



# FRUITS !

Nom du projet : **Réalisez un traitement dans un environnement Big Data sur le Cloud**

Présenté par : Nathan FARDIN

# Plan

I. Introduction

II. Environnement big  
data

III. Traitement des images

IV. Conclusion

# Le besoin

Proposer à l'entreprise une solution permettant de déployer un modèle d'extraction des features d'images de grande envergure capable de s'adapter à une forte augmentation du volume de données.



# Les données

Un dossier contenant 22 688 images de fruits au format 100x100 sur fond blanc uni.

# Conformité RGPD



**Choix de serveur Européens pour  
l'hébergement de notre EMR et de  
notre S3**

## Finalité

- Les informations sur des individus ne peuvent être enregistrées que dans un but spécifique, légal et légitime.

## Durée de conservation limitée

- une durée précise doit être fixée en fonction du type de données et de l'objectif du fichier.

## Droits des personnes

- Les individus ont des droits sur leurs données, incluant le droit d'accès, de rectification, et parfois le droit à l'effacement

## Proportionnalité et pertinence

- Les données enregistrées doivent être strictement nécessaires et pertinentes par rapport à l'objectif du fichier.

## Sécurité et confidentialité

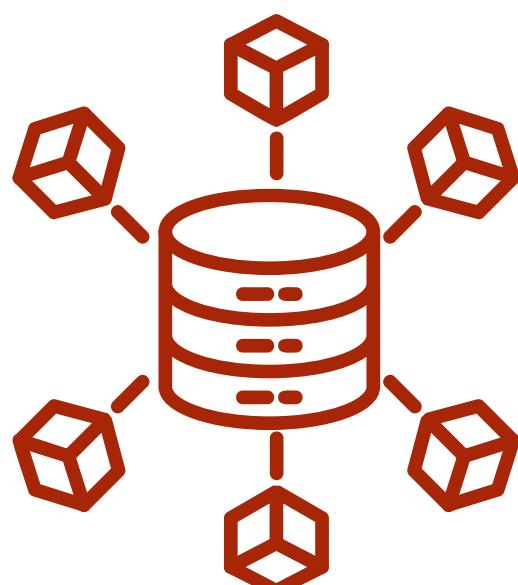
- Le responsable du fichier doit assurer la sécurité des informations détenues, en limitant l'accès .

# BIG DATA

**FRUITS! va devoir faire face à un afflux conséquent de données provenant de ces utilisateurs**

**Le traitement des données devra se faire rapidement pour garantir une bonne expérience pour les clients**

**La solution devra donc être en mesure de pouvoir s'adapter à un volume croissant de données reçu tout en fournissant des performances stables lors des extractions**



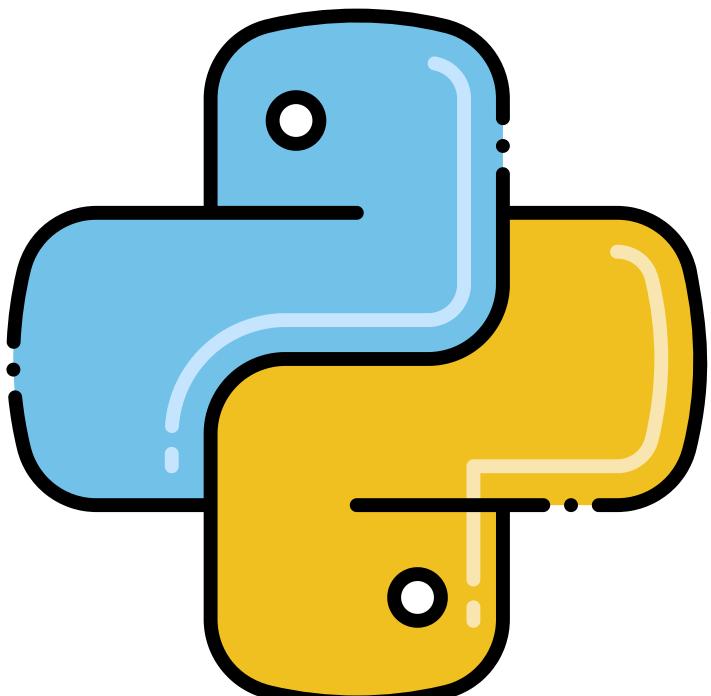
# Spark et PySpark

**Framework de calcul distribué qui permet d'optimiser le traitement des données via plusieurs workers**

**Prendre en charge la coordination de l'exécution des différentes tâches**

**Effectuer l'ensemble des actions en temps réel**

**PySpark est la librairie qui nous permet d'utiliser Spark sur Python**

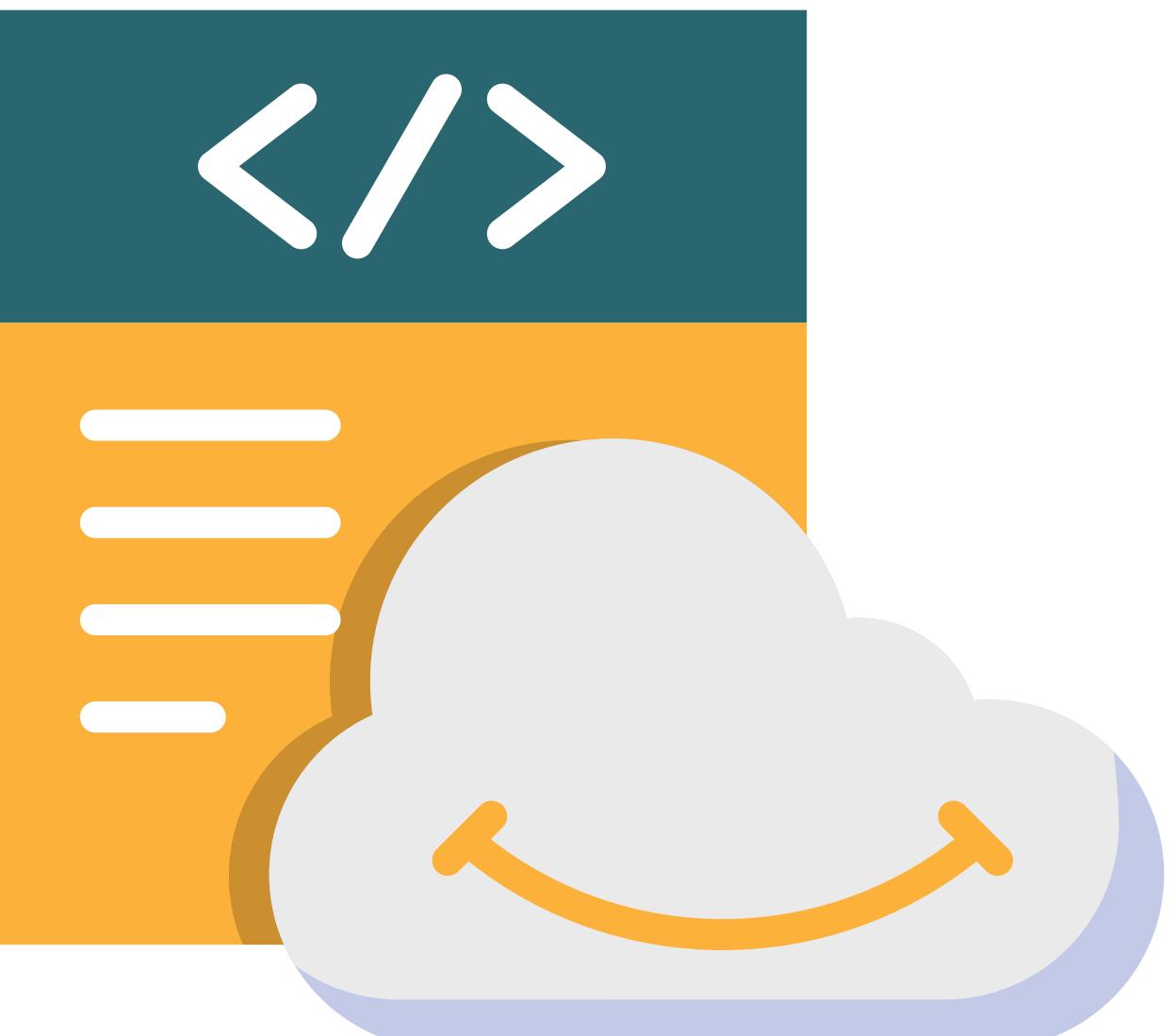


# AWS

**Plateforme cloud d'Amazon, offrant des services à la demande comme du stockage ou de la puissance de calcul.**

## Avantages clés :

- Service évolutif et flexible pour s'adapter aux besoins
- Paiement à l'usage
- Accès sécurisé aux différents services
- Divers emplacements dans le monde





# AWS : IAM

## Identity and Access Management

The screenshot shows the AWS IAM console interface. On the left, there's a navigation sidebar with links like 'Tableau de bord', 'Gestion des accès' (Groups, Persons, Roles, Policies), and 'Politiques des autorisations'. The main area displays two tables:

**Rôles (6) Infos**

<input type="checkbox"/> Nom du rôle	Entités de confiance
<a href="#">AmazonEMR-InstanceProfile-20250305T184000</a>	Service AWS: ec2
<a href="#">AmazonEMR-InstanceProfile-20250305T233414</a>	Service AWS: ec2
<a href="#">AmazonEMR-ServiceRole-20250305T184017</a>	Service AWS: elasticma
<a href="#">AWSServiceRoleForEMRCleanup</a>	Service AWS: elasticma

**Politiques des autorisations (3) Infos**

<input type="checkbox"/> Nom de la politique	Type	Entités attachées
<a href="#">AdministratorAccess</a>	Gérées par AWS – fonction...	2
<a href="#">AmazonEMR-Instance...</a>	Gérées par le client	1
<a href="#">s3fullacces</a>	Client en ligne	0

Permet de gérer divers éléments critique quant à la sécurité et à l'accès aux différents services d'AWS.

Par exemple en créant et en gérant les clés d'accès SSH.  
Ou en permettant de créer des rôles avec des droits d'accès définis à certains services.

# AWS : S3

## Simple Storage Service

p9-nf Info

Objets Propriétés Autorisations Métriques Gestion Points d'accès

Objets (4) Supprimer

Les objets sont les entités fondamentales stockées dans Amazon S3. Vous pouvez utiliser l'[inventaire Amazon S3](#) pour obtenir une liste personnes puissent accéder à vos objets, vous devez leur accorder explicitement des autorisations. [En savoir plus](#)

Rechercher des objets en fonction du préfixe

<input type="checkbox"/>	Nom	Type	Dernière modification	Taille	Classe de
<input type="checkbox"/>	<a href="#">bootstrap-emr.sh</a>	sh	09 Mar 2025 10:46:16 AM CET	241.0 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	-	-	-

Il servira à stocker des éléments clé :

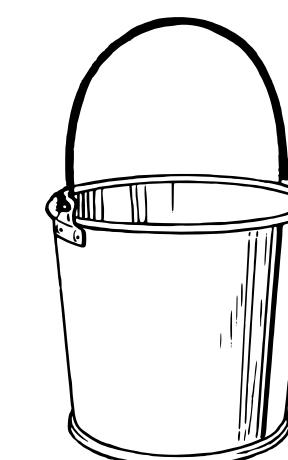
- Notebook
- Images
- Résultats
- Script de bootstrap

Solution de gestion du stockage

Espace de stockage évolutif et indépendant.

Accès au stockage sécurisé et nécessitant les bonnes autorisations (politiques AIM).

Liaison possible avec un EMR selon les autorisations accordé.



# AWS : EMR

## Elastic Map Reduce



Solution permettant d'exécuter des traitements distribués à grande échelle

Gestion d'instance EC2 (Elastic Compute Cloud) pour les calculs

**Groupes d'instances (3) Info**

Avec la configuration des groupes d'instances, chaque type de nœud est constitué du même type d'instance et de la même option d'achat d'instances : à la demande ou Spot.

Type et nom	ID	Statut	Instances	Option d'achat et prix	Taille EBS (GiB)
Primaire	ig-PPUVOQ7W4WMG	Résilié Job flow terminated.	0	À la demande	-
m5.xlarge	ci-03633731DXZXP7Y...	Résilié Instance was terminated.	-	À la demande (0.204 USD/h)	64 (2 volumes)
Principal (Unité principale)	ig-19SW8E91CTB7B	Résilié Job flow terminated.	0	À la demande	-
m5.xlarge	ci-03659928JOWGUW...	Résilié Instance was terminated.	-	À la demande (0.204 USD/h)	64 (2 volumes)
Tâche (Tâche - 1)	ig-3B5HENUBGTQHH	Résilié Job flow terminated.	0	À la demande	-
m5.xlarge	ci-03714353THJ8W41...	Résilié Instance was terminated.	-	À la demande (0.204 USD/h)	64 (2 volumes)
m5.xlarge	ci-04186883GQBC37I...	Résilié Instance was terminated.	-	À la demande (0.204 USD/h)	64 (2 volumes)

# AWS : EMR Configuration

## ▼ Nom et applications - requis Info

Donnez un nom à votre cluster et choisissez les applications que vous voulez y installer.

Nom

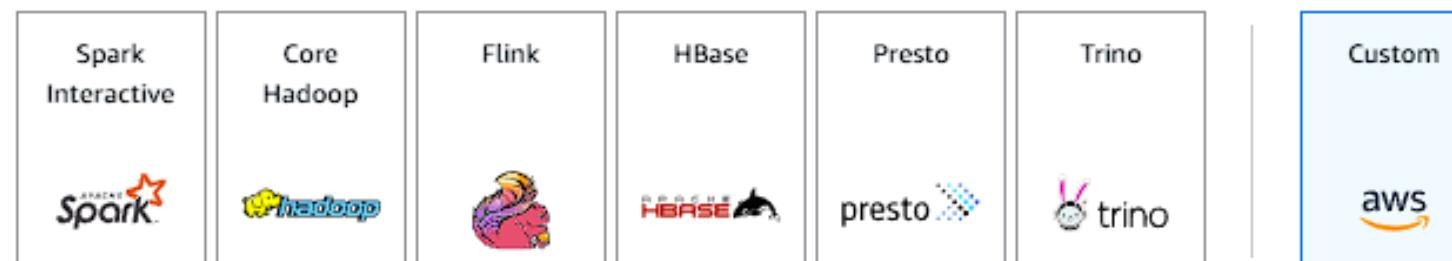
test

## Version Amazon EMR Info

Une version contient un ensemble d'applications susceptibles d'être installées sur votre cluster.

emr-7.8.0

## Offre d'applications



- AmazonCloudWatchAgent 1.300032.2
  - HCatalog 3.1.3
  - Hue 4.11.0
  - Livy 0.8.0
  - Pig 0.17.0
  - TensorFlow 2.16.1
  - Zeppelin 0.11.1
- Flink 1.20.0
  - Hadoop 3.4.1
  - JupyterEnterpriseGateway 2.6.0
  - Oozie 5.2.1
  - Presto 0.287
  - Tez 0.10.2
  - ZooKeeper 3.9.3
- HBase 2.6.1
  - Hive 3.1.3
  - JupyterHub 1.5.0
  - Phoenix 5.2.1
  - Spark 3.5.4
  - Trino 467

Choix des librairies indispensable pour l'exécution du code.

Un driver (instance principale)  
Deux workers (Tâches)

M5 xlarge pour allié efficacité et économie selon les besoins de Fruits!

## Groupes d'instances uniformes

### Primaire

#### Choisir un type d'instance EC2

m5.xlarge  
4 vCore 16 GiB mémoire  
EBS uniquement stockage  
Prix à la demande : 0.204 USD par inst...  
Prix Spot le plus bas : 0.054 USD (eu-nort...)

**Actions ▾**

#### Utiliser la haute disponibilité

Lancez des clusters hautement disponibles et plus résilients avec trois nœuds primaires sur des instances à la demande. Cette configuration s'applique pendant toute la durée de vie de votre cluster. [En savoir plus](#) ↗

#### ► Configuration de nœud - facultatif

## Unité principale

#### Choisir un type d'instance EC2

m5.xlarge  
4 vCore 16 GiB mémoire  
EBS uniquement stockage  
Prix à la demande : 0.204 USD par inst...  
Prix Spot le plus bas : 0.054 USD (eu-nort...)

**Actions ▾**

# AWS : EMR

## Bootstrap

▼ Actions d'amorçage (1) [Info](#) Suppr

Utilisez les actions d'amorçage pour installer des logiciels ou personnaliser la configuration de votre instance.

Nom	Emplacement Amazon S3
boot	<a href="s3://p9-nf/bootstrap-emr.sh">s3://p9-nf/bootstrap-emr.sh</a>

**Permet d'installer les librairies essentielles lors de l'exécution sur l'ensemble des machines du cluster.**

## Identification

Paire de clés Amazon EC2 pour SSH sur le cluster [Info](#)

 p9



**Choix d'une clé SSH permettant la connexion sans mot de passe/login.**

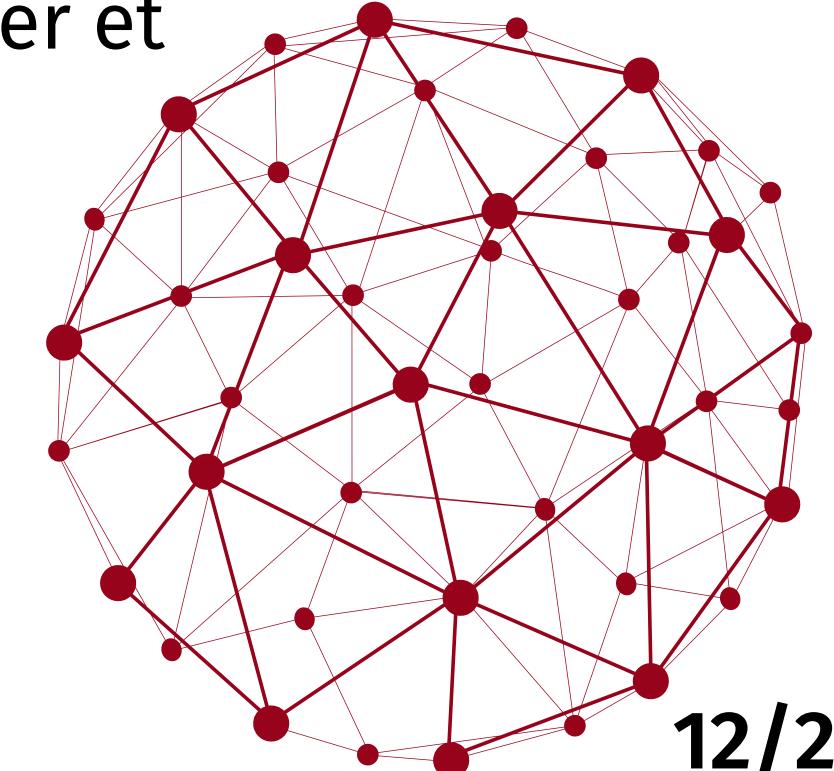
## Paramètres

```
{  
  "Classification": "jupyter-s3-conf",  
  "Properties": {  
    "s3.persistence.bucket": "p9-nf",  
    "s3.persistence.enabled": "true"  
  }  
}
```

**Permet d'accéder au bucket S3 pour interagir avec les données et écrire les résultats.**

# Connexion

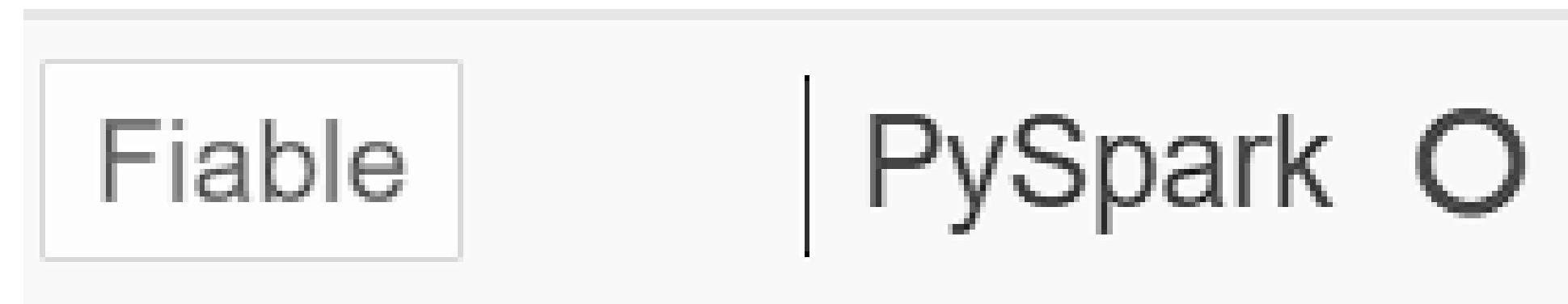
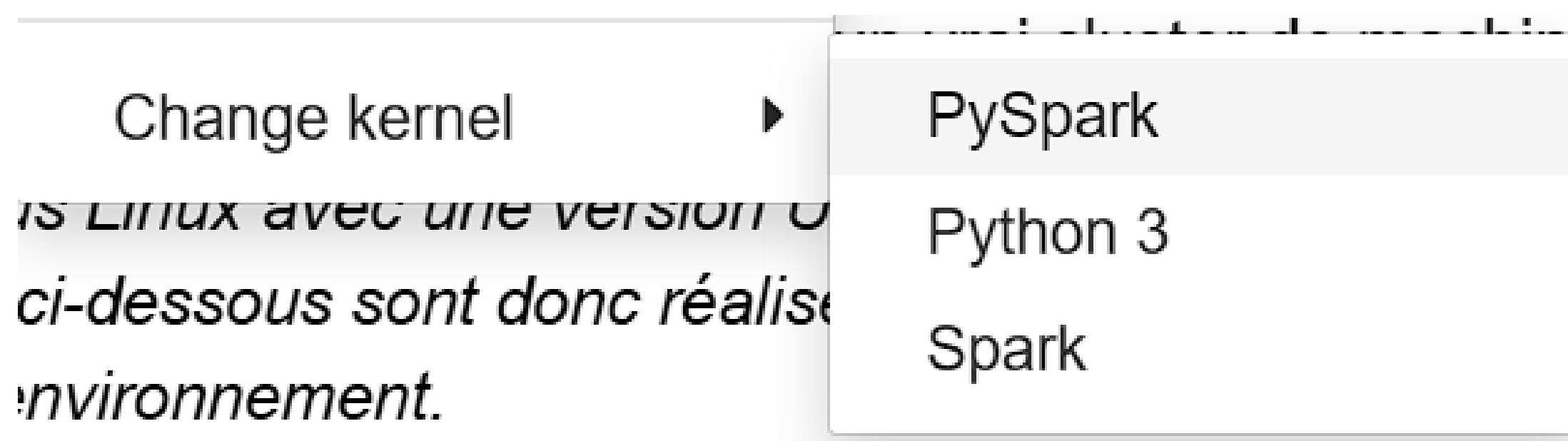
- Ouverture du port d'écoute 22 sur le groupe de sécurité de l'EC2 pour permettre la connexion SSH
- Configuration de Switch Oméga en intégrant une redirection vers le port 9443 correspondant à jupyter
- Utilisation de Putty pour la connexion : ajout de l'adresse du cluster, de la clé SSH et du port dynamique 9443 pour jupyter
- Connexion au notebook en utilisant l'adresse fournis par le cluster et utilisant des ID de base : jovyan/jupyter



# Traitement des images

## Introduction

**Utilisation de code Spark pour permettre la gestion du flux important de données : usage du kernel PySpark et initialisation de la session Spark lors de l'exécution de la première cellule.**

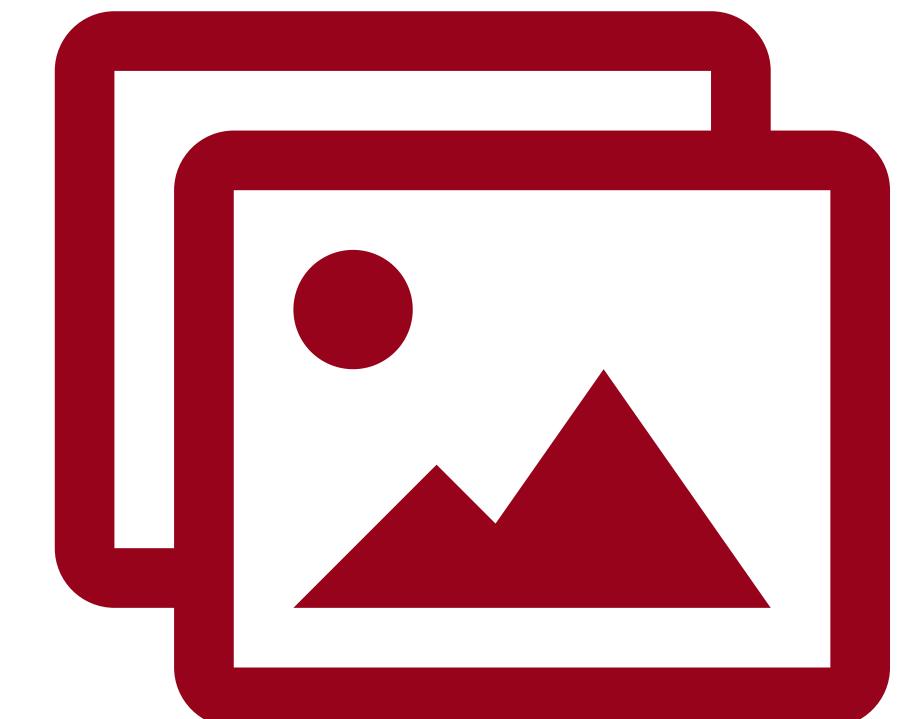


# Traitement des images

## Chargements des images

- Chargement de tous les fichiers dont le nom termine par .jpg dans le s3
- Chargement des informations des images dans un dataframe spark
- Utilisation du chemin d'accès au fichier pour catégoriser l'image

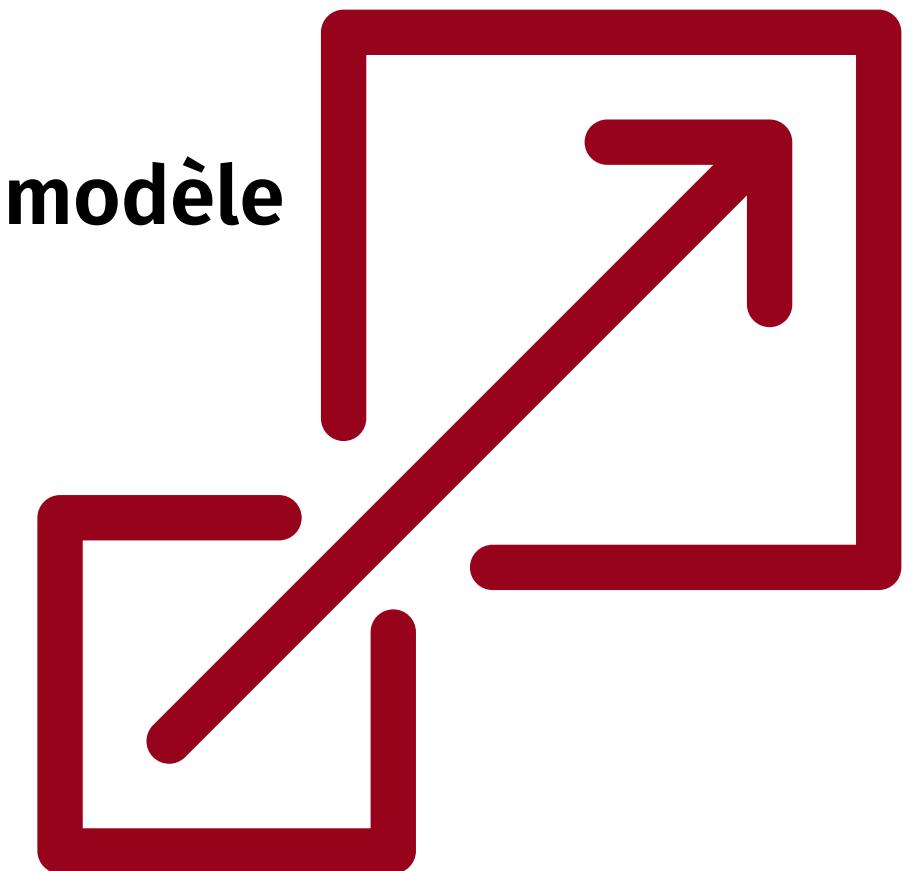
```
images = spark.read.format("binaryFile") \  
    .option("pathGlobFilter", "*.jpg") \  
    .option("recursiveFileLookup", "true") \  
    .load(PATH_Data)
```



# Traitement des images

## Preprocess

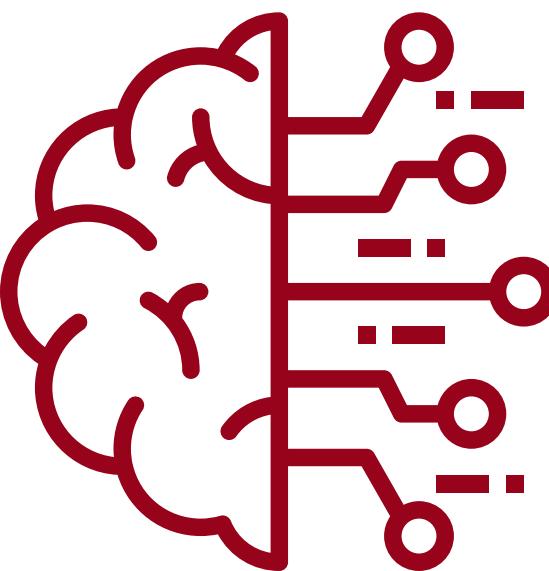
- D'origine, les images ont pour dimension  $100 \times 100 \times 3$ , or le modèle que nous utiliserons attend en entrée le format  $244 \times 244 \times 3$ .
- Ouverture des images sous forme de bytes
- Nous redimensionnerons les images à l'aide de PIL
- Conversion de l'image en tableau Numpy pour le traitement par le modèle



# Modélisation

## Choix du modèle

- Utilisation de mobile net v2
- Optimisé pour l'usage de données importantes en utilisant peu de ressource
- Réseaux de neurones convolutifs (CNN) pré entraîné sur la base ImageNet pour le traitement d'image (classification et extraction de la donnée)
- Transfert learning : utilisation du modèle pré entraîné. Création du modèle en retirant la dernière couche (classification) pour garder uniquement la partie extraction de features.



# Modélisation

## Distribution du poids

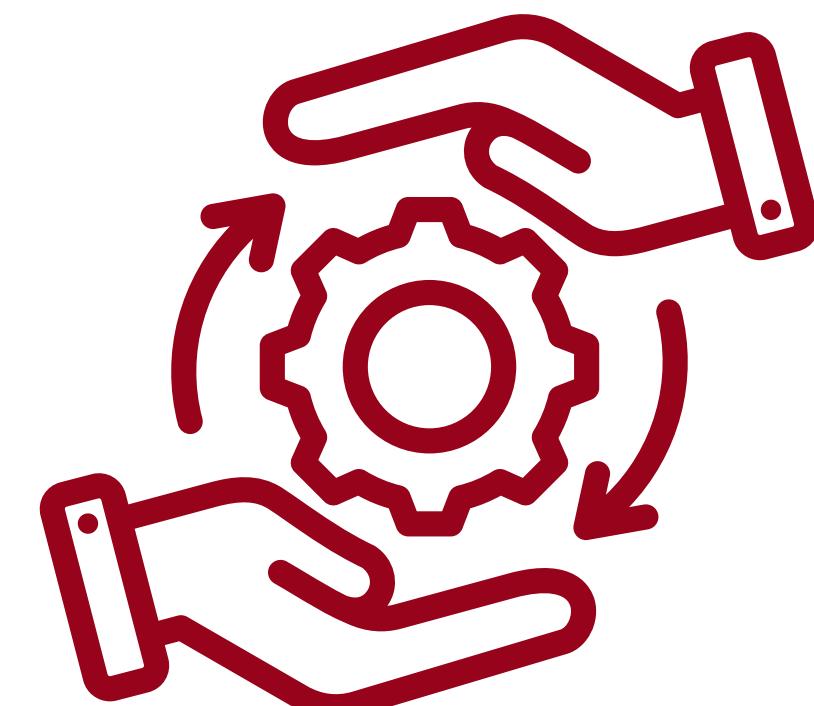
- Diffusion du poids aux différentes machines (chargement sur l'unité principal et diffusion aux workers) pour optimiser les ressources
- Gain de temps, car le modèle est déjà chargé, les images sont traitées en parallèles
- Adaptabilité en ayant la possibilité d'ajouter des workers avec le même code



# Modélisation

## Traitement des images

- Passage des images dans le modèle
- Utilisation de pandas UDF pour stocker les features extraites
- Gain de temps via la distribution des poids des modèles aux différents nœuds du cluster
- Vecteur de sortie sous forme (1,1,1280)



# PCA

## Principal component analysis

- Analyse en composante principale
- Permet de réduire le nombre de dimensions en conservant le maximum d'informations possible
- Cela permettra d'alléger la charge du modèle lors de l'étape de classification que Fruits ! souhaite implanter dans le futur.



# Résultats

**En attente d'une exploitation future, les documents sont stockés sous forme de fichier parquet dans le répertoire /results de notre s3 “p9-nf”**

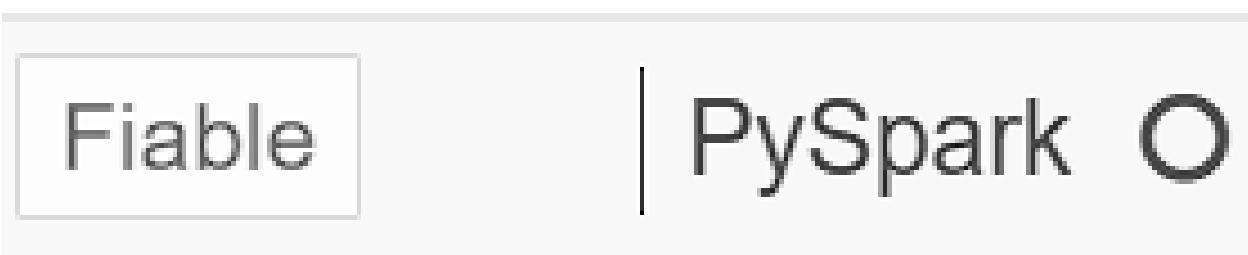
 <a href="#">_SUCCESS</a>	-	21 Mar 2025 08:55:03 AM CET	0 o	Standard
 <a href="#">part-00000-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:52:39 AM CET	3.5 Mo	Standard
 <a href="#">part-00001-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:52:39 AM CET	3.5 Mo	Standard
 <a href="#">part-00002-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:52:39 AM CET	3.5 Mo	Standard
 <a href="#">part-00003-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:52:39 AM CET	3.5 Mo	Standard
 <a href="#">part-00004-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:53:04 AM CET	3.5 Mo	Standard
 <a href="#">part-00005-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:53:04 AM CET	3.5 Mo	Standard
 <a href="#">part-00006-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:53:04 AM CET	3.4 Mo	Standard
 <a href="#">part-00007-8faf985e-cf47-4ac9-a078-7655c9d7c0f8-c000.snappy.parquet</a>	parquet	21 Mar 2025 08:53:04 AM CET	3.4 Mo	Standard



Dossier

# Execution du script

## Utilisation du kernel pyspark



## Création de la session Spark

```
Entrée [1]: # L'exécution de cette cellule démarre l'application Spark
Starting Spark application
ID          YARN Application ID   Kind  State  Spark UI  Driver log  User  Current session?
1  application_1742542387206_0002  pyspark  idle    Link      Link  None   ✓
SparkSession available as 'spark'.
```

Affichage des informations sur la session en cours et liens vers Spark UI :

```
Entrée [2]: %%info
Current session configs: {'driverMemory': '1000M', 'executorCores': 2, 'proxyUser': 'jovyan', 'kind': 'pyspark'}
ID          YARN Application ID   Kind  State  Spark UI  Driver log  User  Current session?
1  application_1742542387206_0002  pyspark  idle    Link      Link  None   ✓
```

# Execution du script

## Chargement des données d'images

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

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

path	modificationTime	length	content
s3://p9-nf/Test/W...	2025-03-16 14:26:55	7353	[FF D8 FF E0 00 1...]
s3://p9-nf/Test/W...	2025-03-16 14:26:56	7350	[FF D8 FF E0 00 1...]
s3://p9-nf/Test/W...	2025-03-16 14:26:55	7349	[FF D8 FF E0 00 1...]
s3://p9-nf/Test/W...	2025-03-16 14:26:55	7348	[FF D8 FF E0 00 1...]
s3://p9-nf/Test/W...	2025-03-16 14:24:17	7328	[FF D8 FF E0 00 1...]

labels

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

```
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://p9-nf/Test/Watermelon/r_106_100.jpg	Watermelon
s3://p9-nf/Test/Watermelon/r_109_100.jpg	Watermelon
s3://p9-nf/Test/Watermelon/r_108_100.jpg	Watermelon
s3://p9-nf/Test/Watermelon/r_107_100.jpg	Watermelon
s3://p9-nf/Test/Watermelon/r_95_100.jpg	Watermelon

# Execution du script

## Création du modèle

```
model = MobileNetV2(weights='imagenet',
                     include_top=True,
                     input_shape=(224, 224, 3))

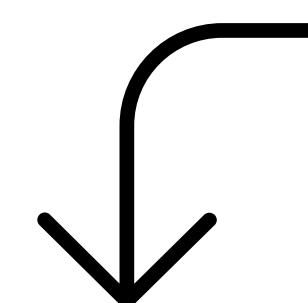
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224.h5
14536120/14536120 1s 0us/step

new_model = Model(inputs=model.input,
                  outputs=model.layers[-2].output)
```



## Determination des poids du modèle

```
broadcast_weights = sc.broadcast(new_model.get_weights())
```



## Recuperation des poids dans les workers

```
def model_fn():
    """
    Returns a MobileNetV2 model with top layer removed
    and broadcasted pretrained weights.
    """
    model = MobileNetV2(weights= None,
                         include_top=True,
                         input_shape=(224, 224, 3))

    new_model = Model(inputs=model.input,
                      outputs=model.layers[-2].output)
    new_model.set_weights(broadcast_weights.value)
    for layer in new_model.layers:
        layer.trainable = False
    return new_model
```

# Execution du script

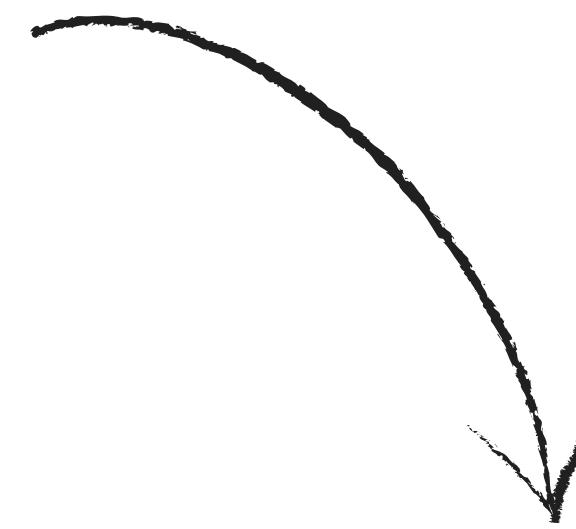
## Extraction des features

```
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)
```



## Réduction des dimension

```
dense_vector = udf(lambda a: Vectors.dense(a), VectorUDT())
dense_df = features_df.select("path", "label", dense_vector("features").alias("dense_features"))

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'), ...

pca = PCA(k=100, inputCol="dense_features", outputCol="pca_features")
model = pca.fit(dense_df)
result = model.transform(dense_df).select("path", "label", "pca_features")

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'), ...
```

# UI spark

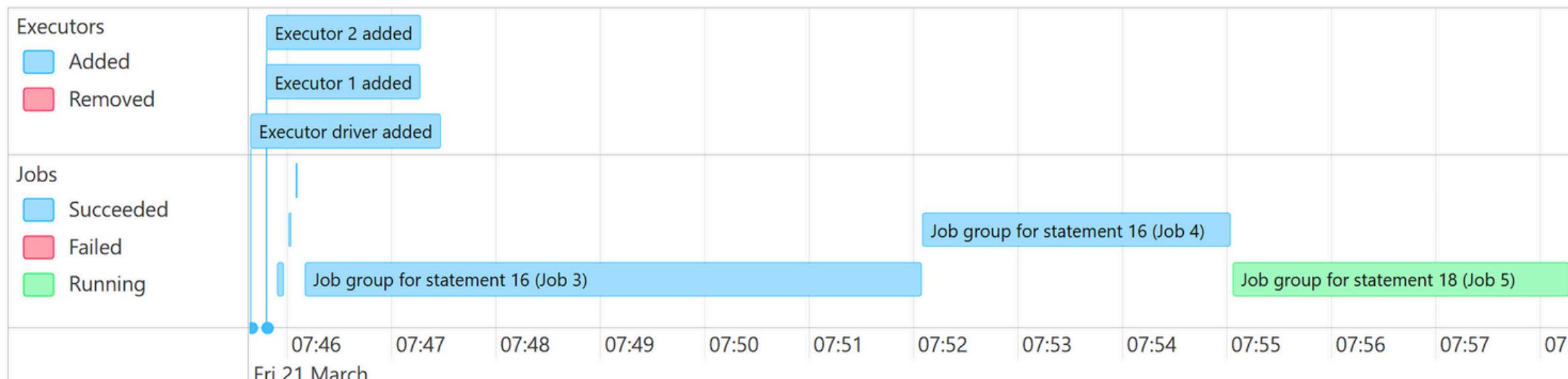
Interface permettant de visualiser l'usage des clusters au cours de l'exécution du script.

## Executors

[▶ Show Additional Metrics](#)

### Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
<b>Active(3)</b>	0	28.1 MiB / 9.9 GiB	17.2 MiB	4	<b>6</b>	0	1507	1513	1.1 h (3 s)	190 MiB	94.8 MiB	182 MiB	0
<b>Dead(0)</b>	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
<b>Total(3)</b>	0	28.1 MiB / 9.9 GiB	17.2 MiB	4	<b>6</b>	0	1507	1513	1.1 h (3 s)	190 MiB	94.8 MiB	182 MiB	0



# Conclusion

**En utilisant les services d'AWS, nous sommes en mesure de mettre en place un environnement big data pour Fruits!**

**Cette solution offre à l'entreprise de multiples avantages :**

- Economie en comparant au coup d'achat de matériel ou de location de serveurs dédiés**
- Adaptabilité et scalabilité facilitée pour s'adapter à l'afflux de données**
- Base utilisable pour ajouter une étape de classification**

