« Déployez un modèle dans le cloud»

Soutenance de Projet n°8

Alexandre PRÉVOT – Renel Cadin DEMAROU

6 octobre 2021

Ordre du jour

0.0 Contexte de travail

- Outils utilisés
- Ressources consultées
- Compétences évaluées

0.1 Problématique et objectifs

- Classification d'images de fruits
- Mise en place d'un environnement Big Data (pré-traitement)

1. Contexte et solution envisagée

- Les données
- Chaîne de traitement choisie
- Problématique Big Data
- Distribution des calculs

2. Chaîne de prétraitement

- Principes généraux du code
- Notebook sur instance EC2
- Détail des étapes de calcul
- Implémentation du script à exécuter pour EMR

3. Architecture cloud choisie

- Vue globale de l'architecture
- Exécution du script sur cluster EMR
- Accès aux résultats (Notebook EMR, log out)
- Monitoring (Ganglia)

0.0 Contexte de travail

Compétences développées

- Paralléliser des opérations de calcul avec Pyspark
- Utiliser les outils du cloud pour manipuler des données dans un environnement Big Data
- Identifier les outils du cloud permettant de mettre en place un environnement Big Data

Ressources documentaires

- Cours OpenClassrooms:
  - « Réalisez des calculs distribués sur des données massives »
  - « Lancez une session de notebook Jupyter sur AWS »
- Documentation d'AWS: S3, EC2, EMR, IAM:
  - https://docs.aws.amazon.com/
- Tutoriels d'utilisation de Pyspark:
  - https://sparkbyexamples.com/pyspark-tutorial/
- Compréhension architecture Spark:
  - https://meritis.fr/larchitecture-framework-spark/

Outils utilisés

0.1 Rappel de la problématique

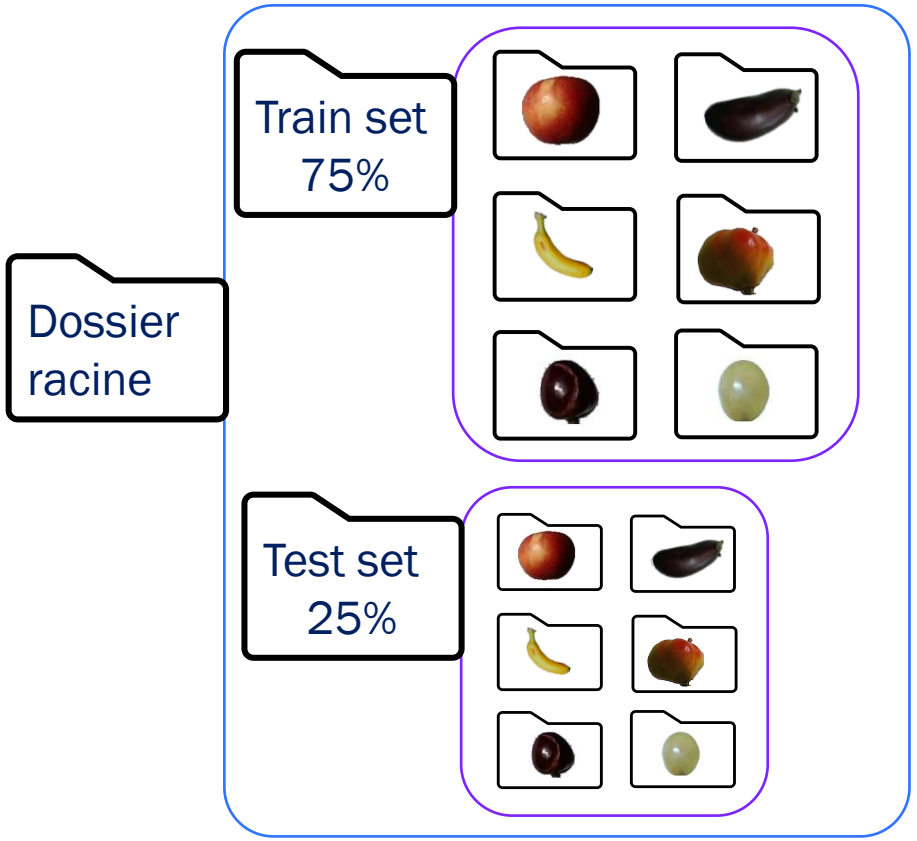
Il s'agit de réaliser une étude de faisabilité de la classification dans un environnement de calcul distribué, propice au **traitement de données massives** ! Il sera nécessaire de **mettre en place les premières briques de cet environnement dans le Cloud.**

- Contexte :
  - **Fruits!** propose de développer un application permettant de scanner des fruits afin d'en obtenir des informations
  - **Objectif: Entraîner un modèle de classification d'image.**
  - **Problème: Volume de données trop important.**
- Action à entreprendre:
  - **Mettre en place un environnement Big Data permettant une première chaîne de traitement des données:**
    - **Preprocessing**
    - **Réduction dimensionnelle**

1. Contexte et solution envisagée

Les données

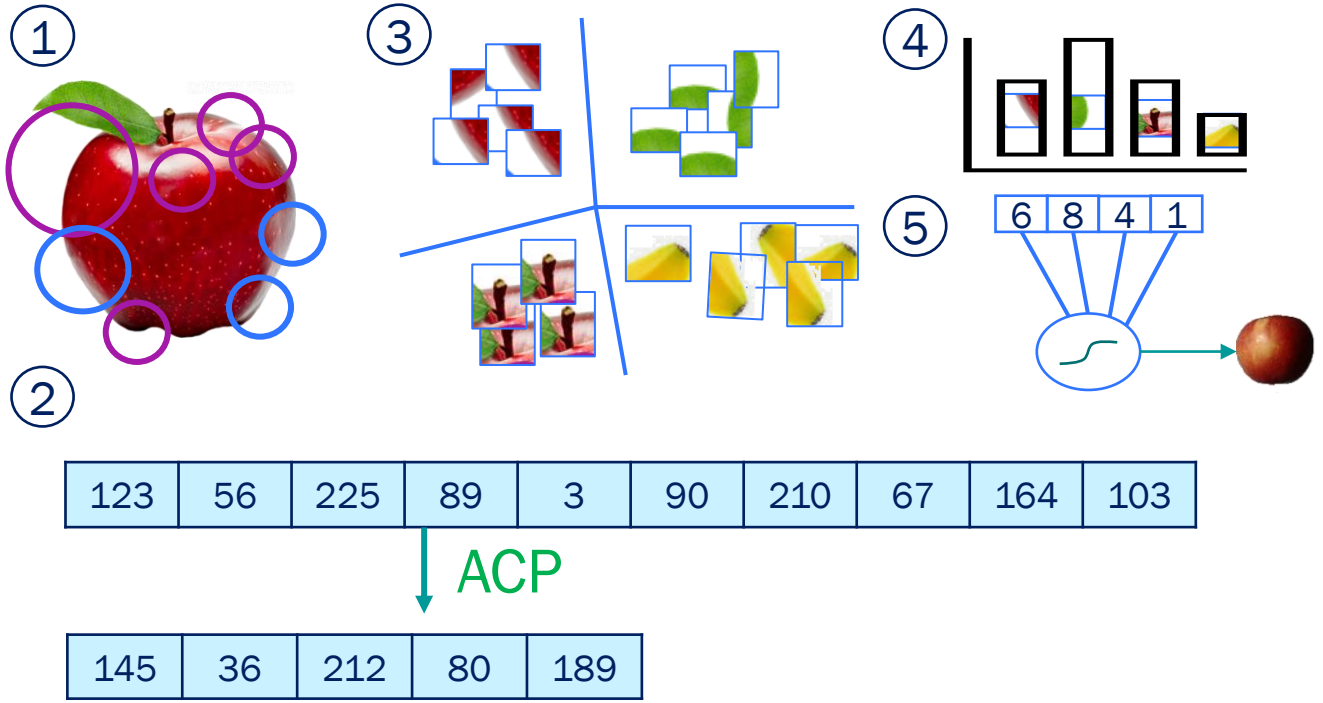
- 90 000 images de fruits
- Un dossier par type de fruit (131 dossiers)
- Tailles d'images variables



Traitements à effectuer

Objectif: Classification d'images

- **1. Détection des points d'intérêts de l'image avec ORB**
- **2. Réduction dimensionnelle des descripteurs (ACP)**
- 3. Attribution d' « Visual Word » à chaque point d'intérêt (K-means)
- 4. Bag of « Visual Words » pour chaque image (fréquence d'apparition)
- 5. Classification (régression logistique)



1. Contexte et solution envisagée

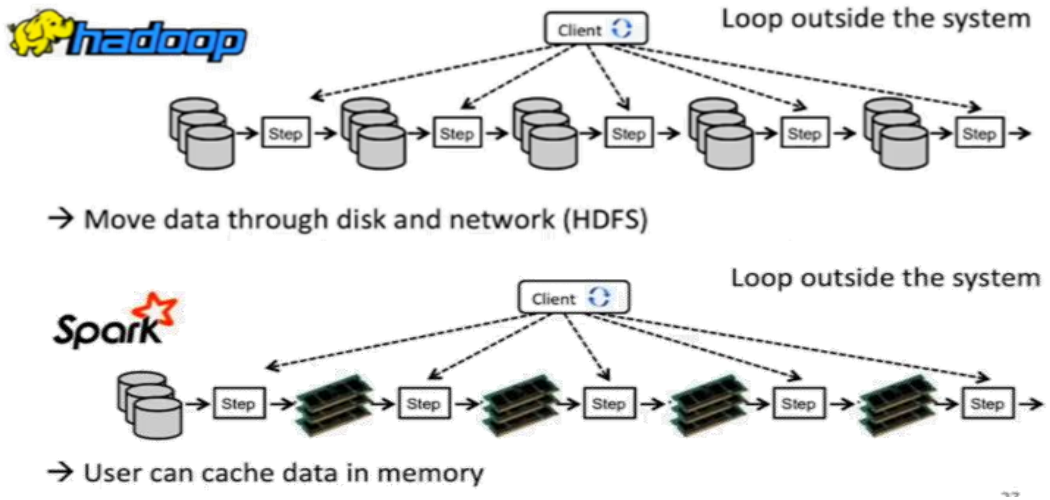
Nécessité d'un environnement Big Data

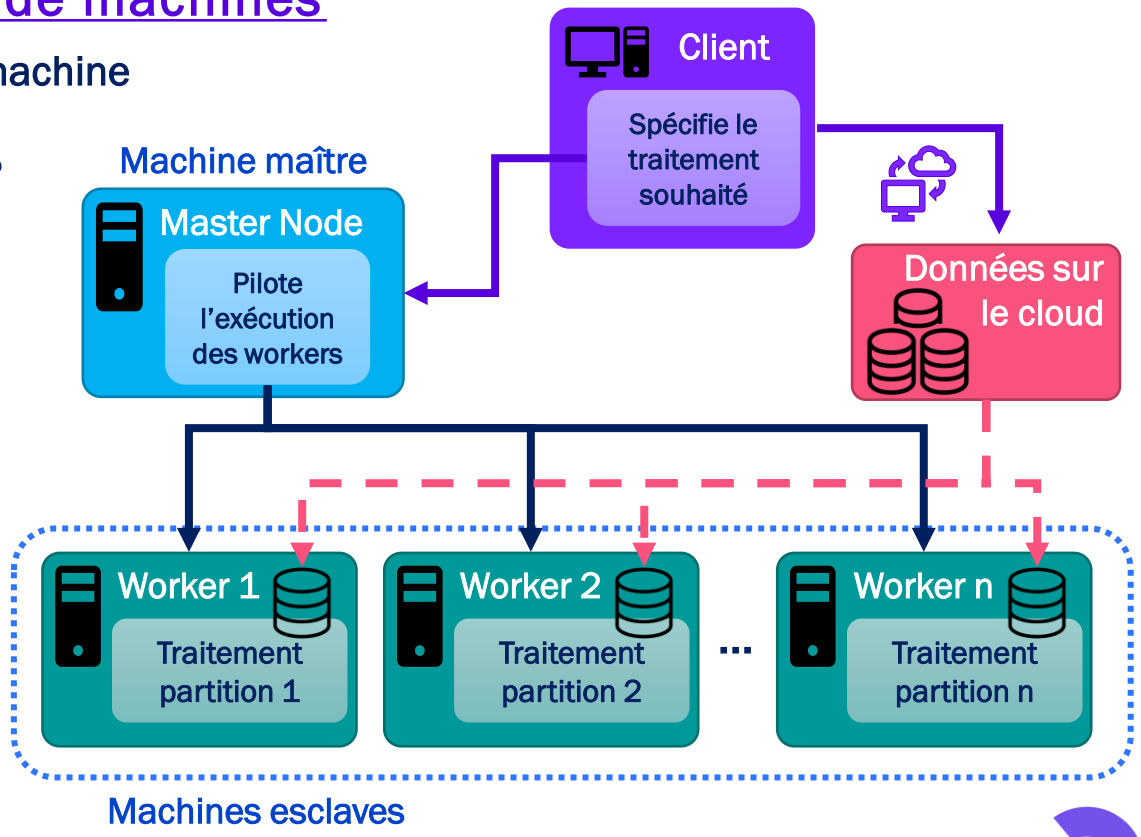
① Volume de données important
- Permettre le stockage dans le Cloud

② Besoin de rapidement traiter ces données
- Threads de calcul en parallèle sur une machine ? insuffisant

Solution: distribution des calculs sur un cluster de machines

- Parallélisation **horizontale: répartir les données sur plusieurs machine**
- Architecture maître-esclaves: pour la gestion des ressources
- Framework de calcul distribué **Spark** ou **Hadoop MapReduce ?**

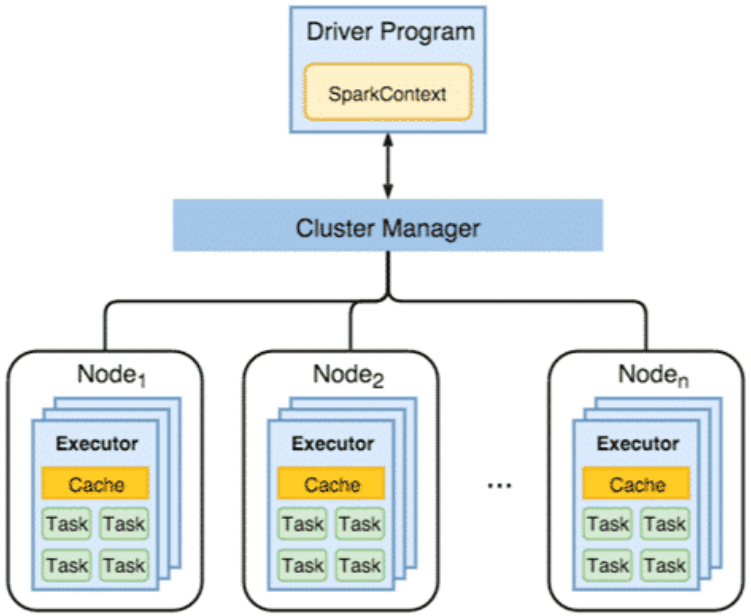




1. Contexte et solution envisagée

Fonctionnement de Spark

- **Driver** ≈ Machine Maître
- **SparkContext:** Point d'accès aux fonctionnalités de Spark dans le cluster
- **Cluster Manager**: chef d'orchestre des workers
- **Executors:** Exécution de tâches (processus JAVA)



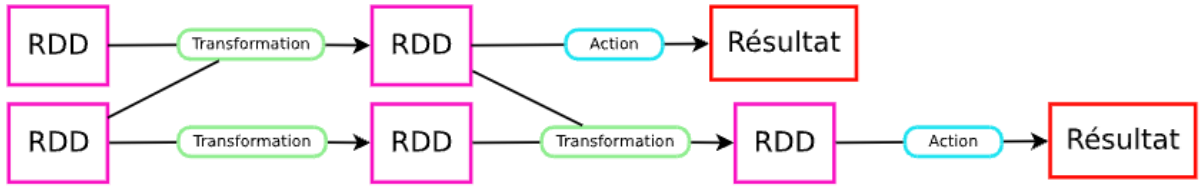
Actions et transformation Spark

- RDD (Resilient Distributed Dataset) / Dataframe Spark

- Les RDD dictent la manière dont les calculs sont distribués:
  - **Répartition** des données sur les executors (en parititions)
  - **Duplication** des partitions

- Le concept de « *Lazy Evaluation* » autorise deux traitements sur les RDD ou DF
  - **Transformation** (RDD ⟶ RDD)
  - **Action** (RDD ⟶ Résultat),ex: lecture / écriture d'un DF transformé

  Représentation en **graphe acyclique orienté**

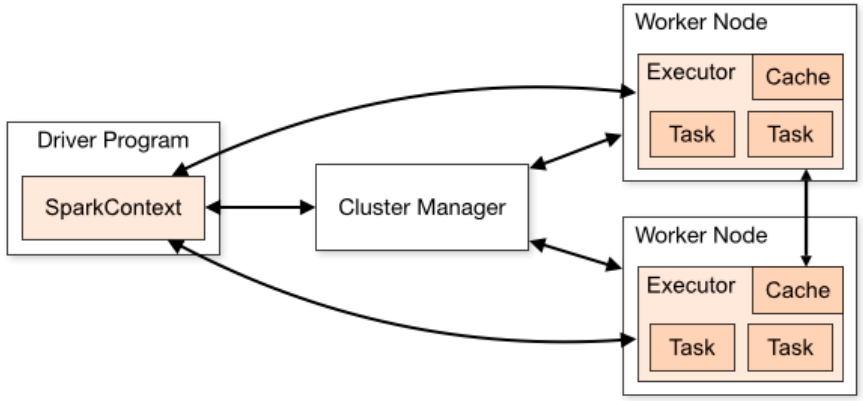


- Les traitements Spark ne sont **exécutés que lorsqu'une action est demandée**
- Tolérance aux pannes: retraçage des transformations possible pour chaque partition

2. Chaîne de prétraitement

Étapes générales du code

① Initialisation de l'environnement de calcul distribué



② Import des images dans un dataframe Spark

Chemin d'accès	Label	Caractéristiques	Données
s3://../Image_1	Pomme	260x234 pixels	[[[14,255,430....]
S3://../Image_2	Raisin	322x240pixels	[[[123,10,100....]
...	...	...	...
S3://..Image_n	Banane	300x210pixels	[[[230,50,112....]

③ Extraction de descripteurs ORB

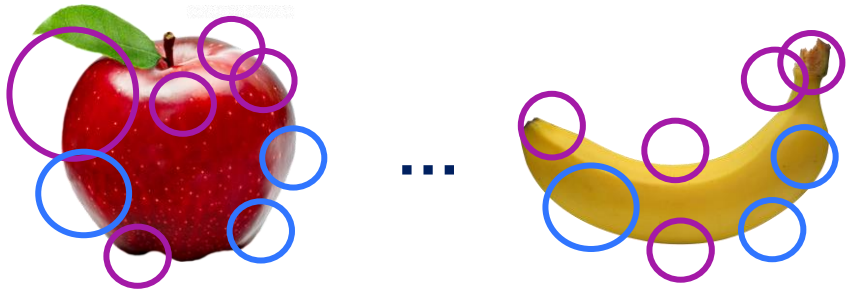


Image_1 ... Image_n

④ Réduction dimensionnelle (ACP)

123	56	225	89	3	90	210	67	164	103

ACP

145	36	212	80	189

⑤ Export du dataframe transformé sur le Cloud (S3)

2. Chaîne de prétraitement

Réalisation du prétraitement sur une instance EC2

- Le Notebook 'Notebook-EC2' a été exécuté sur une instance EC2
- Code en PySpark: nécessite Spark 2.4.7 avec Hadoop 2.7, et Java 8
- Étude de faisabilité sur 25 images de 5 fruits différents.

① Initialisation de l'environnement Spark (S3)

```
spark = (SparkSession.builder.appName("Prevot-P8")\
         .config('spark.hadoop.fs.s3a.impl',
                 'org.apache.hadoop.fs.s3a.S3AFileSystem')\
         .getOrCreate())

# SparkContext est le point d'accès à toutes les fonctionnalités de Spark,
# Il est contenu dans la Spark Session
sc = spark.sparkContext
sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint", "s3.us-east-2.amazonaws.com")
sc.setSystemProperty('com.amazonaws.services.s3.enableV4', 'true')
```


2. Chaîne de prétraitement

② Import des images dans un dataframe Spark

- Accès aux données avec via client boto3
- Création de dataframes Spark (un par dossier de fruit)
  1. Import des images avec **pyspark.ml.ImageSchema.readImage**
  2. Affectation du label avec **pyspark.sql.functions.lit**
- **Fusion des dataframes spark en un seul**

Label	image		
	Chemin d'accès	Caractéristiques	Données
Pomme	s3://../Image_1	260x234 pixels	[[[14,255,430....]
Raisin	S3://../Image_2	322x240pixels	[[[123,10,100....]
...	...	...	...
...	...	...	...
Banane	S3://..Image_n	300x210pixels	[[[230,50,112....]

```
df.printSchema()
```

```
root
 |-- image: struct (nullable = true)
 |    |-- origin: string (nullable = true)
 |    |-- height: integer (nullable = false)
 |    |-- width: integer (nullable = false)
 |    |-- nChannels: integer (nullable = false)
 |    |-- mode: integer (nullable = false)
 |    |-- data: binary (nullable = false)
 |-- label: string (nullable = false)
```

Chaque ligne du dataframe correspond à une image. Pour chaque image nous disposons de:

- Première colonne (image): origin = chemin d'accès au fichier sur s3
- Première colonne (image): height = hauteur (pixels)
- Première colonne (image): width = largeur (pixels)
- Première colonne (image): nChannels = 3 (RVB)
- Première colonne (image): mode = format d'encodage de l'image
- Première colonne (image): data = tableau contenant l'image encodée en binary
- Deuxième colonne (label): Etiquette de l'image (fruit)

2. Chaîne de prétraitement

③ Extraction des descripteurs ORB

- Création d'une fonction permettant d'extraire les descripteurs ORB d'une image:
  1. Mettre l'image au format tableau np.array
  2. Importer descripteur ORB d'OpenCV
  3. Extraction ORB sur l'image
  4. Récupération d'un liste de descripteurs

- Application de cette fonction à chaque ligne du Spark Dataframe. Cette **transformation** est possible grâce à **pyspark.sql.functions.udf: (User defined function)**

- Chaque image possède un nombre variable de descripteur (30 maximum)

- Chaque descripteur contient 32 composantes

Label	image			Descripteurs ORB
	Chemin d'accès	Caractéristiques	Données	
Pomme	s3://../Image_1	260x234 pixels	[[[14,255,430....]	[[24, 145,...]]
Raisin	S3://../Image_2	322x240pixels	[[[123,10,100....]	[[34, 210,...]]
...	...	...	...	...
...	...	...	...	...
Banane	S3://..Image_n	300x210pixels	[[[230,50,112....]	[[121, 15,...]]

```
root
 |-- image: struct (nullable = true)
 |    |-- origin: string (nullable = true)
 |    |-- height: integer (nullable = false)
 |    |-- width: integer (nullable = false)
 |    |-- nChannels: integer (nullable = false)
 |    |-- mode: integer (nullable = false)
 |    |-- data: binary (nullable = false)
 |-- label: string (nullable = false)
 |-- Descripteurs ORB: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: integer (containsNull = true)
```

2. Chaîne de prétraitement

③ Extraction des descripteurs ORB

- Explosion avec **pyspark.sql.functions.udf** afin d'obtenir **une ligne par descripteur**
- **Vectorisation** (**pyspark.ml.VectorUDT**) des descripteurs pour permettre au module de machine Learning de Spark de lire les données

④ Réduction dimensionnelle (ACP)

- Utilisation de **pyspark.ml.feature.PCA** pour créer un modèle d'ACP sur lequel nous allons « fit » les descripteurs.
- **Transformation des vecteurs (réduction en 20 composantes principales**

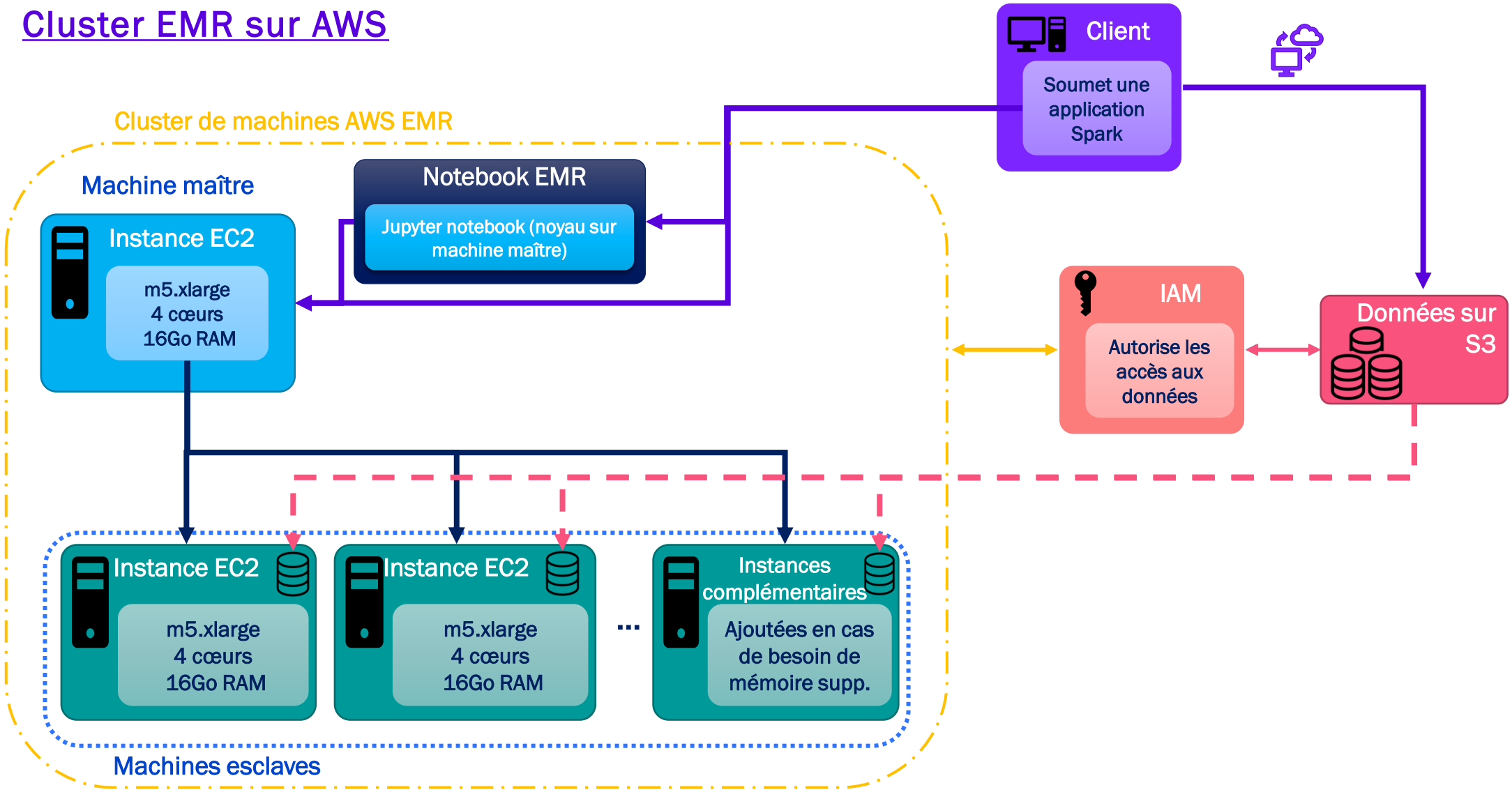
⑤ Écriture du DF transformé sur S3

Desc ORB (Vecteur 32 comp.)	Label	image		
		Chemin d'accès	Caractéristiques	Données
[24, 145,...]	Pomme	s3://../Image_1	260x234 pixels	[[[14,255,430....]
[12, 56,...]	Pomme	S3://../Image_1	455x210pixels	[[[34,23,100....]
...	...	...	...	...
[161, 235,...]	Banane	S3://..Image_n	300x210pixels	[[[230,50,112....]

Desc ORB (Vecteur 20 comp.)	Label	image		
		Chemin d'accès	Caractéristiques	Données
[24, 145,...]	Pomme	s3://../Image_1	260x234 pixels	[[[14,255,430....]
[12, 56,...]	Pomme	S3://../Image_1	455x210pixels	[[[34,23,100....]
...	...	...	...	...
[161, 235,...]	Banane	S3://..Image_n	300x210pixels	[[[230,50,112....]

3. Architecture Cloud choisie

Cluster EMR sur AWS



3. Architecture Cloud choisie

Création du Cluster EMR

1

Configuration des logiciels

Libérer emr-5.31.1

☑ Hadoop 2.10.0	☐ Zeppelin 0.8.2	☑ Livy 0.7.0
☐ JupyterHub 1.1.0	☐ Tez 0.9.2	☐ Flink 1.11.0
☐ Ganglia 3.7.2	☐ HBase 1.4.13	☐ Pig 0.17.0
☐ Hive 2.3.7	☐ Presto 0.238.3	☐ ZooKeeper 3.4.14
☐ MXNet 1.6.0	☐ Sqoop 1.4.7	☐ Mahout 0.13.0
☐ Hue 4.7.1	☐ Phoenix 4.14.3	☐ Oozie 5.2.0
☑ Spark 2.4.6	☐ HCatalog 2.3.7	☐ TensorFlow 2.1.0

2

Modifier les paramètres du logiciel

◉ Entrer la configuration ○ Charger JSON à partir de S3

```
[{"configurations":[{"classification":"export","properties":
{"PYSPARK_PYTHON":"/usr/bin/python3"}}],"classification":"spark-env","properties":{}}]
```

3

Ajouter des étapes (facultatif)

Une étape est une unité de travail que vous soumettez au cluster. Par exemple, une étape peut contenir une ou plusieurs tâches Hadoop ou Spark. Vous pouvez également soumettre des étapes supplémentaires à un cluster après le début de son exécution. En savoir plus

Concurrency: ☐ Run multiple steps at the same time to improve cluster utilization

After last step completes: ◉ Clusters enters waiting state ○ Résilier automatiquement le cluster après la fin de la dernière étape

Type d'étape Sélectionner une étape — Ajouter une étape

Nom	Action sur échec	Emplacement JAR	Arguments
Application Spark	Continuer	command-runner.jar	spark-submit --deploy-mode client s3://p8-prevot/app-opencv-acp.py False

4

Cluster scaling

Adjust the number of Amazon EC2 instances available to an EMR cluster via EMR-managed scaling more

Cluster scaling ☑ Enable Cluster Scaling

Core and task units

Minimum: 2

Maximum: 6

◉ Use EMR-managed scaling ○ Create a custom automatic scaling policy

5

Actions d'amorçage

Les actions d'amorçage sont des scripts exécutés lors de la configuration avant le démarrage de Hadoop sur chaque nœud de cluster. Vous pouvez les utiliser pour installer des logiciels supplémentaires et personnaliser vos applications. En savoir plus

Type d'action d'amorçage	Nom	Emplacement JAR	Arguments facultatifs
Action personnalisée	Custom action	s3://p8-prevot/bootstrap-emr.sh	

Ajouter une action d'amorçage Sélectionner une action d'amorçage — Configurer et ajouter

6

Options de sécurité

Paire de clés EC2 alex-ec2-rsa

☑ Cluster visible pour tous les utilisateurs IAM du compte

Autorisations

◉ Par défaut ○ Personnalisé

Utilisez les rôles IAM par défaut. Si des rôles sont absents, ils seront créés automatiquement pour vous avec des stratégies gérées pour les mises à jour automatiques de stratégies.

Rôle EMR EMR_DefaultRole ☐ Use EMR_DefaultRole_V2

Profil d'instance EC2 EMR_EC2_DefaultRole

Rôle Auto Scaling EMR_AutoScaling_DefaultRole

3. Architecture Cloud choisie

Résultats

```
########## Import des librairies et modules: TerminÃ© ##########
Session Spark initialisÃ©e avec succÃ¨s
########## CrÃ©ation du dataframe contenant les images: TerminÃ© ##########
+--------------------+-----------+
|               image|      label|
+--------------------+-----------+
|[s3://p8-prevot/d...|apple_red_2|
|[s3://p8-prevot/d...|   carrot_1|
|[s3://p8-prevot/d...|   carrot_1|
|[s3://p8-prevot/d...|apple_red_2|
|[s3://p8-prevot/d...|apple_red_2|
+--------------------+-----------+
only showing top 5 rows

########## Transformtion du dataframe, ajout de descripteurs ORB: TerminÃ© ##########
+--------------------+-----------+--------------------+
|               image|      label|    Descripteurs ORB|
+--------------------+-----------+--------------------+
|[s3://p8-prevot/d...|apple_red_2|[[162, 180, 38, 1...|
|[s3://p8-prevot/d...|   carrot_1|[[156, 70, 98, 13...|
|[s3://p8-prevot/d...|apple_red_2|[[150, 237, 198, ...|
|[s3://p8-prevot/d...|   carrot_1|[[92, 90, 139, 62...|
|[s3://p8-prevot/d...|   carrot_1|[[88, 173, 66, 18...|
+--------------------+-----------+--------------------+
only showing top 5 rows

########## Transformtion du dataframe, explosion des descripteurs: TerminÃ© ##########
+--------------------+--------+--------------------+
|               image|   label|         Descripteur|
+--------------------+--------+--------------------+
|[s3://p8-prevot/d...|carrot_1|[156, 70, 98, 137...|
|[s3://p8-prevot/d...|carrot_1|[191, 104, 240, 2...|
|[s3://p8-prevot/d...|carrot_1|[226, 41, 192, 15...|
|[s3://p8-prevot/d...|carrot_1|[200, 165, 82, 22...|
|[s3://p8-prevot/d...|carrot_1|[66, 165, 98, 191...|
+--------------------+--------+--------------------+
only showing top 5 rows
```

```
########## Transformtion du dataframe, Vectorisation des descripteurs: TerminÃ© ##########
+--------------------+--------+--------------------+
|               image|   label|    Descripteur-vect|
+--------------------+--------+--------------------+
|[s3://p8-prevot/d...|carrot_1|[156.0,70.0,98.0,...|
|[s3://p8-prevot/d...|carrot_1|[191.0,104.0,240....|
|[s3://p8-prevot/d...|carrot_1|[226.0,41.0,192.0...|
|[s3://p8-prevot/d...|carrot_1|[200.0,165.0,82.0...|
|[s3://p8-prevot/d...|carrot_1|[66.0,165.0,98.0,...|
+--------------------+--------+--------------------+
only showing top 5 rows

########## EntraÃ®nement de l'ACP sur les descripteurs: TerminÃ© ##########
########## Transformtion du dataframe, rÃ©duction par ACP: TerminÃ© ##########
+--------------------+--------+--------------------+
|               image|   label|     Descripteur-ACP|
+--------------------+--------+--------------------+
|[s3://p8-prevot/d...|carrot_1|[299.752928733064...|
|[s3://p8-prevot/d...|carrot_1|[158.436716708492...|
|[s3://p8-prevot/d...|carrot_1|[98.3969008371953...|
|[s3://p8-prevot/d...|carrot_1|[270.672706369656...|
|[s3://p8-prevot/d...|carrot_1|[375.186627174899...|
+--------------------+--------+--------------------+
only showing top 5 rows

########## Action sur le dataframe, enregistrement au format parquet sur S3: TerminÃ© ##########
```

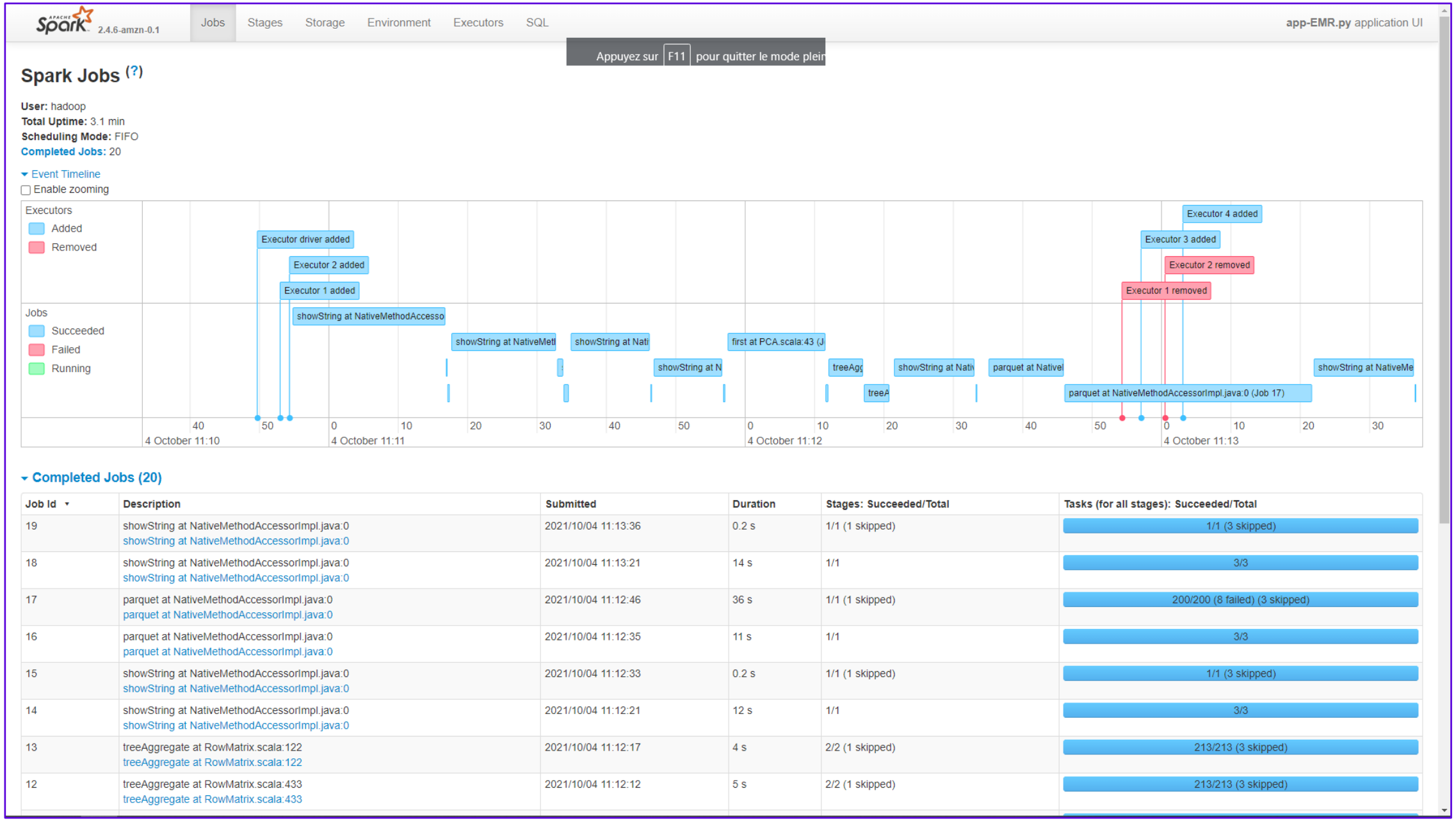
```
df = sqlContext.read.parquet('s3://p8-prevot/results/')
```

```
df.show()
```

```
+--------------------+--------+--------------------+
|               image|   label|     Descripteur-ACP|
+--------------------+--------+--------------------+
|[s3://p8-prevot/d...|carrot_1|[299.752928733064...|
|[s3://p8-prevot/d...|carrot_1|[158.436716708492...|
|[s3://p8-prevot/d...|carrot_1|[98.3969008371953...|
|[s3://p8-prevot/d...|carrot_1|[270.672706369656...|
|[s3://p8-prevot/d...|carrot_1|[375.186627174899...|
+--------------------+--------+--------------------+
```

3. Architecture Cloud choisie

[Monitoring: Spark History Server]



3. Architecture Cloud choisie

Monitoring: YARN Timeline Server

Logged in as: dr.who

Application application_1633345723144_0001

- Application History
  - About
  - Applications
    - FINISHED
    - FAILED
    - KILLED
- Tools

Application Overview	
User:	hadoop
Name:	app-EMR.py
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Mon Oct 04 11:10:42 +0000 2021
Launched:	N/A
Finished:	Mon Oct 04 11:13:36 +0000 2021
Elapsed:	2mins, 54sec
Tracking URL:	History
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	CORE

Show 20 entries Search:

Attempt ID	Started	Node	Logs
appattempt_1633345723144_0001_000001	Mon Oct 4 13:10:44 +0200 2021	http://ip-172-31-29-219.eu-west-3.compute.internal:8042	Logs

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Conclusion et perspectives d'évolution

Conclusions tirées du travail effectué

- Étude **de faisabilité de l'application de classification** de fruits:
  1. Distribution horizontale des calculs sur plusieurs machines
  2. Traitement: Utilisation de Spark
  3. Stockage de données massives sur le cloud S3
  4. Application PySpark sur une instance EC2
  5. Application PySPark sur un cluster EMR
  6. Utilisation d'outils de monitoring

- **Le cluster EMR offre une possibilité de mise à l'échelle** en ajoutant plusieurs machines si besoin.

- Le système de gestions de ressources est **tolérant aux pannes**
- Possibilité de consulter les résultats, les explorer grâce au notebook EMR
- Coût maîtrisé pour une utilisation normale (0.06$/heure par instance)

Axes d'amélioration

- **Terminer le traitement des données**
  1. Entraînement de l'ACP sur toutes les images
  2. Utiliser K-Means pour regrouper les descripteurs en visual words
  3. Entraînement du classifieur
  4. Optimisation de la classification (tester d'autres méthodes SURF, Transfer Learning)

- Déterminer le nombre minimal d'instances requises
- Optimiser le calcul:
  1. Analyser les temps d'exécution des étapes
  2. Identifier les étapes coûteuses
  3. Trouver les alternatives aux tâches coûteuses

- Monitoring: Utiliser d'autres outils (Ganglia)

Merci

J’espère avoir répondu aux attentes du projet n°8, qui clôture cette année de formation riche en apprentissage.

Je serai heureux d’échanger avec vous, et de répondre à vos questions.

alex290698@gmail.com