



16/03/2024

Réalisez un traitement dans un environnement Big Data sur le Cloud

Armand FAUGERE [LinkedIn](#)

armand-faugere@live.fr

Sommaire

- I) Cadrage du projet et données d'entrée
- II) Mise en place de l'environnement big data
- III) Mise en œuvre du traitement
- IV) Conclusion



I) Cadrage du projet et données d'entrée



I) Cadrage du projet et données d'entrée



❑ Contexte :

- solutions innovantes pour la récolte de fruits
- application mobile avec prise de photo et identification du fruit
- ➔ Sensibiliser à la diversité des fruits
- ➔ Première version du moteur de classification des images de fruits
- ➔ Travaux de traitement initial réalisés et à utiliser



❑ But :

- Réaliser une démonstration de la mise en place d'une instance EMR opérationnelle

❑ Objectifs :

- S'appropriier le travail déjà réalisé
- Respecter le principe **RGPD**
- Mettre en œuvre une solution de traitement en local
- Préparer l'environnement de traitement big data
- Mettre en œuvre la solution de traitement via une instance EMR
- Réaliser une conclusion

➔ Un notebook réalisé sous linux (Ubuntu)

➔ <https://www.kaggle.com/datasets/moltean/fruits>

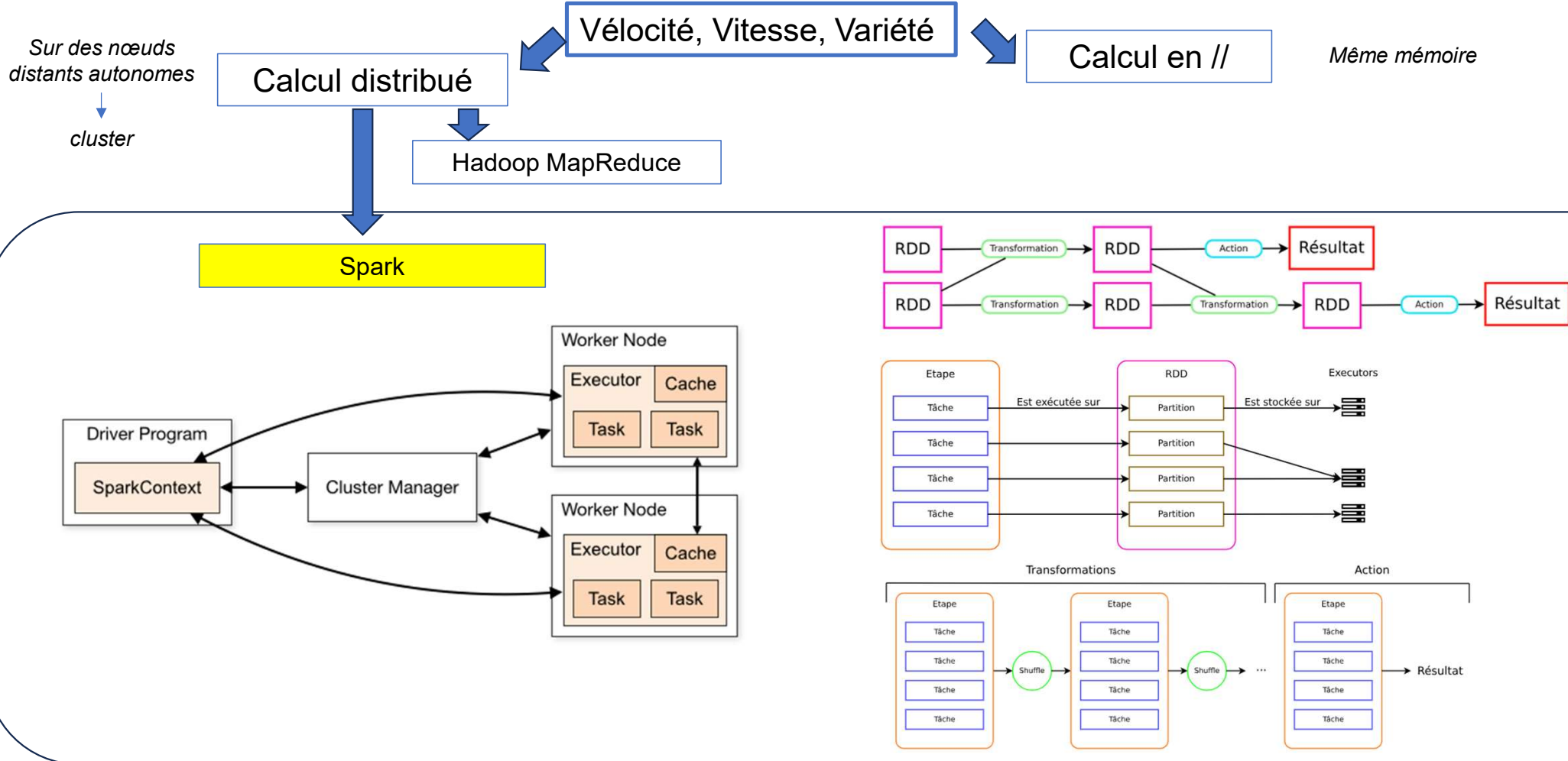
- The total number of images: 90483.
- Training set size: 67692 images (one fruit or vegetable per image).
- Test set size: 22688 images (one fruit or vegetable per image).
- The number of classes: 131 (fruits and vegetables).
- Image size: 100x100 pixels

II) Mise en place de l'environnement big data



II) Mise en place de l'environnement big data

Le Big data



II) Mise en place de l'environnement big data

Ajustement notebook et Test en local



Google colab, pyspark, pyarrow, io, os, tensorflow, Jupiter Notebook, Python, pandas, numpy,



Mise en place
google colab / drive



Ajustement du
notebook



Test en local



- ☐ création d'un dossier google drive
- ☐ Mise en place des liens dans le notebook
- ☐ exécution du notebook



- ☐ Mise à jour
- ☐ Vérification des fonctions
- ☐ Ajout d'une réduction de dimension (pca)



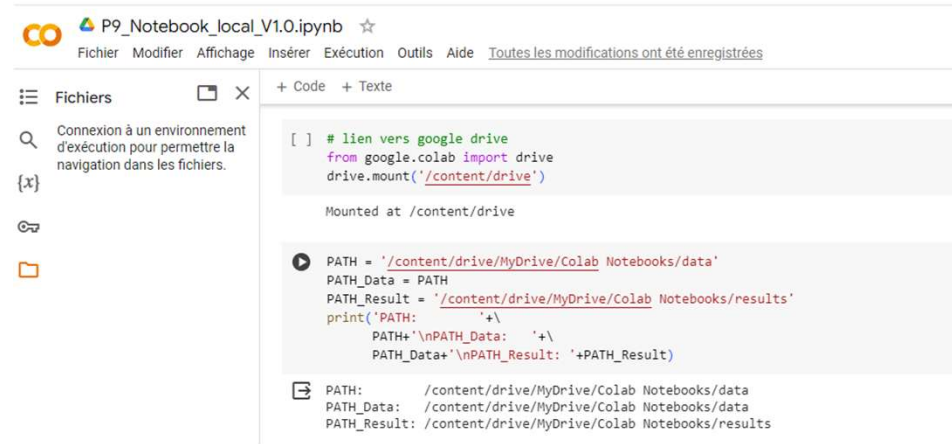
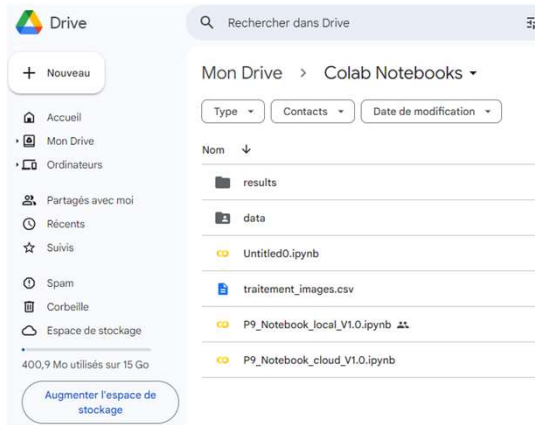
- ☐ Test du notebook avec enregistrement des images dans un répertoire results

1

Ajustement
notebook et
test en local

II) Mise en place de l'environnement big data

Ajustement notebook et Test en local

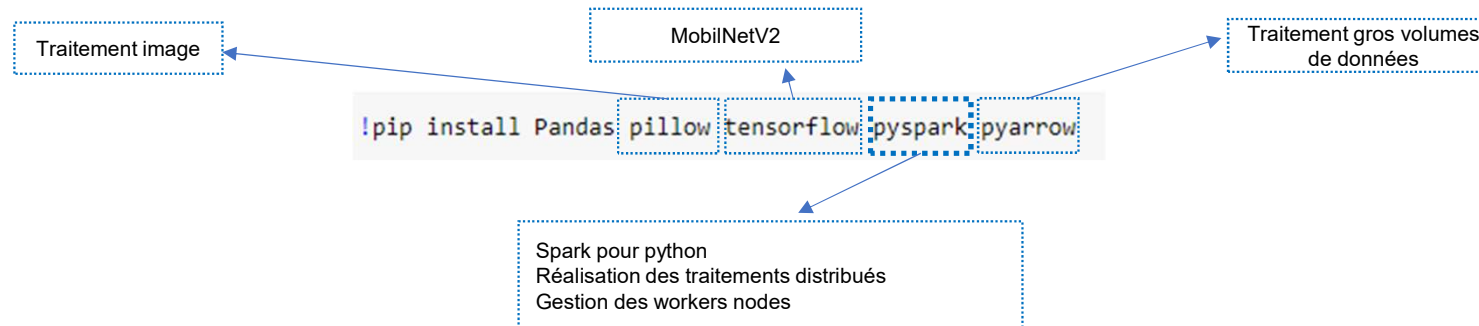


Espace documentaire google drive

Mise en place des liens

II) Mise en place de l'environnement big data

Ajustement notebook et Test en local



```
import pandas as pd
from PIL import Image
import numpy as np
import io
import os

import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras import Model
from pyspark.sql.functions import col, pandas_udf, PandasUDFType, element_at, split
from pyspark.sql import SparkSession
```

```
from pyspark.ml.feature import PCA
from pyspark.ml.functions import array_to_vector
from pyspark.ml.functions import vector_to_array
```

II) Mise en place de l'environnement big data

Ajustement notebook et Test en local



```
spark = (SparkSession
    .builder
    .appName('P9')
    .master('local')
    .config("spark.sql.parquet.writeLegacyFormat", 'true')
    .getOrCreate()
)
```

SparkSession (driver process)

```
sc = spark.sparkContext
```

Objet pour gérer les propriétés globales de l'application

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

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

Chargement des images

II) Mise en place de l'environnement big data

Ajustement notebook et Test en local



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

Instanciation du modèle

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

Transfert learning, on récupère l'avant dernière couche

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	-	-

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

```
def model_fn():
    """
    Returns a MobileNetV2 model with top layer removed
    and broadcasted weights.
    """
    model = MobileNetV2(weights='imagenet',
                        include_top=True,
                        input_shape=(224, 224, 3))
    for layer in model.layers:
        layer.trainable = False
    new_model = Model(inputs=model.input,
                      outputs=model.layers[-2].output)
    new_model.set_weights(broadcast_weights.value)
    return new_model
```

Chargement du modèle sur le driver et diffusion des poids aux workers

II) Mise en place de l'environnement big data

Ajustement notebook et Test en local



Preprocessing + Extraction 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)
```

Exécution des fonctions

```
features_df = images.repartition(20).select(col("path"),
                                             col("label"),
                                             featurize_udf("content").alias("features"),
                                             featurize_udf("content").alias("features_vectors")).withColumn("features_vectors", array_to_vector("features_vectors"))

features_df.show(5, True)
```

path	label	features	features_vectors
file:/content/dri...	Carambula	[0.0017966835, 0.0...	[0.00179668352939...
file:/content/dri...	Carambula	[0.06304055, 0.0...	[0.06304054707288...
file:/content/dri...	Avocado	[0.45384294, 0.0...	[0.45384293794631...
file:/content/dri...	Cactus fruit	[0.54921436, 0.08...	[0.54921436309814...
file:/content/dri...	Grape Blue	[0.0, 0.0, 0.0, 0...	[0.0,0.0,0.0,0.0,...]

only showing top 5 rows


```
df_pca.write.mode("overwrite").parquet(PATH_Result)
```



```
df = pd.read_parquet(PATH_Result, engine='pyarrow')
```

df.head()	path	label	features	features_vectors	pca_vectors	pca_features
0	file:/content/dmViv/Drive/Cviba/Notebooks/da	Carambola	[0.0017966835, 0.0, 0.0, 0.0, 1.365782, 0.0, 0.0]	(['type', '1', 'size', 'none', 'indices', 'none', 'va', ''], ['type', '1', 'size', 'none', 'indices', 'none', 'va', ''])	[7.028649156740277, -3.18289134575842, 5.8662]	
1	file:/content/dmViv/Drive/Cviba/Notebooks/da	Carambola	[0.06304055, 0.0, 0.0, 0.0, 1.1865275, 0.0, 1.]	(['type', '1', 'size', 'none', 'indices', 'none', 'va', ''], ['type', '1', 'size', 'none', 'indices', 'none', 'va', ''])	[8.867739418873435, 3.095568660514266, -5.75]	
2	file:/content/dmViv/Drive/Cviba/Notebooks/da	Avocado	[0.4382594, 0.0, 0.0, 0.0, 1.15254884, 0.0, 0.0]	(['type', '1', 'size', 'none', 'indices', 'none', 'va', ''], ['type', '1', 'size', 'none', 'indices', 'none', 'va', ''])	[8.10792707605341, -7.706970763694773, 3.256]	
3	file:/content/dmViv/Drive/Cviba/Notebooks/da	Cacao Fruit	[0.54921436, 0.06829143, 0.0, 0.0, 0.0606524, 0.0, 0.0]	(['type', '1', 'size', 'none', 'indices', 'none', 'va', ''], ['type', '1', 'size', 'none', 'indices', 'none', 'va', ''])	[4.81737956142003, 4.801124221633475, 0.3507]	
4	file:/content/dmViv/Drive/Cviba/Notebooks/da	Grape Seed	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.374551]	(['type', '1', 'size', 'none', 'indices', 'none', 'va', ''], ['type', '1', 'size', 'none', 'indices', 'none', 'va', ''])	[1.5470275048035912, -1.6488618916190248, 0.0]	

II) Mise en place de l'environnement big data

Environnement de traitement



AWS, S3, ECS, EMR, AWS CLI, pyspark, pyarrow, io, os, tensorflow, JupiterHub, Python, Pandas, numpy



2

Mise en place
de
l'environnement
bigdata



S3



EC2



EMR



IAM

Ajustement du
notebook pour
utilisation AWS



- ☐ bootstrap +
instanciation
SPARK
- ☐ Liens vers S3

Chargement
des données
sur S3



- ☐ Images
- ☐ script pour
bootstraping

Création et
paramétrage
cluster sur EMR



- ☐ Libraires et autres
packages (EMR,
Hadoop...)
- ☐ Lieu du serveur
- ☐ Paires de clefs
EC2
- ☐ Bootstrap actions
- ☐ Instances EC2 :
nb de machines à
utiliser (1 driver 4
cores) et € associés

Cluster prêt
→ waiting

II) Mise en place de l'environnement big data

Environnement de traitement



Extrait de l'écran S3

AWS CLI

Amazon S3

Buckets

- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

Amazon S3 > Buckets > fruitsaf33

fruitsaf33

Info

Objects Properties Permissions Metrics Management Acc

Objects (4) Info

Refresh Copy S3 URI Copy URL Download Open

Delete Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	bootstrap-emr.sh	sh	March 4, 2024, 08:48:05 (UTC+01:00)	337.0 B	Standard
<input type="checkbox"/>	data/	Folder	-	-	-

```
(projet_9_OC) PS C:\Users\ARMAN> aws s3 ls
Association de fichier introuvable pour l'extension .py
2024-03-01 18:04:32 aws-logs-905418132460-eu-west-1
2024-02-29 15:11:44 fruitsaf33
2024-03-01 16:48:09 p8-data-af
(projet_9_OC) PS C:\Users\ARMAN> |
```

II) Mise en place de l'environnement big data

Environnement de traitement



❑ Extrait de l'écran EMR création de cluster

Amazon EMR > EMR on EC2: Clusters > Create cluster

Clone "clusteraf" Info

▼ **Name and applications - required** Info
Name your cluster and choose the applications that you want to install to your cluster.

Name

Amazon EMR release Info
A release contains a set of applications which can be installed on your cluster.

Application bundle

Spark

Core Hadoop

HBase

Presto

Trino

Custom

☐ Flink 1.14.2
☐ HCatalog 3.1.3
☐ Hue 4.10.0
☐ Livy 0.7.1
☐ Phoenix 5.1.2
☒ Spark 3.2.1
☐ Tez 0.9.2
☐ ZooKeeper 3.5.7

☐ Ganglia 3.7.2
☐ Hadoop 3.2.1
☐ JupyterEnterpriseGateway 2.1.0
☐ MXNet 1.8.0
☐ Pig 0.17.0
☐ Sqoop 1.4.7
☐ Trino 378

☐ HBase 2.4.4
☐ Hive 3.1.3
☒ JupyterHub 1.4.1
☐ Oozie 5.2.1
☐ Presto 0.272
☒ TensorFlow 2.4.1
☒ Zeppelin 0.10.0

AWS Glue Data Catalogue settings
Use the AWS Glue Data Catalog to provide an external metastore for your application.
☐ Use for Spark table metadata

Operating system options Info
☒ Amazon Linux release
☐ Customised Amazon Machine Image (AMI)

Summary Info

Name and applications - required

Name
clusteraf

Amazon EMR release
emr-6.7.0

Application bundle
Custom (Hadoop 3.2.1, JupyterHub 1.4.1, Spark 3.2.1, TensorFlow 2.4.1, Zeppelin 0.10.0)

Cluster configuration - required

Instance groups
Primary (m5.xlarge), Core (m5.xlarge)

Cluster scaling and provisioning - required

Provisioning configuration
Core size: 4 Instances

❑ Extrait de l'écran EMR de l'état des clusters

Amazon EMR > EMR on EC2: Clusters

EMR Serverless

▼ EMR on EC2

- Clusters
- Notebooks and Git repos
- Events
- Block public access
- Security configurations

▼ EMR on EKS

Clusters (11) Info

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Cluster ID	Cluster name	Status	Creation time (UTC+01:00)	Elapsed time	Normalised instance hours
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	j-3HA757JUM2BDQ	clusteraf	Terminated User request	4 March 2024 09:03	32 minutes, 3 seconds	40
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	j-2OFR8GXX8AFFK	clusteraf	Terminated User request	4 March 2024 08:48	16 minutes, 29 seconds	40
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	j-2OLY63OV3R0X3	clusteraf	Terminated User request	4 March 2024 08:19	27 minutes, 41 seconds	40

II) Mise en place de l'environnement big data

Environnement pour traitement



❑ cluster utilisé pour traitement

aws Services Search [Alt+S]

Amazon EMR > EMR on EC2: Clusters > clusteraf

Updated 1 minute ago Terminate Clone in AWS CLI Clone

clusteraf

▼ Summary

Cluster info Cluster ID j-3HA7S7IUM2BDQ Cluster configuration Instance groups Capacity 1 Primary 4 Core 0 Task	Applications Amazon EMR version emr-6.7.0 Installed applications Hadoop 3.2.1, JupyterHub 1.4.1, Spark 3.2.1, TensorFlow 2.4.1, Zeppelin 0.10.0	Cluster management Log destination in Amazon S3 aws-logs-905418132460-eu-west-1/elasticmapreduce Persistent application UIs Spark history server YARN timeline server Primary node: public DNS ec2-34-243-60-174.eu-west-1.compute.amazonaws.com Connect to the Primary node using SSH	Status and time Status Terminated Creation time 4 March 2024 09:03 (UTC+01:00) Elapsed time 32 minutes, 3 seconds End time 4 March 2024 09:35 (UTC+01:00)
---	--	---	--

Properties Bootstrap actions Instances (hardware) Steps Applications Configurations Monitoring Events Tags (1)

Operating system Info Amazon Linux release 2.0.20240223.0	Cluster logs Info Archive log files to Amazon S3 Turned on Amazon S3 location s3://aws-logs-905418132460-eu-west-1/elasticmapreduce/ Turn on encryption for logs Turned off	Cluster termination Info Termination option Automatically terminate the cluster after idle time Termination protection Off Idle time 2 hours
--	--	---

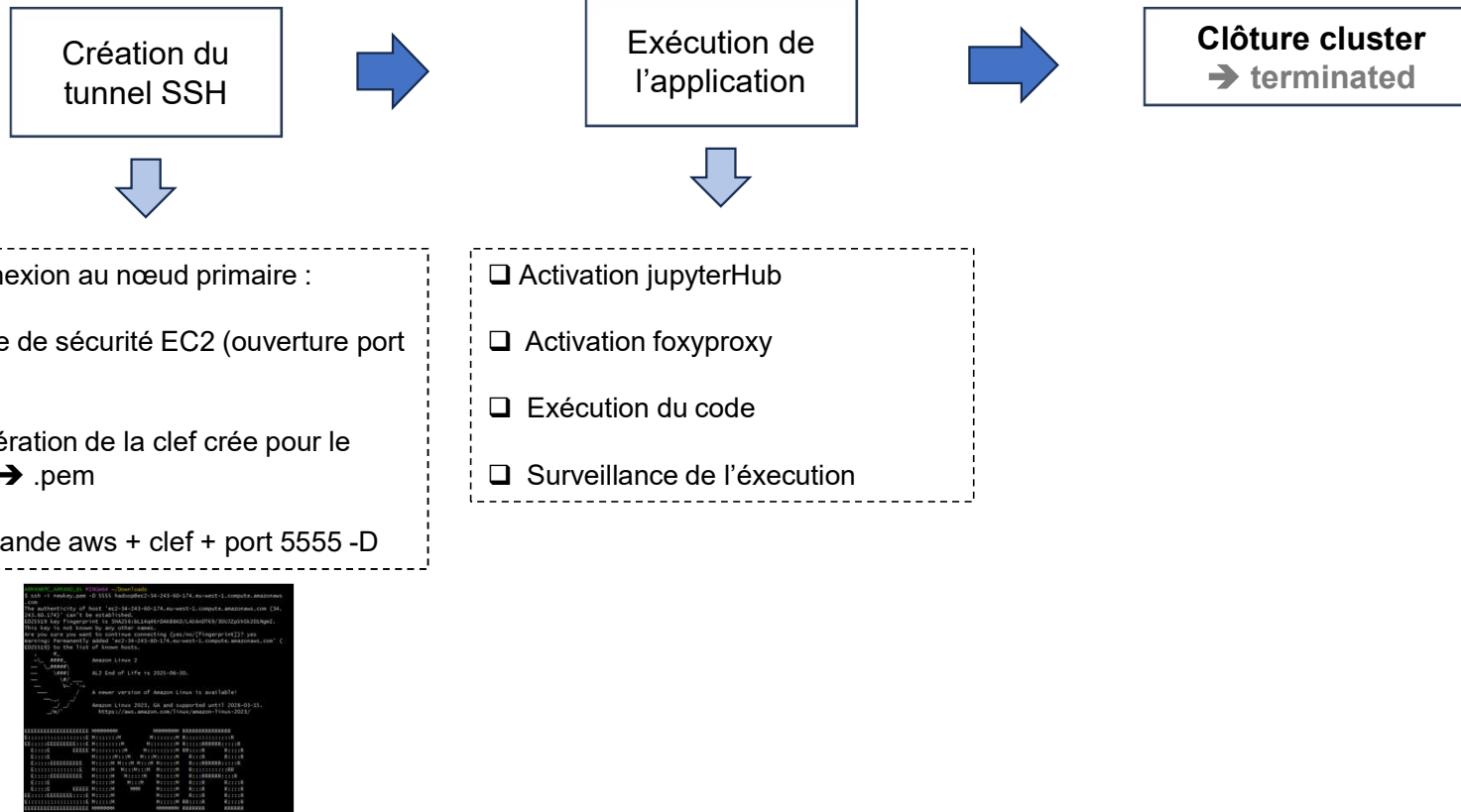
Network and security Info

Network Virtual Private Cloud (VPC) vpc-0a6e9439c9bf6833b Subnet ID and Availability Zone (AZ) subnet-08bfeddfc9b9c55fe eu-west-1c EC2 security groups (firewall)	Security configuration Security configuration none EC2 key pair newkey	Permissions Service role for Amazon EMR AmazonEMR-ServiceRole-20240302T195511 EC2 instance profile AmazonEMR-InstanceProfile-20240302T195454 Auto-scaling role Not configured
---	---	--

III) Mise en œuvre du traitement



AWS, S3, ECS, EMR, AWS CLI, Foxyproxy, gitbash, pyspark, pyarrow, io, os, tensorflow, JupiterHub, Python, Pandas, numpy



III) Mise en œuvre du traitement



```
ARMANAPC ARMAND_85 MINGW64 ~/Downloads
$ ssh -i newkey.pem -D 5555 hadoop@ec2-34-243-60-174.eu-west-1.compute.amazonaws.com
The authenticity of host 'ec2-34-243-60-174.eu-west-1.compute.amazonaws.com (34.243.60.174)' can't be established.
ED25519 key fingerprint is SHA256:bL14q4trDAKB8KD/LA56nDTK9/3OUJZpShSk2D1NgmI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-34-243-60-174.eu-west-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
```



AL2 End of Life is 2025-06-30.

A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
<https://aws.amazon.com/linux/amazon-linux-2023/>

```
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M M::::::::M R::::::::::::R
EE::::::::::::::::::::E M::::::::M M::::::::M R::::::::::::R
E:::::E EEEEE M::::::::M M::::::::M RR:::R R:::R
E:::::E M::::::::M M::::::::M R:::R R:::R
E:::::EEEEEEEEEE M:::::M M:::M M:::M M:::M R:::RRRRRR:::R
E::::::::::::::::::::E M:::::M M:::M M:::M M:::M R::::::::::::RR
E:::::EEEEEEEEEE M:::::M M:::M M:::M M:::M R::::::::::::RR
E:::::E M:::::M M:::M M:::M M:::M R:::R R:::R
E:::::E EEEEE M:::::M MMM M:::::M R:::R R:::R
EE::::::::::::::::::::E M:::::M M:::::M R:::R R:::R
E::::::::::::::::::::E M:::::M M:::::M RR:::R R:::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRR RRRRRR
```

Amazon EMR > EMR on EC2: Clusters > clusteraf

clusteraf

▼ Summary

Cluster info Cluster ID j-3HA757IUM2BDQ Cluster configuration Instance groups Capacity 1 Primary 4 Core 0 Task	Applications Amazon EMR version emr-6.7.0 Installed applications Hadoop 3.2.1, JupyterHub 1.4.1, Spark 3.2.1, TensorFlow 2.4.1, Zeppelin 0.10.0	Cluster management Log destination in Amazon S3 aws-logs-905418132460-eu-west-1/elasticmapreduce Persistent application UIs Spark history server 🔗 YARN Timeline server 🔗 Primary node public DNS ec2-34-243-60-174.eu-west-1.compute.amazonaws.com Connect to the Primary node using SSH
---	--	--

Création du tunnel SSH

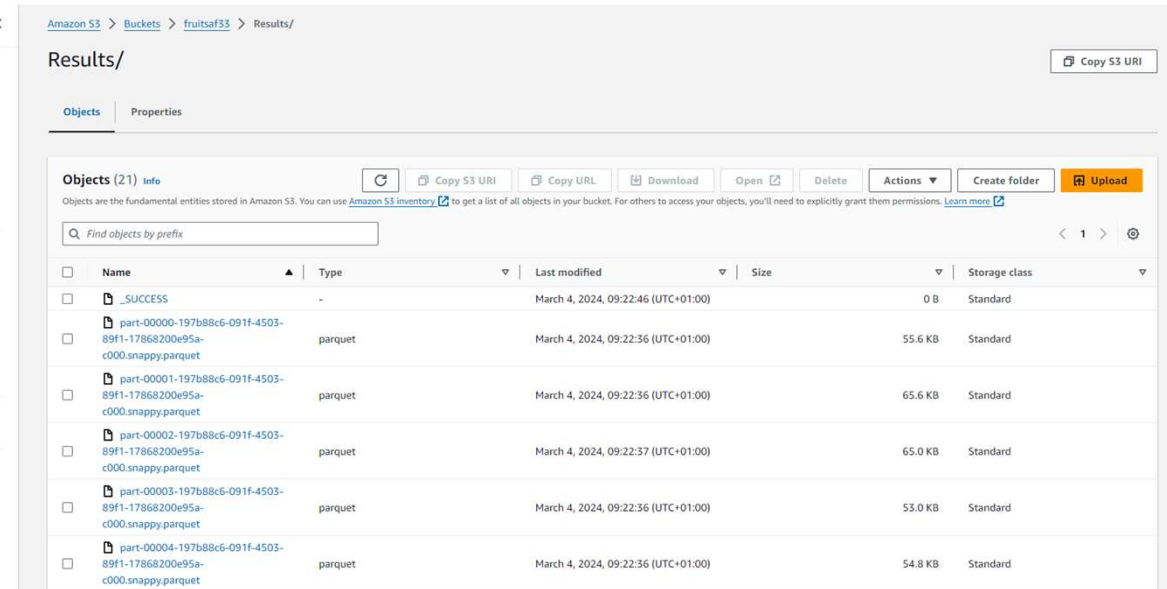
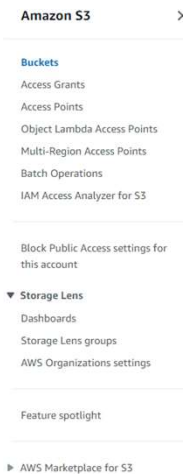
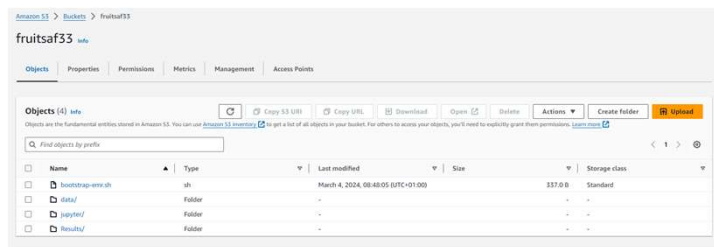
Suivi de l'historique

2024 Armand FAUGERE

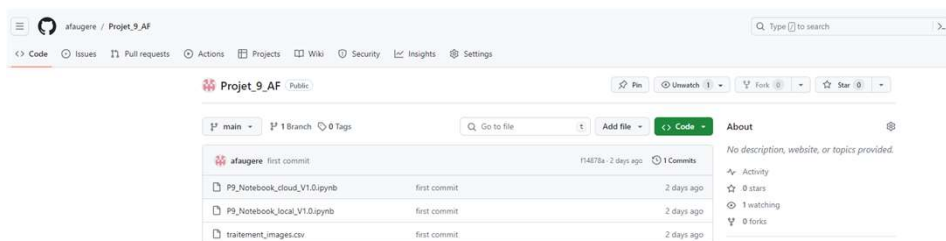
III) Mise en œuvre du traitement



Répertoire S3



Repository github



III) Mise en œuvre du traitement



Facturation AWS

The screenshot displays the AWS Billing and Cost Management console. The left sidebar shows navigation options: Billing and Cost Management, Home, Getting Started, Billing and Payments, Billing and Payments, Credits, Purchase Orders, Cost Analysis, Cost Explorer, Cost Explorer Saved Reports, Cost Anomaly Detection, Free Tier, Data Exports, Cost Organization, Cost Categories, Cost Allocation Tags, Billing Conductor, Budgets and Planning, Budgets, Budgets Reports, Pricing Calculator, Savings and Commitments.

The main content area shows the 'Bills' page for the 'Billing period: March 2024'. The page includes a 'Download all to CSV' button, a 'Print' button, and a 'Billing period: March 2024' dropdown. The 'AWS estimated bill summary' section shows the following details:

Account ID	Billing period	Bill status
905418132460	March 1 - March 31, 2024	Pending

The 'Service provider' is Amazon Web Services EMEA SARL, and the 'Total in USD' is USD 7.41. The 'Estimated grand total' is USD 7.41.

The 'Payment information' section is expanded, showing the 'Highest estimated cost by service provider' for Amazon Web Services EMEA SARL. The table shows the following details:

Service name	Highest service spend	Trend compared to prior month	Region name	Highest AWS Region spend	Trend compared to prior month
Elastic Compute Cloud	USD 5.27	No data to display.	EU (Ireland)	USD 6.12	No data to display.

III) Mise en œuvre du traitement



Démonstration du script en ligne AWS

IV) Conclusion



IV) Conclusion



☐ Rappel Contexte :

- solutions innovantes pour la récolte de fruits
- **application mobile avec prise de photo et identification du fruit**

- Sensibiliser à la diversité des fruits
- Première version du moteur de classification des images de fruits
- Travaux de traitement initial réalisés et à utiliser

Forces

- ☐ Fonctionne sur le cloud pour notre projet
- ☐ Flexibilité (adaptation de la puissance de calcul et autres ressources)
- ☐ Suivi complet et instantané (opérations de traitement, facturation...)
- ☐ Ergonomie de la solution AWS
- ☐ Sécurité

Faiblesses

- ☐ Facturation à la seconde, les € peuvent augmenter rapidement

Risques








- ☐ Perte financière si mauvaise gestion de la solution

Opportunités

- ☐ Piloter le déploiement de la solution en mode progressif et en adaptant les ressources
- ☐ Suivre les dépenses réalisés vs « chiffre d'affaires réalisé »
- ☐ Mettre en place une gestion pilotée de la solution

IV) Conclusion



Réaliser une démonstration de la mise en place d'une instance EMR opérationnelle		
S'approprier le travail déjà réalisé	Le notebook existant a été réutilisé et adapté à la problématique	
Respecter le principe RGPD	Utilisation du serveur européen et de clefs de sécurité	
Mettre en œuvre une solution de traitement en local	Mise au point de la solution de traitement en local avec googlecolab	
Préparer l'environnement de traitement big data	Environnement de traitement mis au point avec AWS (cluster configuré), et froxyproxy	
Mettre en œuvre la solution de traitement via une instance EMR	Mise en oeuvre de la solution de traitement avec enregistrement des résultats et historique de traitement	
Réaliser une conclusion	Une synthèse a été réalisée	

Merci

- Armand FAUGERE
- armand-faugere@live.fr

