

« Réalisez un traitement dans un environnement Big Data sur le Cloud »

Projet n° 8 – OpenClassrooms parcours Data Scientist – Christophe ABALLEA

1. Contexte

La très jeune start-up de l'AgriTech, nommée "Fruits!", cherche à proposer des solutions innovantes pour la récolte des fruits.

La volonté de l'entreprise est de préserver la biodiversité des fruits en permettant des traitements spécifiques pour chaque espèce de fruits en développant des robots cueilleurs intelligents.

La start-up souhaite dans un premier temps se faire connaître en mettant à disposition du grand public une application mobile qui permettrait aux utilisateurs de prendre en photo un fruit et d'obtenir des informations sur ce fruit.

Pour la start-up, cette application permettrait de sensibiliser le grand public à la biodiversité des fruits et de mettre en place une première version du moteur de classification des images de fruits.

De plus, le développement de l'application mobile permettra de construire une première version de l'architecture Big Data nécessaire.

1.1. Objectif

L'objectif de ce projet est de développer une première chaîne de traitement des données qui comprendra le **preprocessing**, l'**extraction des caractéristiques**, et une étape de **réduction de dimension**. Ces caractéristiques permettront par la suite d'utiliser un modèle de classification.

Afin de tenir compte du fait que le volume des données évoluera très rapidement après la livraison du projet, il est demandé de :

- Déployer le traitement des données dans un **environnement Big Data**
- Développer les scripts en **pyspark** pour effectuer du **calcul distribué**

1.2. Déroulement

Ce projet se déroule en trois étapes :

- Établissement des choix techniques
- Validation des choix techniques en les testant en local, avec un nombre d'images limité à 494
- Déploiement de la solution dans un environnement Big Data en mode distribué

2. Choix techniques

2.1. Calcul distribué

L'énoncé du projet impose de développer des scripts en **PySpark** afin de prendre en compte l'augmentation très rapide du volume de données après la livraison du projet.

Pour comprendre rapidement et simplement ce qu'est PySpark et son principe de fonctionnement, il est conseillé de lire cet article : [PySpark : Tout savoir sur la librairie Python.](#)

Extrait du début de l'article :

*« Lorsque l'on parle de traitement de bases de données sur python, on pense immédiatement à la librairie pandas. Cependant, lorsqu'on a affaire à des bases de données trop massives, les calculs deviennent trop lents. Heureusement, il existe une autre librairie python, assez proche de pandas, qui permet de traiter des très grandes quantités de données : **PySpark**.*

Apache Spark est un framework open-source développé par l'AMPLab de UC Berkeley permettant de traiter des bases de données massives en utilisant le calcul distribué, technique qui consiste à exploiter plusieurs unités de calcul réparties en clusters au profit d'un seul projet afin de **diviser le temps d'exécution d'une requête**.

*Mais lorsque la collaboration entre Apache Spark et Python se créent, la librairie **PySpark** voit le jour. Elle propose alors à ses utilisateurs d'utiliser le langage Python, en gardant des performances similaires à des implémentations en Scala.*

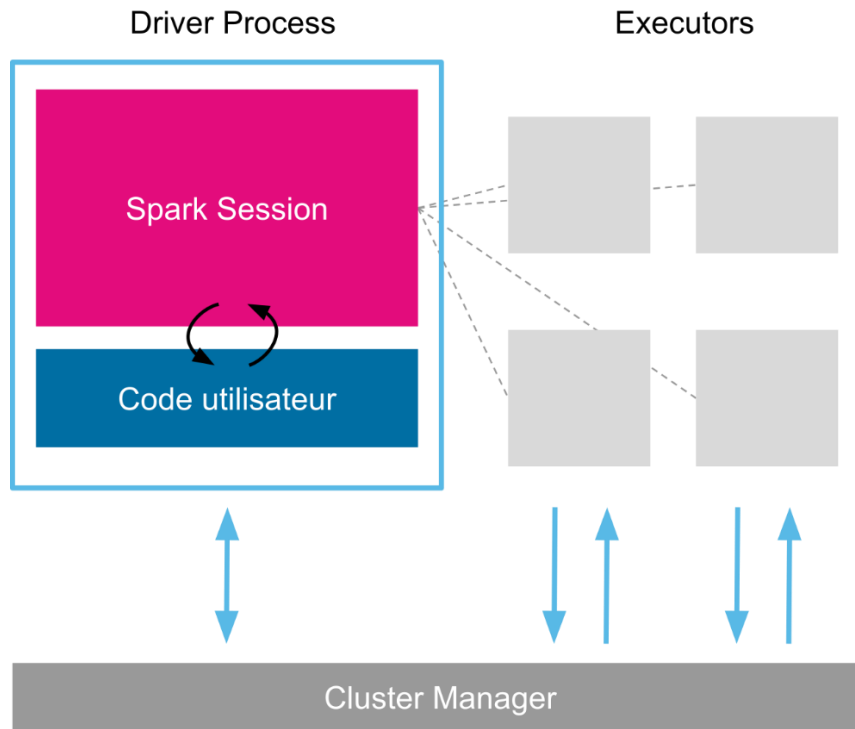
Pyspark est donc une bonne alternative à la librairie pandas lorsqu'on cherche à traiter des jeux de données trop volumineux qui entraînent des calculs trop chronophages. »

PySpark est un moyen de communiquer avec Spark via le langage Python.

Spark, quant à lui, est un outil qui permet de gérer et de coordonner l'exécution de tâches sur des données à travers un groupe d'ordinateurs. C'est un framework open source de calcul distribué in-memory pour le traitement et l'analyse de données massives.

Extrait de l'article plus complet [Apache Spark pour les nuls](#) qui permet de comprendre le fonctionnement de Spark, ainsi que le rôle d'une Spark Session qui sera utilisée dans ce projet :

« Les applications Spark se composent d'un pilote ('driver process') et de plusieurs exécuteurs ('executor processes'). Il peut être configuré pour être lui-même l'exécuteur (local mode) ou en utiliser autant que nécessaire pour traiter l'application, Spark prenant en charge la mise à l'échelle automatique par une configuration d'un nombre minimum et maximum d'exécuteurs.



Le driver (parfois appelé ‘Spark Session’) distribue et planifie les tâches entre les différents exécuteurs qui les exécutent et permettent un traitement réparti. Il est le responsable de l’exécution du code sur les différentes machines.

Chaque exécuteur est un processus Java Virtual Machine (JVM) distinct dont il est possible de configurer le nombre de CPU et la quantité de mémoire qui lui est alloué. Une seule tâche peut traiter un fractionnement de données à la fois. »

Dans les deux environnements (Local et Cloud), **Spark** sera exploité via des scripts python grâce à **PySpark** :

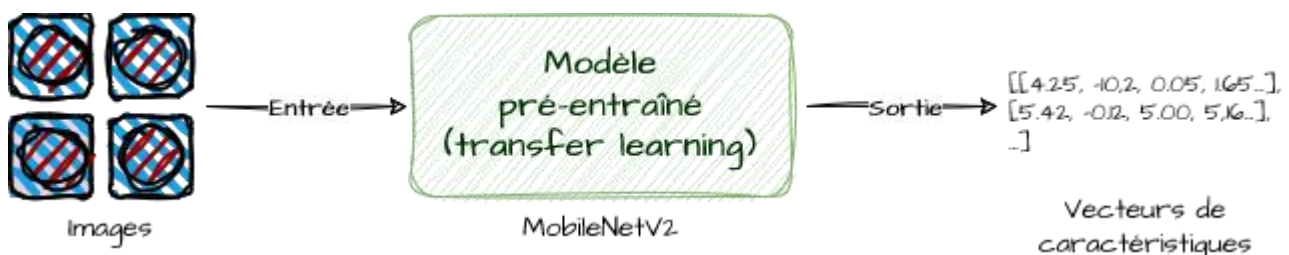
- Le calcul distribué sera simulé en version Local pour vérifier le bon fonctionnement de la solution
- Les opérations seront réalisées sur un cluster de machines pour la version Cloud

2.2. Transfer Learning

L'énoncé du projet impose également de réaliser une première chaîne de traitement des données qui comprendra le preprocessing et une étape de réduction de dimension, sans qu'il soit nécessaire d'entraîner un modèle pour le moment.

Le **transfert learning** consiste à utiliser un modèle entraîné sur une tâche spécifique comme point de départ pour une autre tâche. Cette technique est couramment employée dans le domaine de la reconnaissance visuelle.

Le modèle **MobileNetV2**, pré-entraîné sur une grande quantité d'images (**imagenet**) sera utilisé ici pour récupérer les vecteurs de caractéristiques des images.



La dernière couche du modèle est une couche softmax qui permet la classification des images parmi 1000 catégories, elle ne sera pas utile pour ce projet. L'avant dernière couche correspond à un vecteur réduit de dimension (1,1,1280), elle sera utilisée comme vecteur de caractéristiques des images.

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	-	-

MobileNetV2 a été retenu pour sa rapidité d'exécution, particulièrement adaptée pour le traitement d'un gros volume de données ainsi que la faible dimensionnalité du vecteur de caractéristique en sortie.

3. Déploiement de la solution en environnement local

3.1. Environnement de travail

La première étape consiste à installer Spark :

- Linux Ubuntu : [How to Install Spark on Ubuntu](#)
- Microsoft Windows : [Setup Apache Spark Environment on Windows 11? Step By Step Guide](#)
- Mac Os : [Install Apache Spark Latest Version on Mac OS](#)

Une autre solution consiste à utiliser l'environnement en ligne Google Colab qui présente un certain nombre d'avantages :

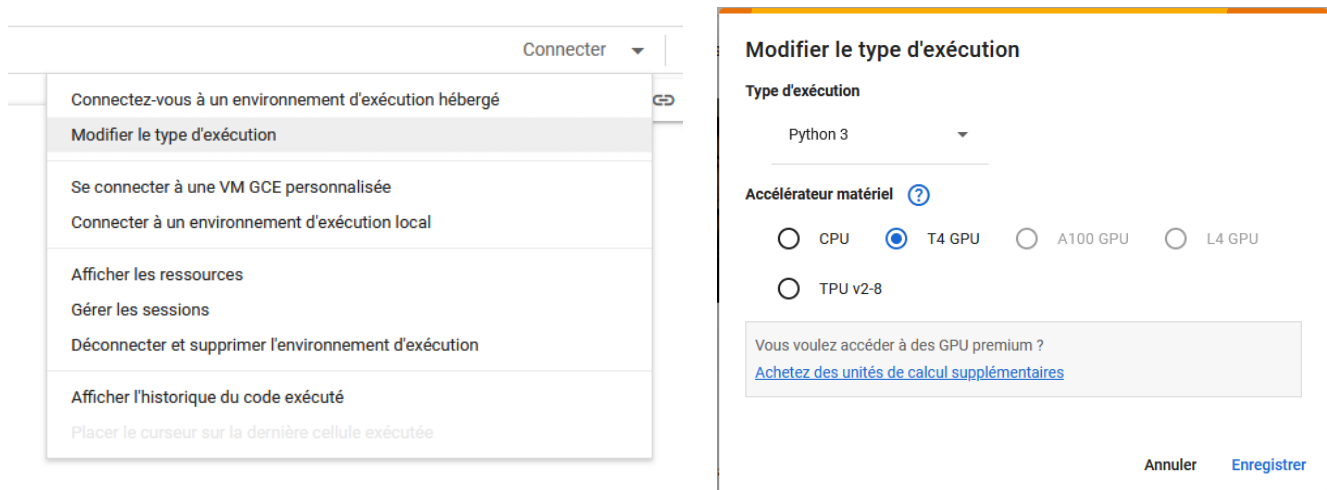
- Accès gratuit à du matériel GPU : puissance de calcul supérieure
- Configuration locale simplifiée : Jupyter Notebook préconfiguré, disponibilité de bibliothèques de machine learning
- Accès à Google Drive intégré
- Installation et configuration de PySpark simplifiée
- Gestion des dépendances simplifiée, pas d'environnement virtuel à installer
- ...

C'est cette solution qui a été retenue pour le déploiement en local.

3.2. Mise en œuvre de la solution

L'accès à Google Colab se fait via l'url <https://colab.research.google.com/>

Une fois le notebook créé, il est possible de spécifier un environnement basé sur du matériel GPU :



Le notebook **P9_Notebook_Local.ipynb** contient l'ensemble du code et déroule les opérations suivantes :

- Installation des packages et import des librairies
- Définition des chemins d'accès aux fichiers : les images étant stockées sur Google Drive, une demande d'autorisation de connexion est à valider
- Création de la session Spark
- Chargement des données
- Préparation du modèle MobileNetV2 et broadcast des poids pré-entraînés 'imagenet'
- Extraction des vecteurs de caractéristiques des images via Pandas UDF
- Réduction de dimension (PySpark StandardScaler + PCA)
- Validation des résultats après enregistrement des caractéristiques au format 'parquet'

L'affichage du SparkContext montre qu'il s'agit d'une session 'in-memory', toutes les opérations sont effectuées au sein d'une même machine virtuelle. Il s'agit donc bien d'un environnement local, sans calcul distribué sur plusieurs machines :

```
[7] spark
```





































































```
SparkSession - in-memory
SparkContext
Spark UI
Version
  v3.5.1
Master
  local
AppName
  p9
```

3.3. Validation

Le fichier **Results** a bien été généré sur Google Drive. Il se compose de 16 morceaux qui correspondent aux 16 partitions traitées en parallèles :

```
features_df = images.repartition(16).select(
    col("path"),
    col("label"),
    featurize_udf("content").alias("features")
)
```

Mon Drive > OC_P9 > Results ▾

Type ▾	Contacts ▾	Date de modification ▾		
Nom	↑	Propriétaire	Dernière modification ▾	Taille du fich
	._SUCCESS	 moi	19:49 moi	0 octet
	._SUCCESS.crc	 moi	19:49 moi	8 octets
	.part-00000-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:48 moi	2 Ko
	.part-00001-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:48 moi	2 Ko
	.part-00002-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:48 moi	2 Ko
	.part-00003-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:48 moi	2 Ko
	.part-00004-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:48 moi	2 Ko
	.part-00005-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:48 moi	2 Ko
	.part-00006-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00007-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00008-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00009-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00010-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00011-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00012-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00013-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00014-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	.part-00015-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet.crc	 moi	19:49 moi	2 Ko
	part-00000-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:48 moi	227 Ko
	part-00001-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:48 moi	228 Ko
	part-00002-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:48 moi	227 Ko
	part-00003-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:48 moi	230 Ko
	part-00004-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:48 moi	227 Ko
	part-00005-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:48 moi	228 Ko
	part-00006-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	229 Ko
	part-00007-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	231 Ko
	part-00008-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	221 Ko
	part-00009-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	223 Ko
	part-00010-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	228 Ko
	part-00011-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	230 Ko
	part-00012-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	226 Ko
	part-00013-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	231 Ko
	part-00014-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	229 Ko
	part-00015-614ff375-b7c6-4d48-8758-5da5d4f8be81-c000.snappy.parquet	 moi	19:49 moi	229 Ko

La lecture du fichier, qui contient les caractéristiques avant et après réduction de dimension, valide le processus :

```
Nombre d'images : 494
```

```
Taille du vecteur 'features' : 1280
```

```
Nombre de composantes expliquant 95 % de la variance : 60
```

```
Taille du vecteur 'pca_features' : 60
```

4. Déploiement de la solution sur le cloud

Une fois la solution fonctionnelle, l'objectif est de la déployer à plus grande échelle sur un vrai cluster de machines.

Plusieurs contraintes se posent :

- Quel prestataire de Cloud choisir ?
- Quelles solutions de ce prestataire adopter ?
- Où stocker un volume conséquent de données ?

4.1. Choix du prestataire Cloud : AWS (Amazon Web Services)

Le prestataire le plus connu et qui offre à ce jour l'offre la plus large dans le cloud computing est **Amazon Web Services** (AWS).

Certaines des offres AWS sont parfaitement adaptées à la problématique et c'est la raison pour laquelle leurs services seront utilisés ici.

L'objectif premier est de pouvoir, grâce à AWS, louer de la puissance de calcul à la demande. L'idée étant de pouvoir, quelle que soit la charge de travail, obtenir suffisamment de puissance de calcul pour traiter les images, même si le volume de données venait à fortement augmenter.

La capacité d'utiliser cette puissance de calcul à la demande permet de diminuer significativement les coûts en comparaison avec ceux d'une location de serveur complet sur une durée fixe (1 mois, 1 année par exemple).

4.2. Choix de la solution technique : EMR (Elastic Map Reduce)

Deux types de solutions sont disponibles : les solutions **IaaS (Infrastructure as a Service)** fournissent une infrastructure informatique de base (serveur, stockage, réseaux) sous forme de service à la demande ; alors que les solution **PaaS (Platform as a Service)** offrent un environnement complet (serveurs, bases de données, OS, environnement de développement...) pour le développement, le déploiement et la gestion d'application.

- Solution IaaS

Dans cette configuration **AWS** met à disposition des serveurs vierges en fournissant un accès administrateur : les instances **EC2** (Elastic Compute Cloud)

Amazon propose plus de 750 configurations différentes disponibles (combinaisons processeur / stockage / réseau / OS / modèle d'achat).

AVANTAGES	INCONVENIENTS
<ul style="list-style-type: none"> • Liberté totale de mise en œuvre de la solution • Possibilité de paramétrage à l'identique de la solution testée en local • Aucune modification de script à prévoir 	<ul style="list-style-type: none"> • Nécessité d'installer l'intégralité des outils (Spark, Java, Python, Jupyter Notebook...) • Nécessité d'implémenter les librairies sur toutes les machines (workers) du cluster • Nécessite de bonnes compétences en Administration système • Chronophage

- Solution PaaS

Parmi la très large offre AWS, le service **EMR (Elastic Map Reduce)** permet de louer des instances EC2 avec des applications et outils préinstallés et configurés. C'est une plateforme de Big Data dans le Cloud destinée à l'exécution des tâches de traitement de données distribuées à grande échelle, ce qui correspond parfaitement à ce projet.

Les principaux outils choisis pour ce projet sont disponibles dans l'environnement EMR : Spark, Tensorflow et Jupyter. EMR permet aussi l'installation personnalisée de packages complémentaires sur l'ensemble des machines du cluster, lors de l'instanciation.

AVANTAGES	INCONVENIENTS
<ul style="list-style-type: none"> • Solutions matérielles et logicielles optimisées par les ingénieurs AWS • Simplicité et rapidité de mise en œuvre • Assurance de disposer de versions matérielles et logicielles compatibles • Clonage possible des clusters • Évolutivité grâce à la possibilité d'une mise à l'échelle automatisée des ressources • Gestion de la sécurité optimisée 	<ul style="list-style-type: none"> • Contrôle limité sur l'infrastructure • Dépendance vis-à-vis des outils installés

La solution **PaaS** avec le service **EMR** d'Amazon Web Services sera retenue. Elle permet une mise en œuvre rapide et efficace, en s'affranchissant de toutes les contraintes matérielles et d'installation, tout en fournissant un environnement évolutif et optimisé pour le projet.

4.3. Choix de la solution de stockage : Amazon S3 (Simple Storage Service)

Les instances EC2 sont facturées à la minute. Il est donc recommandé de résilier un serveur lorsqu'il n'est plus utilisé, sachant qu'il est très rapide de recréer un cluster d'instances par clonage (moins de 10 minutes suffisent).

Stocker les données sur un instance EC2 impliquerait de déposer les images à chaque nouvelle instanciation et de devoir sauvegarder à un autre emplacement le fichier de caractéristiques généré.

Utiliser **Amazon S3** permet de s'affranchir de toutes ces problématiques. L'espace disque disponible est illimité, et il est indépendant de nos serveurs EC2. L'accès aux données sera d'autant plus rapide en restant dans l'environnement d'AWS et s'assurant de choisir la même région pour nos serveurs EC2 et S3.

L'accès aux données stockées dans un bucket S3 se fait très simplement en préfixant le path par **s3://**.

4.4. Configuration de l'environnement de travail

La première étape consiste à installer et de configurer AWS **CLI (Command Line Interface)**. Cette interface de ligne de commande permet d'interagir avec les différents services d'AWS, comme S3 par exemple.

Pour pouvoir utiliser AWS CLI, il faut le configurer en créant préalablement un utilisateur avec les autorisations adéquates. Dans ce projet il faut que l'utilisateur ait à minima un contrôle total sur le service S3.

La gestion des utilisateurs et de leurs droits s'effectue via le service IAM d'AWS. Une fois l'utilisateur créé et ses autorisations configurées, une paire de clés peut être créée pour une connexion sans saisie systématique des login/mot de passe.

L'accès aux serveurs EC2 se fait via SSH qu'il faut également configurer. Ici aussi, un système de clés permet d'éviter une identification manuelle à chaque connexion.

Toutes ses étapes de configuration sont parfaitement décrites ici : [Réalisez des calculs distribués sur des données massives / Découvrez Amazon Web Services](#).

4.5. Upload des images sur Amazon S3

Ici aussi les étapes sont décrites avec précision dans [Réalisez des calculs distribués sur des données massives / Stockez des données sur S3](#).

Seules les images contenues dans le dossier Test du jeu de données du projet seront recopiées sur S3, via quelques commandes fournies par AWS CLI :

- Création d'un bucket :

```
aws s3 mb s3://cap9-fruits
```

A noter qu'un bucket (conteneur de fichiers sur S3) doit être identifié par un nom unique au monde.

La vérification s'effectue par la commande **aws s3 ls**

- Transfert des images après s'être placé en local dans le répertoire les contenant :

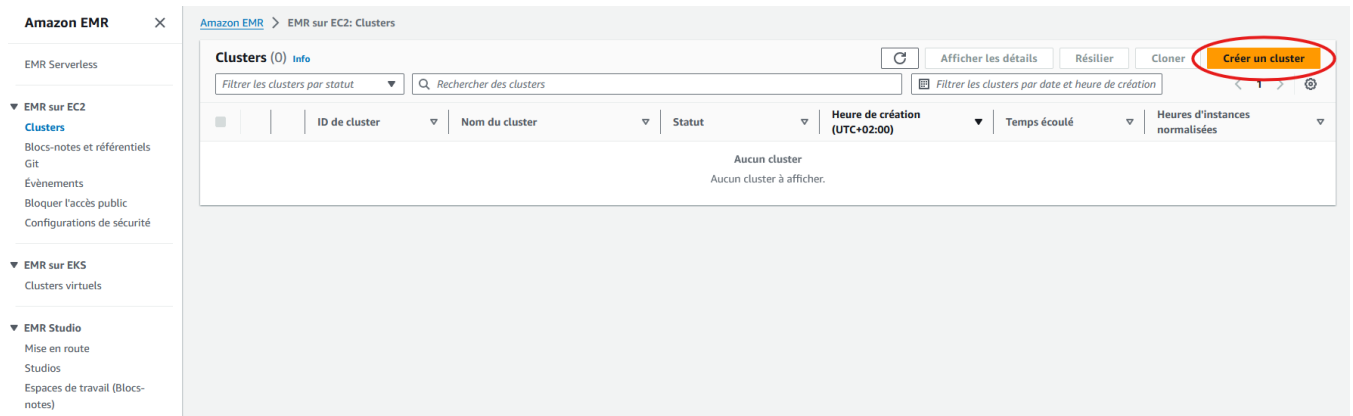
```
aws s3 sync . s3://cap9-fruits
```

Les données du projet sont maintenant disponibles sur Amazon S3.

4.6. Configuration du serveur EMR

Malgré quelques évolutions de l'interface Amazon EMR, l'essentiel des étapes est détaillé dans le tutoriel [Réalisez des calculs distribués sur des données massives / Déployez un cluster de calculs distribués](#).

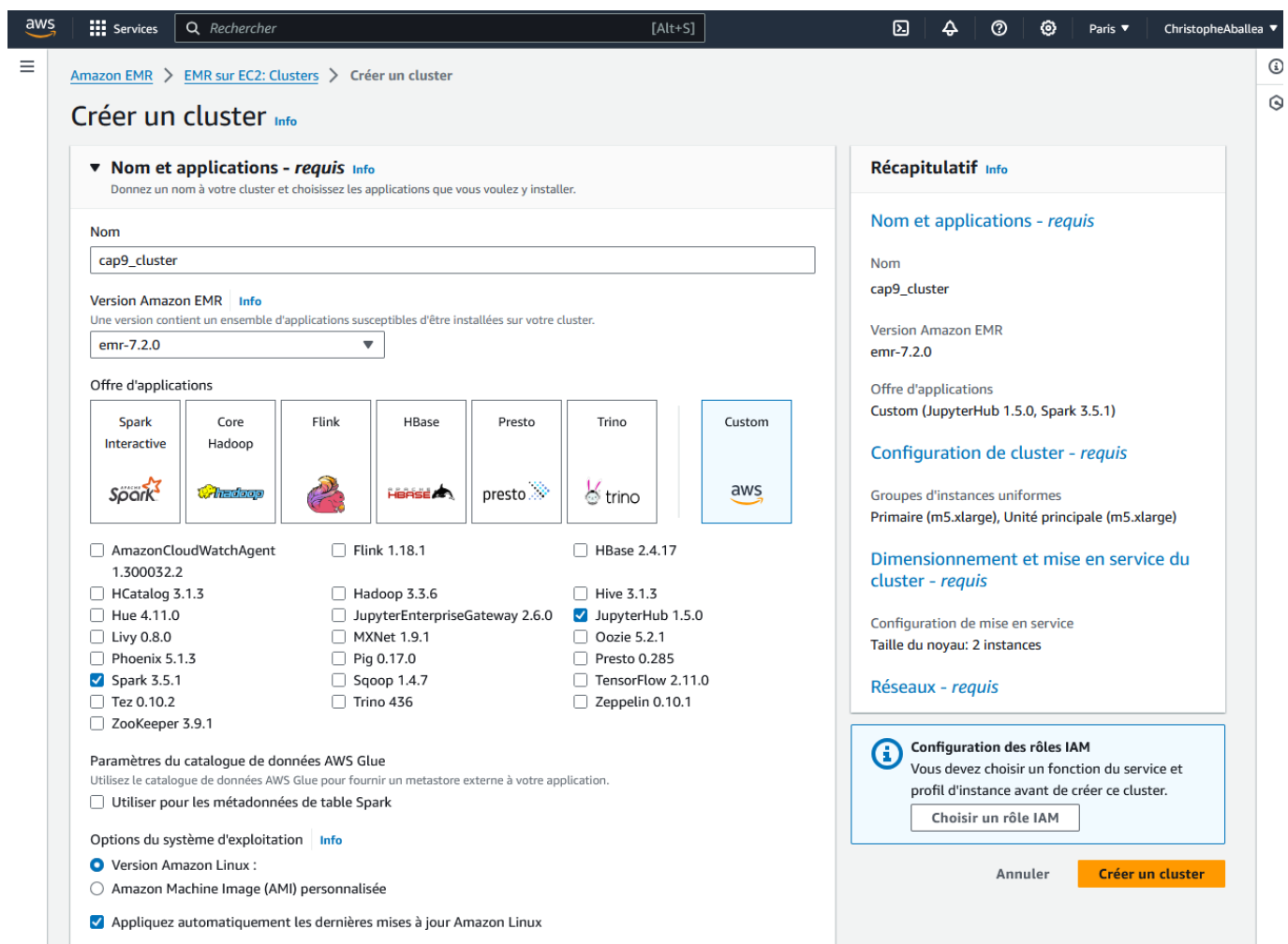
Étapes particulières permettant de configurer EMR selon les besoins du projet, à partir de l'interface Amazon EMR :



4.6.1. Étape 1 : Nom et applications

1. Nom du cluster : **cap9_cluster**
2. Version EMR : **emr-7.2.0**
3. Applications : **Spark 3.5.1** et **JupyterHub 1.5.0**

Afin de bénéficier d'une version Tensorflow plus récente (2.17.0), Tensorflow 2.11.0 n'est pas sélectionné ici.



4.6.2. Étape 2 : Configuration de cluster

Il s'agit ici de choisir les groupes d'instances EC2. Le groupe **Primaire** correspond au driver et le groupe **Unité principale** aux workers.

M5 est un type équilibré pouvant servir à tout type d'application, **xlarge** étant le moins onéreux.

▼

Configuration de cluster - *requis* [Info](#)

Choisissez une méthode de configuration pour les groupes de nœuds primaires, principaux et de tâches de votre cluster.

☒

Groupe d'instances uniformes

Choisissez le même type d'instance EC2 et la même option d'achat (à la demande ou Spot) pour tous les nœuds de votre groupe de nœuds. [En savoir plus](#)

☐

Flottes d'instances flexibles

Choisissez parmi la plus grande variété d'options de provisionnement pour les instances EC2 de votre cluster. Diversifiez les types d'instances et les options d'achat, et utilisez une stratégie d'allocation. [En savoir plus](#)

Groupe d'instances uniformes

Primaire

Choisir un type d'instance EC2

m5.xlarge

4 vCore 16 GiB mémoire

EBS uniquement stockage

Prix à la demande : 0.224 USD par instanc...

Prix Spot le plus bas : 0.073 USD (eu-west-3a)

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 : 0.224 USD par instanc...

Prix Spot le plus bas : 0.073 USD (eu-west-3a)

Actions ▼

Retirer le groupe d'instances

►

Configuration de nœud - *facultatif*

Ajouter un groupe d'instances de tâches

Vous pouvez ajouter jusqu'à 48 autres groupes d'instances de tâches.

Volume racine EBS

Le volume racine EBS s'applique aux systèmes d'exploitation et aux applications que vous installez sur le cluster. [Contraintes relatives au rapport de volume racine EBS](#)

Taille (Gio)

15

15 - 100 GiB par volume

Volume SSD polyvalent (gp3)

IOPS

3000

3000- 16000 IOPS par volume.

Choisissez un rapport maximum de 500:1 entre les IOPS et la taille du volume.

Débit (Mio/s)

125

125- 1000 MiB/s par volume.

Choisissez un rapport maximum de 0.25:1 entre le débit et les IOPS.

4.6.3. Étape 3 : Dimensionnement et mise en service du cluster

Pour disposer de 2 workers, il suffit de mettre la **taille de l'instance(s)** du groupe **Unité principale** à **2**.

▼ **Dimensionnement et mise en service du cluster - requis** Info

Choisissez la manière dont Amazon EMR doit dimensionner votre cluster.

Choisir une option

☒ **Définir manuellement la taille du cluster**
Utilisez cette option si vous connaissez vos modèles de charge de travail à l'avance.

☐ **Utiliser la mise à l'échelle gérée par EMR**
Surveillez les principales métriques de charges de travail afin qu'EMR puisse optimiser la taille du cluster et l'utilisation des ressources.

☐ **Utiliser un autoscaling personnalisée**
Pour dimensionner de manière programmatique les unités principales et les nœuds de tâches, créez des politiques d'autoscaling personnalisées.

Configuration de mise en service

Définissez la taille de votre unité principalegroupe d'instances. Amazon EMR tente de fournir cette capacité lorsque vous lancez votre cluster.

Nom	Type d'instance	Taille de l'instance(s)	Utiliser l'option d'achat Spot
Unité principale	m5.xlarge	<input type="text" value="2"/>	<input type="checkbox"/>

4.6.4. Étape 4 : Actions d'amorçage

Cette étape est très importante et particulièrement intéressante puisqu'elle permet d'installer des logiciels complémentaires sur tous les nœuds du cluster, avant le traitement des données par Amazon EMR. Ces actions sont aussi exécutées dans le cas d'ajout d'un nœud à un cluster en cours d'exécution.

Dans le cadre de ce projet, les actions d'amorçage sont utiles pour installer les versions de librairies adaptées. Explication de la procédure : [Réalisez des calculs distribués sur des données massives / Bootstrapping](#)

▼ **Actions d'amorçage (1)** Info

Supprimer

Modifier

Ajouter

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

	Nom	Emplacement Amazon S3	Arguments
<input type="radio"/>	Installation librairies	s3://cap9-fruits/bootstrap-emr.sh	-

Fichier **bootstrap-emr.sh** :

```
#!/bin/bash
sudo apt-get update
sudo apt-get install -y build-essential libjpeg-dev zlib1g-dev libssl-dev libffi-dev
sudo python3 setup.py build_ext
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 numpy==1.17.3
sudo python3 -m pip install pandas==1.3.5
sudo python3 -m pip install tensorflow
sudo python3 -m pip install pyarrow
sudo python3 -m pip install boto3
sudo python3 -m pip install s3fs
sudo python3 -m pip install fsspec
```

4.6.5. Étape 5 : Paramètres du logiciel

L'exécution du notebook Jupyter va modifier les sorties de cellules. Par défaut JupyterHub ouvre et enregistre les notebooks sur l'instance Primaire EC2. Pour conserver ces sorties après la résiliation du cluster, ce notebook a été enregistré sur le bucket S3. En paramétrant la persistance des notebooks sur S3, toutes les modifications apportées y seront bien sauvegardées.

▼ Paramètres du logiciel [Info](#)
Remplacez les configurations par défaut pour des applications spécifiques de votre cluster.

☒ Entrer la configuration ☐ Charger JSON à partir d'Amazon S3

```
1 [
2   {
3     "Classification": "jupyter-s3-conf",
4     "Properties": {
5       "s3.persistance.bucket": "cap9-fruits",
6       "s3.persistance.enabled": "true"
7     }
8   }
9 ]
```

JSON Ln 1, Col 1 ✖ : 0 ⚠ : 0

4.6.6. Étape 6 : Configuration de sécurité et paire de clé EC2

L'accès aux instances EC2 s'effectue via le protocole SSH. Pour une connexion sans saisie des login/mot de passe, il faut rattacher la paire de clés préalablement créée pour EC2.

▼ Configuration de sécurité et paire de clés EC2 [Info](#)
Choisissez une configuration de sécurité ou créez-en une nouvelle que vous pourrez réutiliser avec d'autres clusters.

Configuration de sécurité
Sélectionnez les paramètres de chiffrement, d'authentification, d'autorisation et de service de métadonnées d'instance de votre cluster.

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

4.6.7. Étape 7 : Profil d'instance EC2 pour Amazon EMR

Le stockage des différentes données (fichier 'bootstrap-emr.sh', images, notebook Jupyter...) sur Amazon S3 ne peut être fonctionnel qu'à la condition que le cluster EMR ait un accès en lecture/écriture au bucket (compartiment) S3.

Profil d'instance EC2 pour Amazon EMR

Le profil d'instance attribue un rôle à chaque instance EC2 d'un cluster. Le profil d'instance doit spécifier un rôle qui peut accéder aux ressources pour vos étapes et actions d'amorçage.

☐ Choisir un profil d'instance existant

Sélectionnez un rôle par défaut ou un profil d'instance personnalisé avec des stratégies IAM attachées afin que votre cluster puisse interagir avec vos ressources dans Amazon S3.

☒ Choisir un profil d'instance

Laissez Amazon EMR créer un profil d'instance afin de pouvoir spécifier un ensemble personnalisé de ressources auquel il peut accéder dans Amazon S3.

Accès au compartiment S3 [Info](#)

☒ Compartiments S3 ou préfixes spécifiques de votre compte [Info](#)

Choisissez les compartiments ou préfixes auxquels vous voulez que ce profil d'instance accède.

☐ Tous les compartiments S3 de ce compte avec accès en lecture et en écriture

Accordez au profil d'instance l'accès à tous les compartiments pour lesquels l'accès en lecture et en écriture est activé dans votre compte.

Compartiments S3

Nous avons déjà ajouté les ressources que vous avez configurées dans la section [Journaux de cluster](#). Choisissez les compartiments S3 et les préfixes de compartiment dans lesquels vous stockez les journaux et les données de votre cluster, les actions d'amorçage et les étapes.

S3 URI

[Afficher](#)[Parcourir S3](#)[Ajouter](#)

Compartiment S3

Préfixe

Autorisation

aws-logs-82576541...
Hérité des journaux de cluster

elasticmapreduce

Lecture et écriture

[Modifier](#)

cap9-fruits

/

Lecture et écriture

[Supprimer](#)

4.7. Instanciation du cluster EMR

Une fois les étapes de configuration terminées le bouton [Créer un cluster](#) lance la création du cluster.

Votre cluster « cap9_cluster_simplifie » a été créé.

[Amazon EMR](#) > [EMR sur EC2: Clusters](#) > cap9_cluster_simplifie

cap9_cluster_simplifie

Mise à jour il y a moins d'une minute

[Résilier](#)

[Cloner dans AWS CLI](#)

[Cloner](#)

Récapitulatif

Informations sur le cluster

ID de cluster
j-16IRVXFT8EYJ0

Configuration de cluster

Groupes d'instances

Capacité

1 primaire(s) | 1 unité(s) principale(s) | 2 tâche(s)

Applications

Version d'Amazon EMR
emr-7.2.0

Applications installées

JupyterHub 1.5.0, Spark 3.5.1

Gestion des clusters

Destination des journaux dans Amazon S3
[aws-logs-825765415545-eu-west-1/elasticmapreduce](#)

DNS public du nœud primaire

-

Statut et heure

Statut

🔄 Démarrage en cours

Heure de création

23 septembre 2024 17:48 (UTC+02:00)

Temps écoulé

0 secondes

Propriétés

Actions d'amorçage

Instances (Matériel)

Étapes

Applications

Configurations

Surveillance

Évènements

identifications (1)

Journaux de cluster [Info](#)

Archiver les fichiers journaux dans Amazon S3

Activé

Emplacement Amazon S3

[s3://aws-logs-825765415545-eu-west-1/elasticmapreduce/](#)

Chiffrement pour les journaux

Désactivé

Résiliation du cluster et remplacement des nœuds [Info](#)

[Modifier](#)

Option de résiliation

Résilier manuellement le cluster

Protection contre la résiliation

Désactivé

Temps d'inactivité

-

Remplacement des nœuds défectueux

Activé

Réseau et sécurité [Info](#)

Réseau

Cloud privé virtuel (VPC)

[vpc-041556e22a3ac84d](#)

Sous-réseau(x) et zone(s) de disponibilité

[subnet-0522a511399972204](#) | eu-west-1a

► Groupes de sécurité EC2 (pare-feu)

Configuration de sécurité

Configuration de sécurité

Aucun

Paire de clés EC2

christophe-ras-ec2-2

Autorisations

Fonction du service pour Amazon EMR

[AmazonEMR-ServiceRole-20240831T002026](#)

Profil d'instance EC2

[AmazonEMR-InstanceProfile-20240831T002010](#)

Rôle d'autoscaling personnalisé

Non configuré

Le cluster est créé immédiatement et son statut évolue en quelques minutes jusqu'à passer **En attente** :

Statut et heure

Statut

Démarrage en cours

Statut et heure

Statut

Action d'amorçage

Statut et heure

Statut

En attente

Il est alors prêt à être utilisé.

4.8. Création du tunnel SSH à l'instance EC2 primaire

La connexion au nœud primaire se fait via le protocole SSH. Si les éléments de connexion sont fournis à l'instanciation du cluster, il reste quelques étapes pour rendre cette connexion fonctionnelle :

- Ouverture du port 22 sur le serveur pour accepter les connexions entrantes
- Création du tunnel SSH depuis la machine cliente (le poste de travail utilisé en local)
- Configuration d'un logiciel Proxy pour indiquer au navigateur en local d'utiliser le tunnel SSH

4.8.1. Ouverture du port pour les connexions entrantes

La procédure est détaillée ici : [Réalisez des calculs distribués sur des données massives / Lancement d'une application à partir du driver.](#)

A partir de la console Amazon EC2, le paramétrage des **Règles entrantes** s'effectue via la modification du groupe de sécurité **ElasticMapReduce-Master** :

Les règles de groupe de sécurité entrantes ont été modifiées avec succès sur le groupe de sécurité. (sg-09fb8512919c647a0 | ElasticMapReduce-master)

EC2 > Groupes de sécurité > sg-09fb8512919c647a0 - ElasticMapReduce-master

sg-09fb8512919c647a0 - ElasticMapReduce-master

Actions

Détails

Nom du groupe de sécurité

ElasticMapReduce-master

ID du groupe de sécurité

sg-09fb8512919c647a0

Description

Master group for Elastic MapReduce created on 2024-08-24T18:49:17.069Z

ID de VPC

vpc-041556e22a3acf84d

Propriétaire

825765415545

Nombre de règles entrantes

9 Entrées d'autorisation

Nombre de règles sortantes

1 Entrée d'autorisation

Règles entrantes

Règles sortantes

Balises

Règles entrantes (9)

Recherche

Gérer les balises

Modifier les règles entrantes

	Name	ID de règle de grou...	Version IP	Type	Protocole	Plage de ports	Source	Description
<input type="checkbox"/>	-	sg-0220a329bb220bbaf	-	TCP personnalisé	TCP	8443	pl-a5a742cc	-
<input type="checkbox"/>	-	sg-067bfb5edd2b45ce2	-	Tous les TCP	TCP	0 - 65535	sg-09fb8512919c647a...	-
<input type="checkbox"/>	-	sg-076e8d963d4179...	-	Tous les ICMP - IPv4	ICMP	Tous	sg-09fb8512919c647a...	-
<input type="checkbox"/>	-	sg-0c981925bd64116fc	-	Tous les UDP	UDP	0 - 65535	sg-0d25715af52eea6d...	-
<input type="checkbox"/>	-	sg-07b0392f784863d...	-	Tous les TCP	TCP	0 - 65535	sg-0d25715af52eea6d...	-
<input type="checkbox"/>	-	sg-08f9f6103b3485ae2	IPv6	SSH	TCP	22	::/0	-
<input type="checkbox"/>	-	sg-006fb0f02ad63c0c1	IPv4	SSH	TCP	22	0.0.0.0/0	-
<input type="checkbox"/>	-	sg-050652cda08fe6c7f	-	Tous les ICMP - IPv4	ICMP	Tous	sg-0d25715af52eea6d...	-
<input type="checkbox"/>	-	sg-0b9dc5bf4fc7ea24	-	Tous les UDP	UDP	0 - 65535	sg-09fb8512919c647a...	-

4.8.2. Création du tunnel SSH

Un lien fournissant la procédure d'ouverture du tunnel pour Windows, Mac et Linux, est disponible sur la console du cluster :

Gestion des clusters

Destination des journaux dans Amazon S3

aws-logs-825765415545-eu-west-1/elasticmapreduce

DNS public du nœud primaire

ec2-18-201-79-18.eu-west-1.compute.amazonaws.com

Connexion au nœud primaire à l'aide de SSH




Connexion au nœud primaire à l'aide de SSH

Vous pouvez vous connecter au nœud primaire d'Amazon EMR à l'aide de SSH pour effectuer des actions telles que l'exécution de requêtes interactives, l'examen de fichiers journaux, l'envoi de commandes Linux et l'affichage d'interfaces web hébergées sur des clusters Amazon EMR. [En savoir plus](#)

Windows

Mac/Linux

1. Téléchargez PuTTY.exe sur votre ordinateur à partir de : <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> 
2. Démarrez PuTTY.
3. Dans la liste Category (Catégorie), sélectionnez Session.
4. Dans le champ Host Name (Nom d'hôte), entrez `hadoop@ec2-18-201-79-18.eu-west-1.compute.amazonaws.com`
5. Dans la liste Category, développez Connection (Connexion) > SSH, puis sélectionnez Auth.
6. Pour Private key file for authentication (Fichier de clé privée pour l'authentification), sélectionnez Browse (Parcourir) et le fichier de clé privée (`christophe-ras-ec2-2.ppk`) que vous avez utilisé pour lancer le cluster.
7. Cliquez sur Open (Ouvrir).
8. Sélectionnez Yes (Oui) pour ignorer l'alerte de sécurité.

[Afficher les interfaces web hébergées sur des clusters Amazon EMR](#) 

Fermer

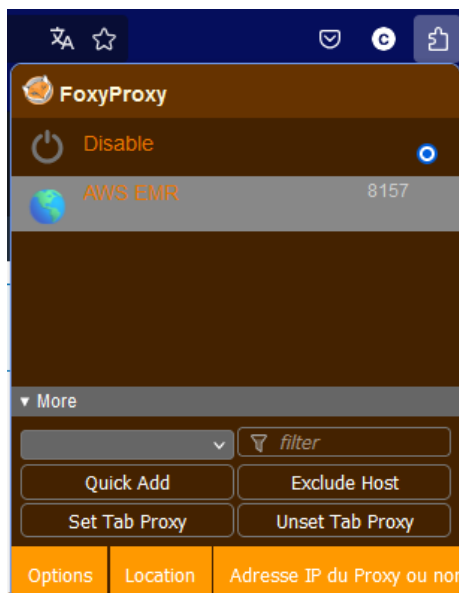
L'affichage de l'écran EMR confirme le succès de la connexion :

[illegible]

4.8.3. Activation du proxy

Si l'ouverture du tunnel permet une interaction avec le serveur en ligne de commande, il faut indiquer au navigateur utilisé en local d'utiliser ce tunnel pour accéder aux interfaces de JupyterHub et Serveur d'historique Spark. C'est l'intérêt du logiciel proxy. Amazon EMR suggère l'utilisation de l'extension Chrome / Firefox **FoxyProxy** et fourni les détails de configuration ici : [Configure proxy settings to view websites hosted on the primary node](#).

Une fois cette configuration réalisée, il est nécessaire d'activer le proxy en utilisant l'extension :



La configuration enregistrée permet l'accès via SSH aux interfaces **JupyterHub** et **Serveur d'historique Spark**, dont les liens sont fournis sur la console du cluster :

Propriétés | Actions d'amorçage | Instances (Matériel) | Étapes | **Applications** | Configurations | Surveillance | Évènements | identifications (1)

Interfaces utilisateur d'application [Info](#)
Les applications installées sur votre cluster Amazon EMR publient des interfaces utilisateur en tant que sites web. Vous pouvez les utiliser pour surveiller l'activité du cluster.

☒ **Interfaces utilisateur d'application sur le cluster**
Les interfaces utilisateur sur le cluster sont disponibles uniquement pendant l'exécution de votre cluster. Utilisez les liens suivants pour démarrer. Pour accéder à toutes les interfaces utilisateur d'application, configurez le tunneling SSH.

☐ **Interfaces utilisateur d'application persistantes**
Les interfaces utilisateur persistantes ne nécessitent pas de tunneling SSH. Elles sont hébergées hors du cluster et sont disponibles pendant 30 jours après la fin d'une application.

Interfaces utilisateur d'application en direct
Ces interfaces utilisateur d'application sur cluster sont disponibles sans tunneling SSH.
[Interfaces utilisateur d'application](#)
[Interface utilisateur du serveur d'historique Spark](#)

Interfaces utilisateur d'application sur le nœud primaire
Celles-ci nécessitent l'activation du tunneling SSH.


[Activer une connexion SSH](#)

Application	URL de l'interface utilisateur
Gestionnaire de ressources	http://ec2-54-74-18-254.eu-west-1.compute.amazonaws.com:8088/
JupyterHub	https://ec2-54-74-18-254.eu-west-1.compute.amazonaws.com:9443/
Nom du nœud HDFS	http://ec2-54-74-18-254.eu-west-1.compute.amazonaws.com:9870/
Serveur d'historique Spark	http://ec2-54-74-18-254.eu-west-1.compute.amazonaws.com:18080/

4.9. Connexion à JupyterHub

JupyterHub est un serveur Jupyter utilisé pour partager des notebooks. Dans l'environnement Amazon EMR, il est préconfiguré avec le kernel PySpark, et donc parfaitement adapté au projet.

Il est nécessaire de confirmer l'accès à un site non sécurisé (protocole http) avant de se connecter avec les identifiant/mot de passe par défaut : **jovyan/jupyter**



Sign in

Username:

jovyan

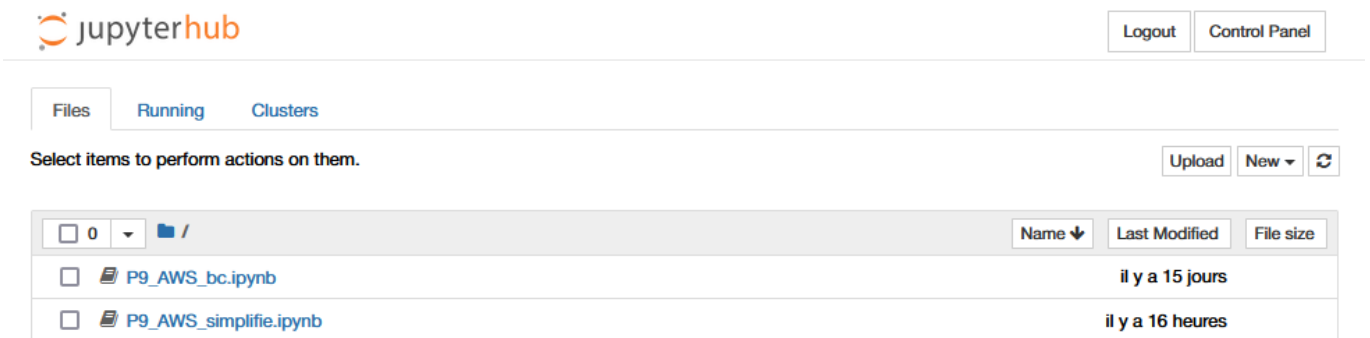
Password:

.....

Sign in

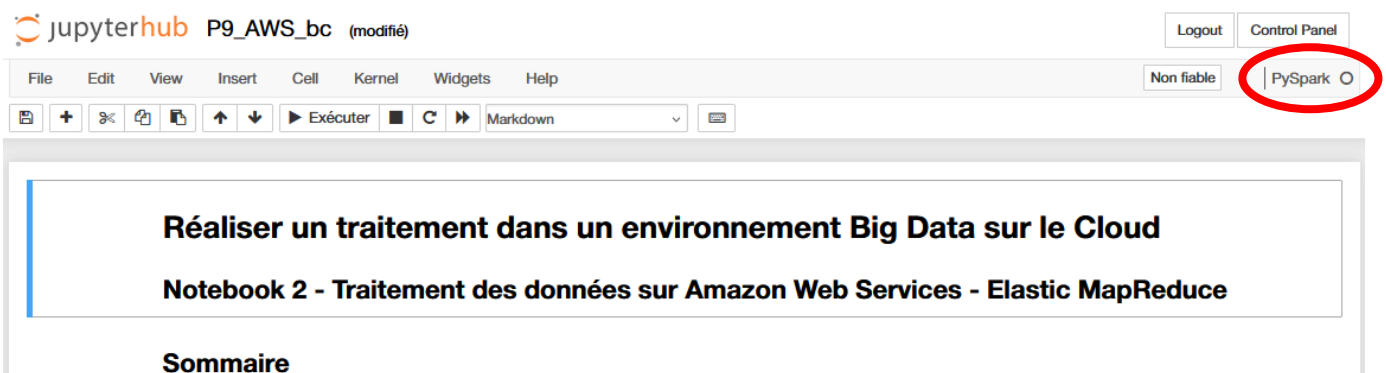
Grâce à la persistance paramétrée à l'instanciation sur cluster, le point d'entrée des fichiers se trouve dans l'arborescence du bucket S3 déclaré lors de la configuration initiale : **s3://cap9-fruits/jupyter/jovyan**

Les notebooks contenant le code du projet ont été transférés à cet emplacement.



4.10. Exécution du code

Afin de profiter de l'environnement PySpark d'Amazon EMR il est nécessaire de spécifier l'utilisation du kernel PySpark au chargement du notebook :



Utiliser JupyterHub préinstallé par AWS présente un certain nombre d'avantages :

- Kernel **PySpark** préinstallé : démarrage d'une session Spark automatique dès l'exécution de la première cellule de code (même s'il s'agit d'une cellule contenant uniquement un commentaire) avec création d'un SparkContext
- Mise à disposition de nombreuses commandes **Sparkmagic**

Deux versions de code ont été testées :

- La première (notebook **P9_AWS_bc.ipynb**) est identique à celle testée en local avec notamment :
 - Préparation du modèle
 - Chargement du modèle MobileNetV2 avec les poids pré-entraînés imagenet
 - Suppression de la dernière couche
 - Broadcast des poids aux workers
 - Extraction des features
 - Chargement du modèle MobileNetV2 avec les poids pré-entraînés imagenet
 - Gel des poids des couches pré-entraînées
 - Suppression de la dernière couche
 - Affectation des poids broadcastés initialement
 - Prédictions
- La seconde (notebook **P9_AWS_simplifie.ipynb**) est une version simplifiée, sans broadcast des poids aux workers, ni gel des couches pré-entraînées. La [documentation de MobileNetV2](#) montre que le modèle peut être chargé sans la dernière couche. De plus, le gel des poids est inutile dans le cadre de ce projet, puisqu'il n'est utilisé que pour faire des prédictions à partir des poids pré-entraînés (Transfer Learning). La phase de préparation du modèle n'a plus lieu d'être :
 - Extraction des features
 - Chargement du modèle MobileNetV2 avec les poids pré-entraînés imagenet, sans la dernière couche
 - Prédictions

4.11. Suivi de l'avancement des tâches

Pendant l'exécution du notebook une barre de progression permet de suivre l'avancée de l'exécution du code d'une cellule, ce qui est particulièrement intéressant pour les cellules prenant plus de temps à s'exécuter :

Enregistrement au format 'parquet' :

```
Entrée [*]: pca_features_df.write.mode("overwrite").parquet(PATH_Result)
```

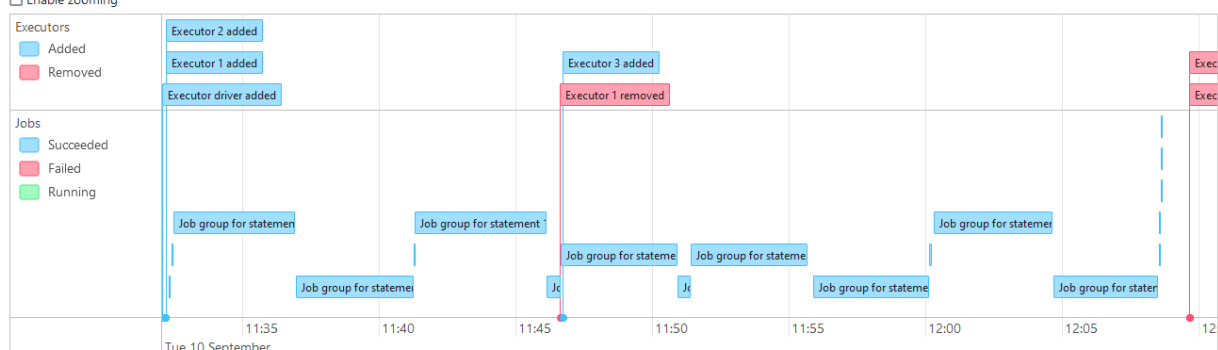
Progress:

Le Serveur d'historique Spark permet de suivre l'avancement des opérations Spark pendant l'exécution du notebook. L'accès à ce serveur reste opérationnel même après la résiliation du cluster (c'est une extension de SparkUI).

Spark Jobs (?)

User: livy
Total Uptime:
Scheduling Mode: FIFO
Completed Jobs: 19

▼ Event Timeline
☐ Enable zooming



Chaque opération est chronométrée :

▼ Completed Jobs (19)

Page: 11 Pages. Jump to 1. Show 100 items in a page. Go

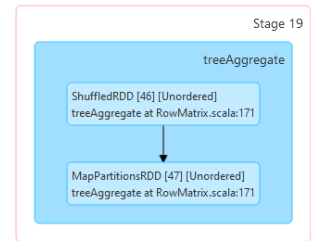
Job Id (Job Group) ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
18 (27)	Job group for statement 27 first at <stdin>:5	2024/09/10 12:08:37	0.4 s	1/1	1/1
17 (27)	Job group for statement 27 count at NativeMethodAccessorImpl.java:0	2024/09/10 12:08:37	39 ms	1/1 (1 skipped)	1/1 (5 skipped)
16 (27)	Job group for statement 27 count at NativeMethodAccessorImpl.java:0	2024/09/10 12:08:37	0.2 s	1/1	5/5
15 (26)	Job group for statement 26 showString at NativeMethodAccessorImpl.java:0	2024/09/10 12:08:32	0.5 s	1/1	1/1
14 (25)	Job group for statement 25 parquet at NativeMethodAccessorImpl.java:0	2024/09/10 12:08:32	0.2 s	1/1	1/1
13 (24)	Job group for statement 24 parquet at NativeMethodAccessorImpl.java:0	2024/09/10 12:04:40	3.8 min	1/1 (1 skipped)	16/16 (709 skipped)
12 (24)	Job group for statement 24 parquet at NativeMethodAccessorImpl.java:0	2024/09/10 12:00:17	4.4 min	1/1	709/709
11 (22)	Job group for statement 22 showString at NativeMethodAccessorImpl.java:0	2024/09/10 12:00:08	5 s	1/1 (1 skipped)	1/1 (709 skipped)
10 (22)	Job group for statement 22 showString at NativeMethodAccessorImpl.java:0	2024/09/10 11:55:52	4.3 min	1/1	709/709
9 (16)	Job group for statement 16 treeAggregate at RowMatrix.scala:171	2024/09/10 11:51:23	4.3 min	2/2 (1 skipped)	20/20 (709 skipped)
8 (16)	Job group for statement 16 isEmpty at RowMatrix.scala:441	2024/09/10 11:50:54	29 s	1/1 (1 skipped)	1/1 (709 skipped)
7 (16)	Job group for statement 16 treeAggregate at Statistics.scala:58	2024/09/10 11:46:38	4.3 min	2/2 (1 skipped)	20/20 (709 skipped)
6 (16)	Job group for statement 16 first at RowMatrix.scala:62	2024/09/10 11:46:06	32 s	1/1 (1 skipped)	1/1 (709 skipped)
5 (16)	Job group for statement 16 first at PCA.scala:44	2024/09/10 11:41:18	4.8 min	2/2	710/710
4 (13)	Job group for statement 13 first at StandardScaler.scala:113	2024/09/10 11:41:15	0.1 s	1/1 (2 skipped)	1/1 (725 skipped)
3 (13)	Job group for statement 13 first at StandardScaler.scala:113	2024/09/10 11:36:55	4.3 min	1/1 (1 skipped)	16/16 (709 skipped)
2 (13)	Job group for statement 13 first at StandardScaler.scala:113	2024/09/10 11:32:27	4.5 min	1/1	709/709
1 (6)	Job group for statement 6 showString at NativeMethodAccessorImpl.java:0	2024/09/10 11:32:24	0.8 s	1/1	1/1
0 (5)	Listing leaf files and directories for 131 paths: s3://cap9-fruits/img/Apple Braeburn, ... load at NativeMethodAccessorImpl.java:0	2024/09/10 11:32:17	4 s	1/1	131/131

Un suivi très détaillé peut être obtenu pour chaque opération :

Details for Stage 19 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 0.3 s
Locality Level Summary: Node local: 4
Shuffle Read Size / Records: 100.1 MiB / 16
Associated Job Ids: 9

▼ DAG Visualization

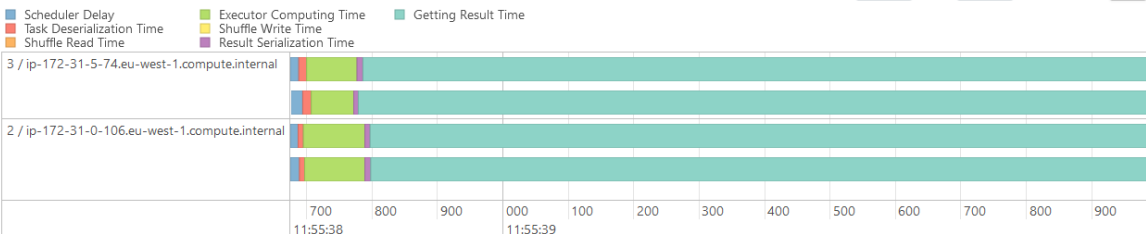


► Show Additional Metrics

▼ Event Timeline

☐ Enable zooming

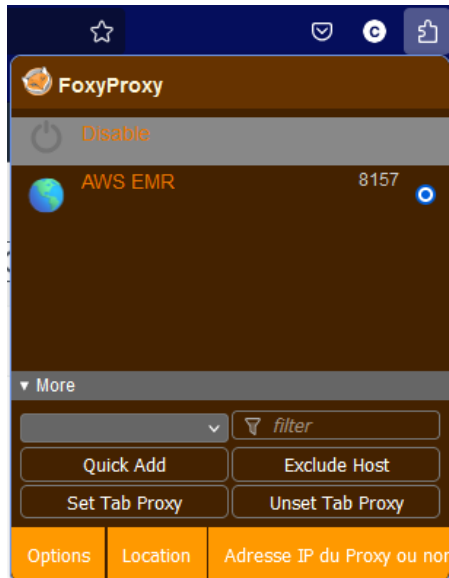
Tasks: 4. 1 Pages. Jump to 1. Show 4 items in a page. Go



4.12. Résiliation du cluster EMR

Pour optimiser les coûts (AWS facture à la demande, même lorsque les machines virtuelles ne sont pas sollicitées), il est important de résilier le cluster, ce qui déclenchera la suppression des instances EC2.

1. Désactivation du tunnel SSH via l'extension FoxyProxy



2. Résiliation du cluster via l'interface EMR

Résilier

Amazon EMR > EMR sur EC2: Clusters > cap9_cluster_bc

cap9_cluster_bc Mis à jour il y a 10 minutes **Résilier** Cloner dans AWS CLI Cloner

▼ Récapitulatif

Informations sur le cluster	Applications	Gestion des clusters	Statut et heure
ID de cluster j-1JPH7YPS2435	Version d'Amazon EMR emr-7.2.0	Destination des journaux dans Amazon S3 aws-logs-825765415545-eu-west-1/elasticmapreduce	Statut En attente
Configuration de cluster Groupes d'instances	Applications installées JupyterHub 1.5.0, Spark 3.5.1	Interfaces utilisateur d'application persistantes Serveur d'historique Spark Serveur de chronologie YARN	Heure de création 24 septembre 2024 08:00 (UTC+02:00)
Capacité 1 primaire(s) 2 unité(s) principale(s) 0 tâche(s)		DNS public du nœud primaire ec2-3-253-196-82.eu-west-1.compute.amazonaws.com Connexion au nœud primaire à l'aide de SSH Connexion au nœud primaire à l'aide de SSM	Temps écoulé 33 minutes, 42 secondes

Propriétés | Actions d'amorçage | Instances (Matériel) | Étapes | Applications | Configurations | Surveillance | Évènements | identifications (1)

Après validation de la confirmation, le statut se met à jour :

Statut et heure	Statut et heure
Statut En cours de résiliation	Statut Résilié

4.13. Clonage éventuel du cluster EMR

Pour recréer le cluster EMR à l'identique, deux options sont disponibles :

Résilier

Cloner dans AWS CLI

Cloner

Génération de la commande AWS CLI de recréation du cluster à l'identique, à copier-coller dans un terminal.

Ouverture de la page de configuration d'un cluster, avec reprise de toutes les options et réglages du cluster.

Commande CLI pour cloner cap9_cluster_simplifie

Pour cloner ce cluster en dehors de la console, copiez la commande et utilisez-la avec AWS CLI.

```
1 aws emr create-cluster \  
2 --name "cap9_cluster_simplifie" \  
3 --log-uri "s3://aws-logs-825765415545-eu-west-1/elasticmapreduce" \  
4 --release-label "emr-7.2.0" \  
5 --service-role "arn:aws:iam::825765415545:role/service-role/AmazonEMR-ServiceRole-20240831T002026" \  
6 --unhealthy-node-replacement \  
7 --ec2-attributes '{"InstanceProfile":"AmazonEMR-InstanceProfile-20240831T002010","EmrManagedMasterSecurityGroup":"sg-09fb8512919c647a0","EmrManagedSlaveSecurityGroup":"sg-0d25715af52eea6d7","KeyName":"christophe-ras-ec2-2","AdditionalMasterSecurityGroups":["sg-063f006dcd5983e22"],"AdditionalSlaveSecurityGroups":["sg-063f006dcd5983e22"],"SubnetId":"subnet-0522a511399972204"}' \  
8 --tags 'for-use-with-amazon-emr-managed-policies=true' \  
9 --applications Name=JupyterHub Name=Spark \  
10 --configurations '[{"Classification":"jupyter-s3-conf","Properties":{"s3.persistence.bucket":"cap9-fruits","s3.persistence.enabled":"true"}}]' \  
11 --instance-groups '[{"InstanceCount":2,"InstanceGroupType":"TASK","Name":"Tâche","InstanceType":"m5.xlarge","EbsConfiguration":{"EbsBlockDeviceConfigs":[{"VolumeSpecification":{"VolumeType":"gp2","SizeInGB":32},"VolumesPerInstance":2}]}],{"InstanceCount":1,"InstanceGroupType":"CORE","Name":"Unité principale"}]'
```

Copier

Powershell Ln 1, Col 1 0 0

Fermer

4.14. Validation des résultats

La sortie de la dernière cellule des notebooks des deux versions est strictement identique :

Vérification du nombre d'images traitées, et des nombres de features extraites avant et après PCA :

```
# Nombre de lignes du DataFrame = nombre d'images traitées  
num_rows = df_spark.count()  
  
# Première ligne du DataFrame  
first_row = df_spark.select("features", "pca_features").first()  
  
# Taille des vecteurs 'features' et 'pca_features'  
features_size = len(first_row["features"])  
pca_features_size = len(first_row["pca_features"])  
  
# Afficher les tailles des vecteurs  
print(f"Nombre d'images : {num_rows}\n")  
print(f"Taille du vecteur 'features' : {features_size}\n")  
print(f"Nombre de composantes expliquant 90 % de la variance : {num_components}")  
print(f"Taille du vecteur 'pca_features' : {pca_features_size}")  
  
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),...  
  
Nombre d'images : 22688  
  
Taille du vecteur 'features' : 1280  
  
Nombre de composantes expliquant 90 % de la variance : 342  
Taille du vecteur 'pca_features' : 342
```

Ces données confirment que les **22 688 images ont bien été traitées**, la bonne prise en compte de l'avant dernière couche du modèle avec les **1 280 features par image**, et qu'après réduction de dimensionnalité chaque image est caractérisée par un **vecteur de dimension 342**.

Pour être certain que les deux méthodes aboutissent aux mêmes résultats, il faut comparer les deux fichiers de caractéristiques. Cette validation est l'objet du notebook **P9_AWS_Validation.ipynb** qui montre une similitude parfaite entre les deux versions, à l'exception du vecteur de caractéristiques des images après réduction de dimension. Tous les vecteurs ont des valeurs différentes.

Après examen des vecteurs d'une image, il s'avère qu'il s'agit de différences d'arrondi. Un test de similarité avec une tolérance à 5 chiffres après la virgule montre une correspondance parfaite entre les vecteurs, ce qui permet d'affirmer que **les deux méthodes sont équivalentes en termes de résultats**.

5. Conclusion

Ce projet a été réalisé en deux temps, en tenant compte des contraintes imposées.

5.1. Implémentation en environnement local

Dans un premier temps, la solution a été développée en local. Pour des raisons de simplicité, le notebook a été exécuté sur Google Colab avec l'option `SparkSession.master('local')` de manière à ce que tout soit exécuté sur une seule et unique machine. Cela a permis de **simuler le calcul partagé** en considérant **chaque cœur d'un processeur comme un worker indépendant**.

La technique du **Transfer Learning** a été utilisée à partir du modèle **MobileNetV2**. Ce modèle a été retenu pour sa **légèreté** et sa **rapidité d'exécution** ainsi que pour la **faible dimension de son vecteur en sortie**.

L'idée étant de valider le bon fonctionnement de la solution, l'extraction des vecteurs de caractéristiques n'a porté que sur un jeu de données limité (494 images).

La solution initialement fournie a été complétée par l'ajout d'une **réduction de dimensionnalité via PCA, en PySpark**.

Les résultats ont été enregistrés sur Google Drive en plusieurs partitions au format 'Parquet'.

La solution a parfaitement fonctionné en mode local.

5.2. Implémentation sur Amazon AWS

Dans un second temps, la solution a été implémentée sur un cluster de machine. L'objectif était de pouvoir anticiper une augmentation du nombre d'images à traiter et donc de la charge de travail.

Le meilleur choix retenu a été l'utilisation du prestataire de services **Amazon Web Services** qui propose notamment :

- de **louer à la demande de la puissance de calcul** (serveurs virtuels). Ce service se nomme **EC2** (Elastic Compute Cloud) et se classe parmi les offres **Infrastructure as a Service** (IaaS)
- de **stocker des fichiers** sur un espace de taille illimitée, via AWS **S3** (Simple Storage Service)
- de **louer à la demande une infrastructure Hadoop gérée**. Ce service se nomme **EMR** (Elastic MapReduce) et se classe parmi les offres **Platform as a Service** (PaaS)

L'utilisation de la plateforme **EMR** a permis de construire un cluster en spécifiant les options suivantes :

- Location de 3 instances EC2 (1 driver, 2 workers)
- Installation et configuration automatique de Spark et JupyterHub sur le driver
- Installation automatique des librairies complémentaires sur les workers (notamment une version récente de TensorFlow)
- Connexion au compartiment S3

L'ensemble des images du jeu de données Test a pu être traité dans un environnement de calcul distribué sur le Cloud, avec une facturation raisonnable corrélée à l'utilisation de la plateforme AWS. Les possibilités de mise à l'échelle qu'offre AWS sont nombreuses (nombre d'instances EC2, mémoire, processeurs...) et permettront de faire à une montée en charge importantes des images à traiter.