	✓
EFS	✓	✓	✗	✓
S3	✓	✓	✓	✗
Glacier	✗	✓	✓	✗

EMR - Applications et groupes d’instances





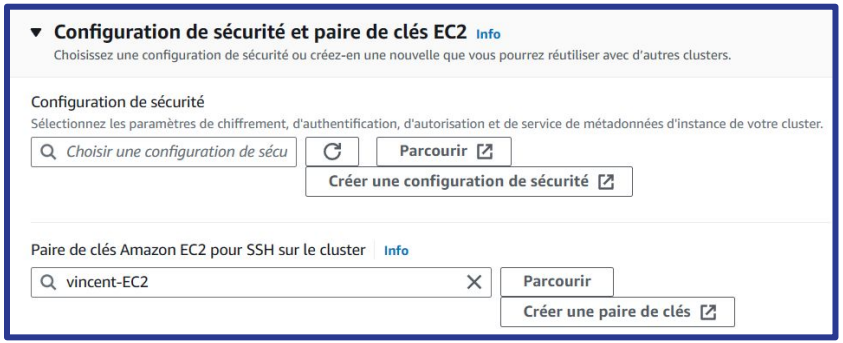
EMR - Bootstrap

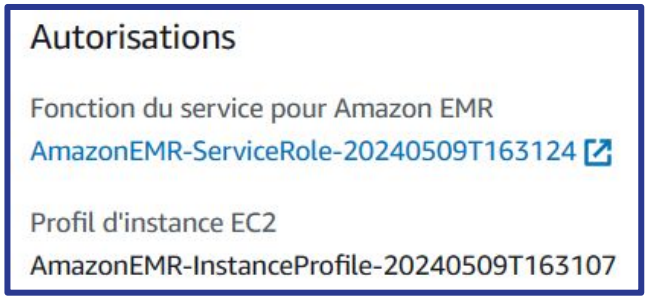



```
#!/bin/bash
sudo python3 -m pip install -U setuptools
sudo python3 -m pip install -U pip
sudo python3 -m pip install wheel
sudo python3 -m pip install pillow
sudo python3 -m pip install pandas==1.2.5
sudo python3 -m pip install matplotlib==3.4
sudo python3 -m pip install pyarrow
Ssudo python3 -m pip install boto3 s3fs
sudo python3 -m pip install fsspec
```

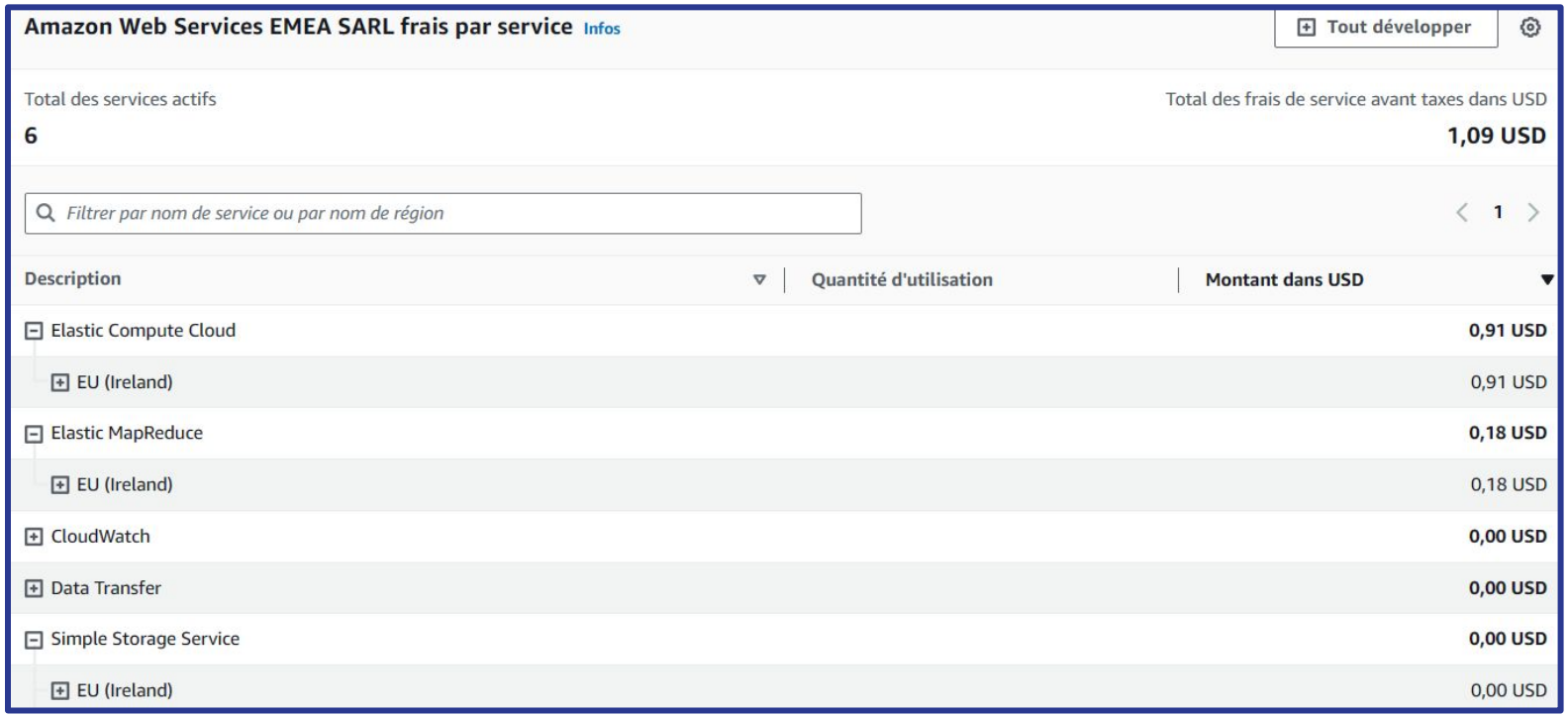
- Les actions d’amorçage sont exécutées avant l'installation des applications et avant qu'Amazon EMR ne commence à traiter les données.

clés EC2 pour SSH et autorisations (rôle IAM)





Respect RGPD



Amazon Web Services EMEA SARL frais par service Infos

Tout développer

Total des services actifs

6

Total des frais de service avant taxes dans USD

1,09 USD

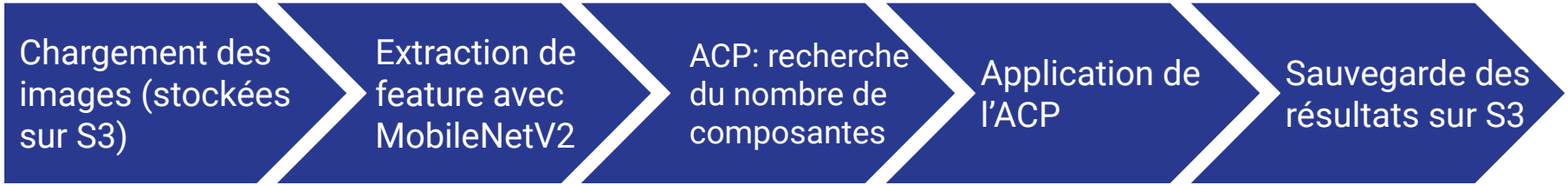
Filtrer par nom de service ou par nom de région

1

Description	Quantité d'utilisation	Montant dans USD
Elastic Compute Cloud		**0,91 USD**
EU (Ireland)		0,91 USD
Elastic MapReduce		**0,18 USD**
EU (Ireland)		0,18 USD
CloudWatch		**0,00 USD**
Data Transfer		**0,00 USD**
Simple Storage Service		**0,00 USD**
EU (Ireland)		0,00 USD

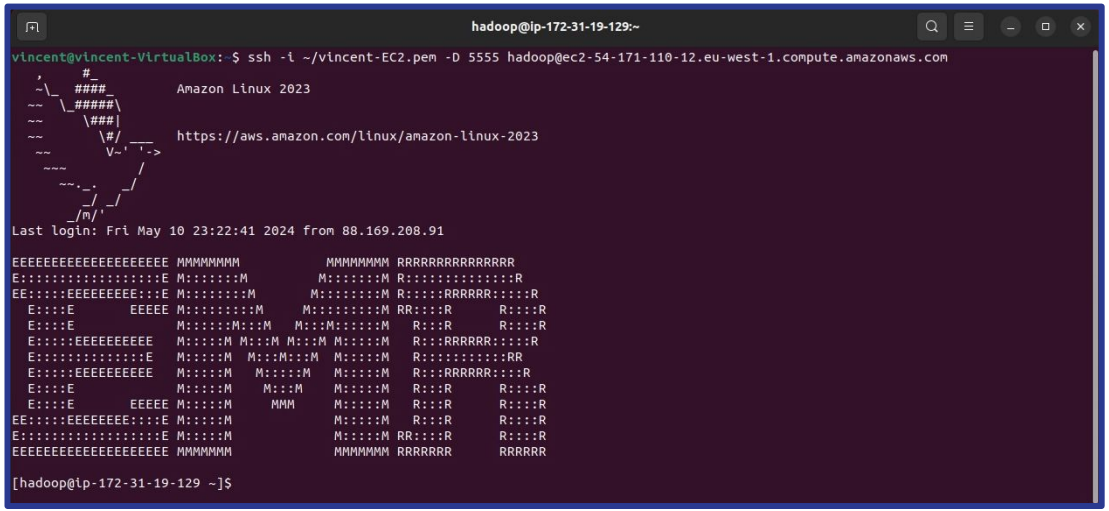
Partie 2: Réalisation de la chaîne de traitement des images

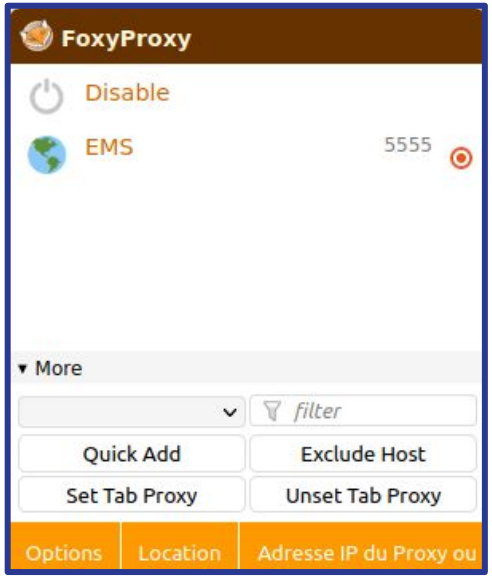
Étapes de la chaîne de traitement



Prérequis - JupyterHub

- Application préinstallée sur EMR
- Contrainte: exécution depuis le **réseau local du driver**
- Solution: **tunnel SSH** vers driver
- Mise en oeuvre: avec FoxyProxy





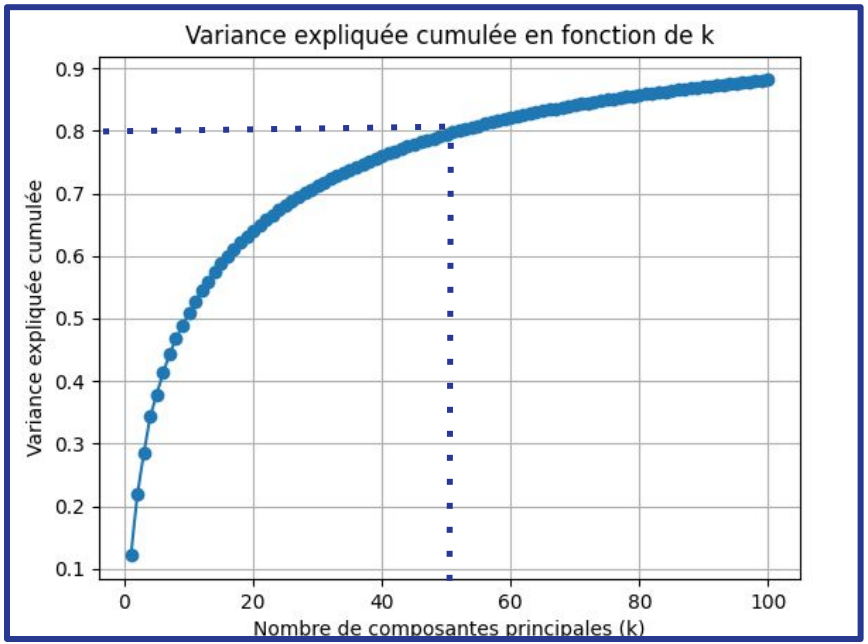
Extraction de features

- **Transfert learning**: modèle MobileNetV2 pré-entraîné (rapide, faible dimensionnalité sortie)
- **Dataframe Spark** (Pandas trop lent pour données massives)
- Chargement des poids sur le driver et **diffusion des poids sur les workers**

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	-	

ACP

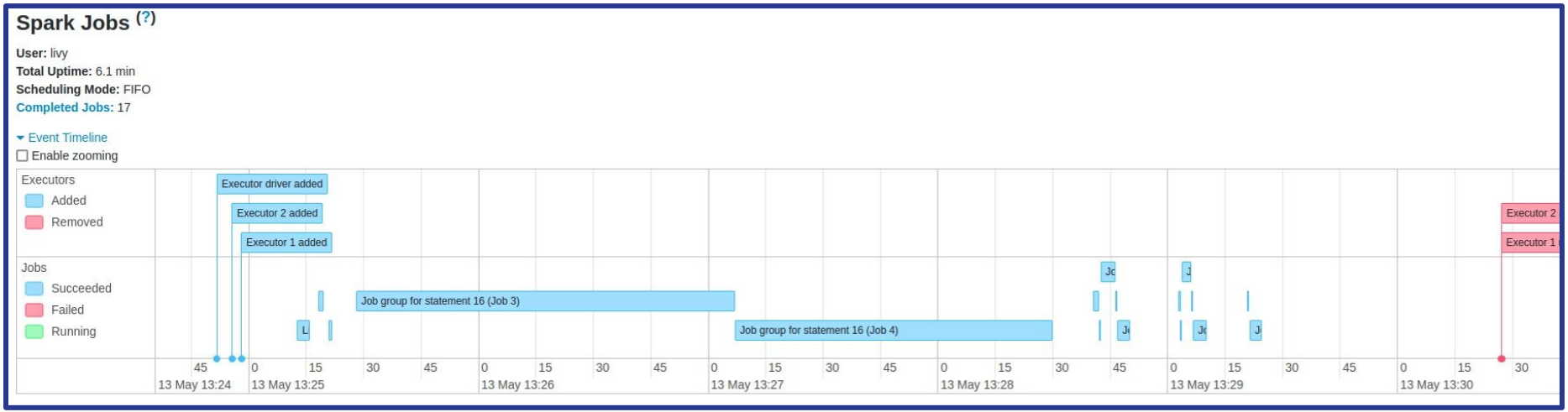
- Principe: réduire la taille des données de sortie en conservant le maximum d'information
- Déterminer le nombre de composantes pour conserver 80% de la variance expliquée



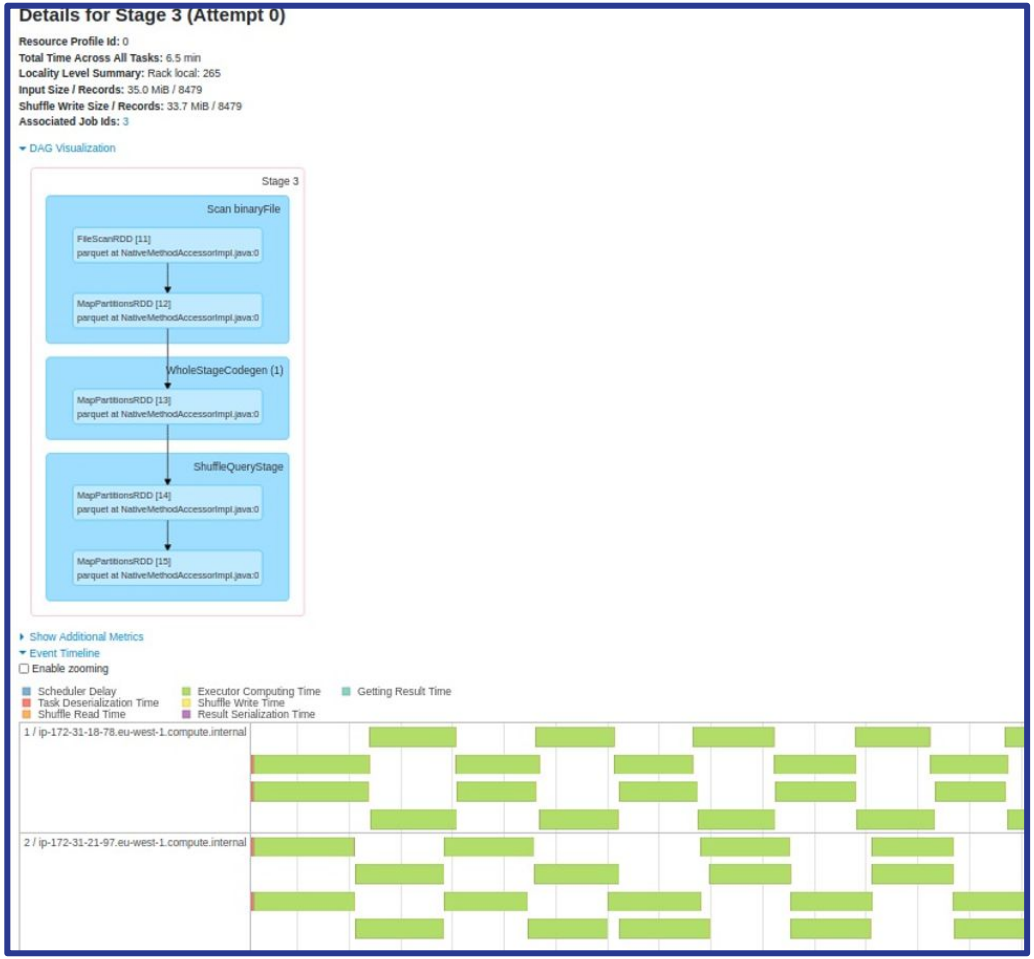

```
Variance cumulée de 0.8 atteinte à k = 52
```

Spark UI

- Permet de visualiser les Jobs/Stages/Tâches



Spark UI - ex: Stage3 - 265 tasks



Partie 3: Démo d’exécution du script pyspark sur le Cloud