

Projet 8 : Déployez un modèle dans le cloud

Oumeima EL GHARBI

OpenClassrooms – Data Scientist

Soutenance : 03/03/2023

Plan

Introduction

- Problématique
- Jeu de données

I. Création d'un environnement Big Data

- Stockage : S3
- Calcul distribué : EMR

II. Chaîne traitement des images

- Transfert Learning
- ACP

III. Démo AWS

Conclusion

Introduction

Problématique :

«Vous êtes Data Scientist dans une très jeune start-up de l'AgriTech, nommée "Fruits!", qui cherche à proposer des solutions innovantes pour la récolte des fruits. [...]

Votre 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.»

Cadre :

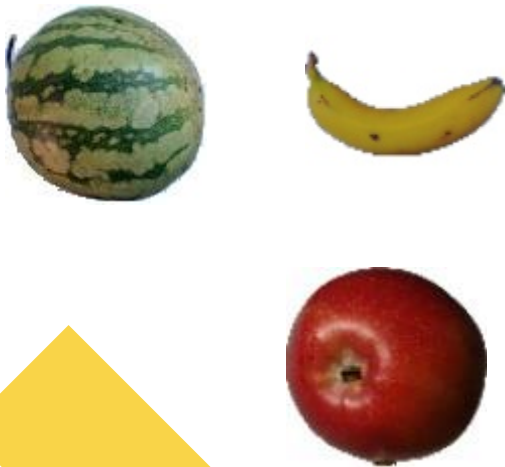
Data science : Transfert Learning

MLOps : solutions PaaS d'AWS

- Stockage : S3 (Simple Storage Service)
- Calcul distribué : EMR (Elastic Map Reduce)

Jeu de données

- Test1 : 34 images pour tester le notebook sur Google Colab
- Test : 22668 images stockées sur S3
- Bucket S3 : oc-p8-data



Mon Drive > AI > OC > P8 ▾

Nom ↑

data

P8_Notebook_Colab.ipynb ★

Amazon S3 > Buckets > oc-p8-data > data/

data/

Objects Properties

Objects (4)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 to manage them permissions. [Learn more](#)

Refresh Copy S3 URI Copy URL Download

Find objects by prefix

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	Results/	Folder
<input type="checkbox"/>	Results1/	Folder
<input type="checkbox"/>	Test/	Folder
<input type="checkbox"/>	Test1/	Folder

Test/

Objects Properties

Objects (131)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 to manage them permissions. [Learn more](#)

Refresh Copy S3 URI Copy URL Download

Find objects by prefix

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	Apple Braeburn/	Folder
<input type="checkbox"/>	Apple Crimson Snow/	Folder
<input type="checkbox"/>	Apple Golden 1/	Folder
<input type="checkbox"/>	Apple Golden 2/	Folder
<input type="checkbox"/>	Apple Golden 3/	Folder
<input type="checkbox"/>	Apple Granny Smith/	Folder
<input type="checkbox"/>	Apple Pink Lady/	Folder
<input type="checkbox"/>	Apple Red 1/	Folder

Apple Red 1/

Objects Properties

Objects (164)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 to manage them permissions. [Learn more](#)

Refresh Copy S3 URI Copy URL Download

Find objects by prefix

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	3_100.jpg	jpg
<input type="checkbox"/>	32_100.jpg	jpg
<input type="checkbox"/>	321_100.jpg	jpg
<input type="checkbox"/>	322_100.jpg	jpg
<input type="checkbox"/>	323_100.jpg	jpg
<input type="checkbox"/>	324_100.jpg	jpg
<input type="checkbox"/>	325_100.jpg	jpg

I) Création d'un environnement Big Data

- 
- 
- Stockage : S3
 - Calcul distribué : EMR

I) Création d'un environnement Big Data

1) Stockage : S3

- Choix de la solution S3

Solution	Proximité des serveurs	Persistance	Faible coût	Lecture aléatoire
<u>EC2</u>	✓	✗	<u>✗</u>	✓
<u>EFS</u>	✓	✓	<u>✗</u>	✓
<u>S3</u>	✓	✓	<u>✓</u>	✗
<u>Glacier</u>	✗	✓	<u>✓</u>	✗

I) Création d'un environnement Big Data

1) Stockage : S3

Amazon S3 > Buckets > oc-p8-data

oc-p8-data [Info](#)

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (3)

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](#)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<div><div></div>bootstrap-emr.sh</div>	sh	March 1, 2023, 16:36:55 (UTC+01:00)	325.0 B	Standard
<input type="checkbox"/>	<div><div></div>data/</div>	Folder	-	-	-
<input type="checkbox"/>	<div><div></div>jupyter-s3-conf.json</div>	json	March 1, 2023, 20:16:52 (UTC+01:00)	170.0 B	Standard

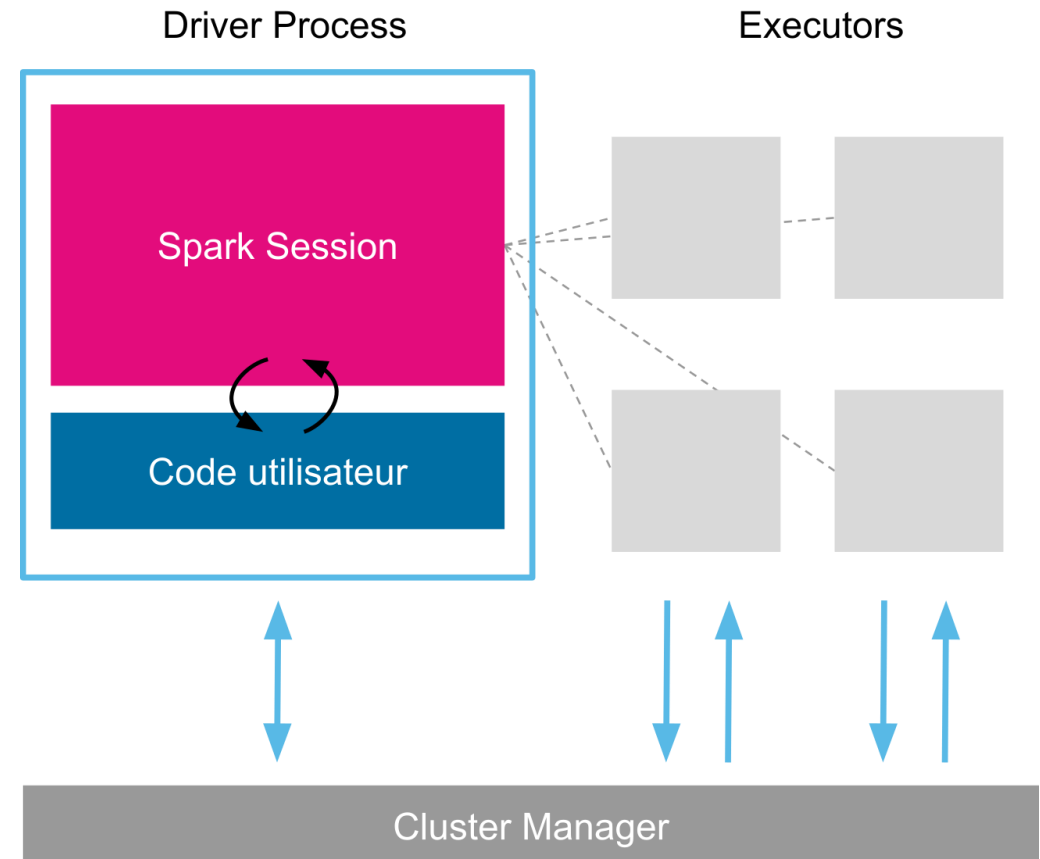
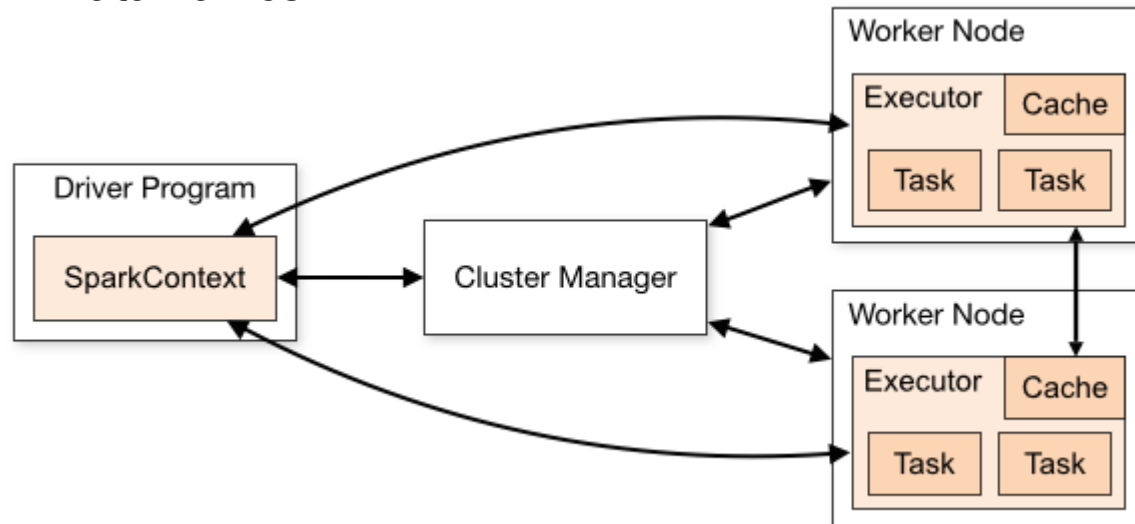
7

1) Création d'un environnement Big Data

2) Calcul distribué : EMR

Application Spark

- Basé sur le paradigme MapReduce
- Architecture Maître / Esclave ; cluster Manager (yum sur EMR)
- RDD (Resilient Distributed Datasets) : Datasets distribués résilients => tolérance aux pannes e distribution des calculs
- DataFrames

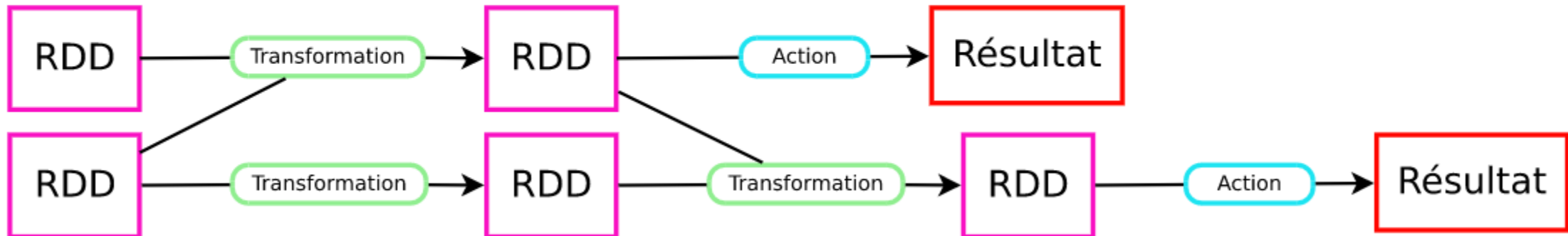


I) Création d'un environnement Big Data

2) Calcul distribué : EMR

Application Spark

- Transformations et les Actions réalisées sur les RDD
- DAG (Directed Acyclic Graph) : Graphe Acyclique Orienté



I) Création d'un environnement Big Data

2) Calcul distribué : EMR

Création d'un cluster EMR

- Création dure 10 à 15 minutes
- Cluster : P8_Fruits
- Clé pour l'instance EC2
- Configuration json pour la persistance de S3 dans le cluster EMR
- Bootstrapping : installation des packages python
- emr-6.3.0
- JupyterHub 1.2.0
- Tensorflow 2.4.1
- Spark 3.1.0

Cluster: P8_Fruits Terminated Terminated by user request

Summary

Application user interfaces

Monitoring

Hardware

Configurations

Events

Steps

Bootstrap actions

Summary

ID: j-RIHGMD6VI6SP

Creation date: 2023-03-01 20:51 (UTC+1)

End date: 2023-03-01 21:51 (UTC+1)

Elapsed time: 59 minutes

After last step completes: Cluster waits

Termination protection: Off

Tags: --

Master public DNS:

ec2-34-254-96-47.eu-west-1.compute.amazonaws.com

[Connect to the Master Node Using SSH](#)

Application user interfaces

Persistent user interfaces [🔗](#): [Spark history server](#), [YARN timeline server](#)

On-cluster user --
interfaces [🔗](#):

Configuration details

Release label: emr-6.3.0

Hadoop distribution: Amazon

Applications: JupyterHub 1.2.0, TensorFlow 2.4.1, Spark 3.1.1

Log URI: [s3://aws-logs-815565965465-eu-west-1/elasticmapreduce/](#) [🔗](#)

EMRFS consistent view: Disabled

Custom AMI ID: --

Network and hardware

Availability zone: eu-west-1b

Subnet ID: [subnet-04546476ea2ee5783](#) [🔗](#)

Master: Terminated 1 m5.xlarge

Core: Terminated 2 m5.xlarge

Task: --

Cluster scaling: Not enabled

1) Création d'un environnement Big Data

2) Calcul distribué : EMR

- Ajout de SSH en IPv4 et IPv6 dans la groupe de Sécurité
- Clé SSH et Tunnel SSH

SSH


Connect to the Master Node Using SSH

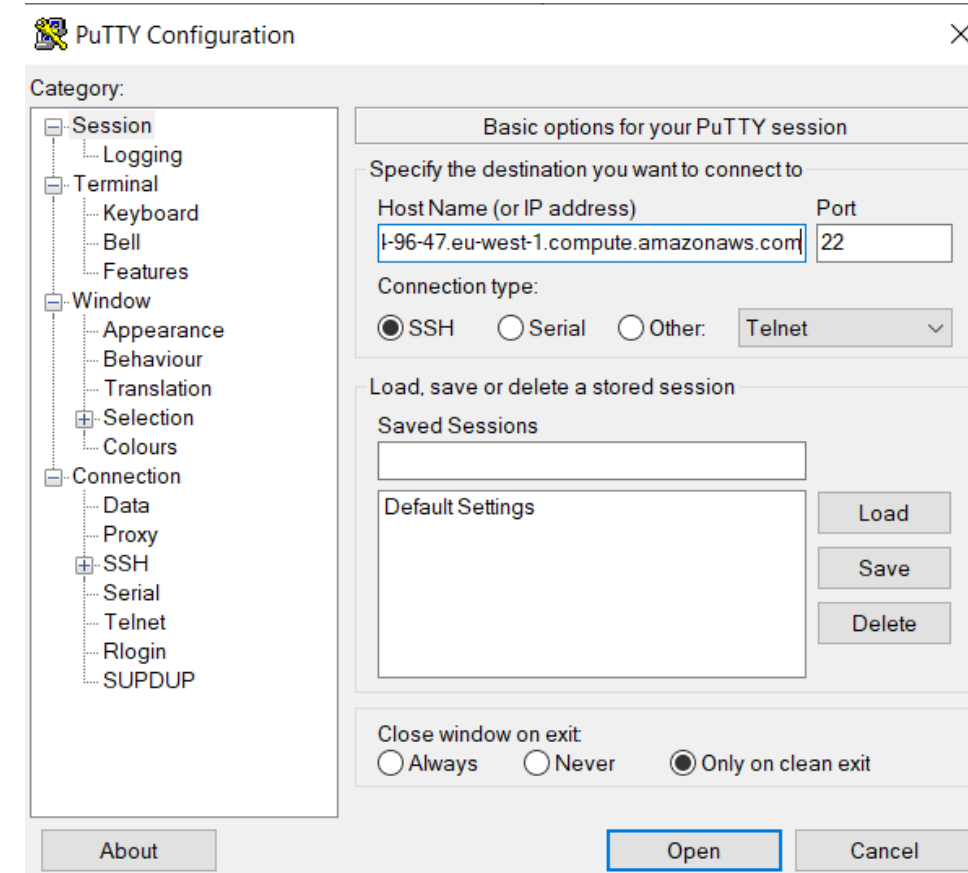
You can connect to the Amazon EMR master node using SSH to run interactive queries, examine log files, submit Linux commands, and so on.

[Learn more](#) 

Windows

Mac / Linux

1. Download PuTTY.exe to your computer from:
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html> 
2. Start PuTTY.
3. In the Category list, click Session.
4. In the Host Name field, type `hadoop@ec2-34-254-96-47.eu-west-1.compute.amazonaws.com`
5. In the Category list, expand Connection > SSH, and then click Auth.
6. For Private key file for authentication, click Browse and select the private key file (`oumeima-oc-p8-ec2.ppk`) used to launch the cluster.
7. Click Open.
8. Click Yes to dismiss the security alert.



1) Création d'un environnement Big Data

2) Calcul distribué : EMR

- Connection SSH réussie

```
hadoop@ip-172-31-11-70:~  
Using username "hadoop".  
Authenticating with public key "oumeima-oc-p8-ec2"  
Last login: Wed Mar  1 18:09:52 2023  
  
  _ | _ | _ )  
  _ | ( _ /  Amazon Linux 2 AMI  
  _ | \ _ | _ |  
  
https://aws.amazon.com/amazon-linux-2/  
82 package(s) needed for security, out of 130 available  
Run "sudo yum update" to apply all updates.  
  
EEEEEEEEEEEEEEEEEEEE MMMMMMMM          MMMMMMMM RRRRRRRRRRRRRRRR  
E::::::::::::::::::::E M::::::::M          M::::::::M R:::::::::R  
EE::::::::EEEEEEEE::::E M::::::::M          M::::::::M R::::RRRRRR::::R  
  E::::E          EEEEE M::::::::M          M::::::::M RR::::R      R::::R  
k  E::::E          M:::::M:::M  M:::M:::::M  R:::R      R::::R  
  E::::EEEEEEEEEE  M:::::M M:::M M:::M M:::::M  R::RRRRRR::::R  
D  E::::::::::::::::E M:::::M  M:::M:::M  M:::::M  R:::::::::RR  
  E::::EEEEEEEEEE  M:::::M  M:::::M  M:::::M  R::RRRRRR::::R  
O  E::::E          M:::::M  M:::M  M:::::M  R:::R      R::::R  
  E::::E          EEEEE M:::::M  MMM  M:::::M  R:::R      R::::R  
b EE::::::::EEEEEEEE::::E M:::::M          M:::::M  R:::R      R::::R  
  E::::::::::::::::::::E M:::::M          M:::::M RR::::R      R::::R  
  EEEEEEEEEEEEEEEEE MMMMMMMM          MMMMMMMM RRRRRRR      RRRRRR  
  
[hadoop@ip-172-31-11-70 ~]$
```

1) Création d'un environnement Big Data

2) Calcul distribué : EMR

- Proxy
- Navigateur : Firefox
- Extension : FoxyProxy



Edit Proxy EMR

Title or Description (optional)

EMR

Color

#66cc66

Send DNS through SOCKS5 proxy

On ☒

Proxy Type

SOCKS5

Proxy IP address or DNS name ★

localhost

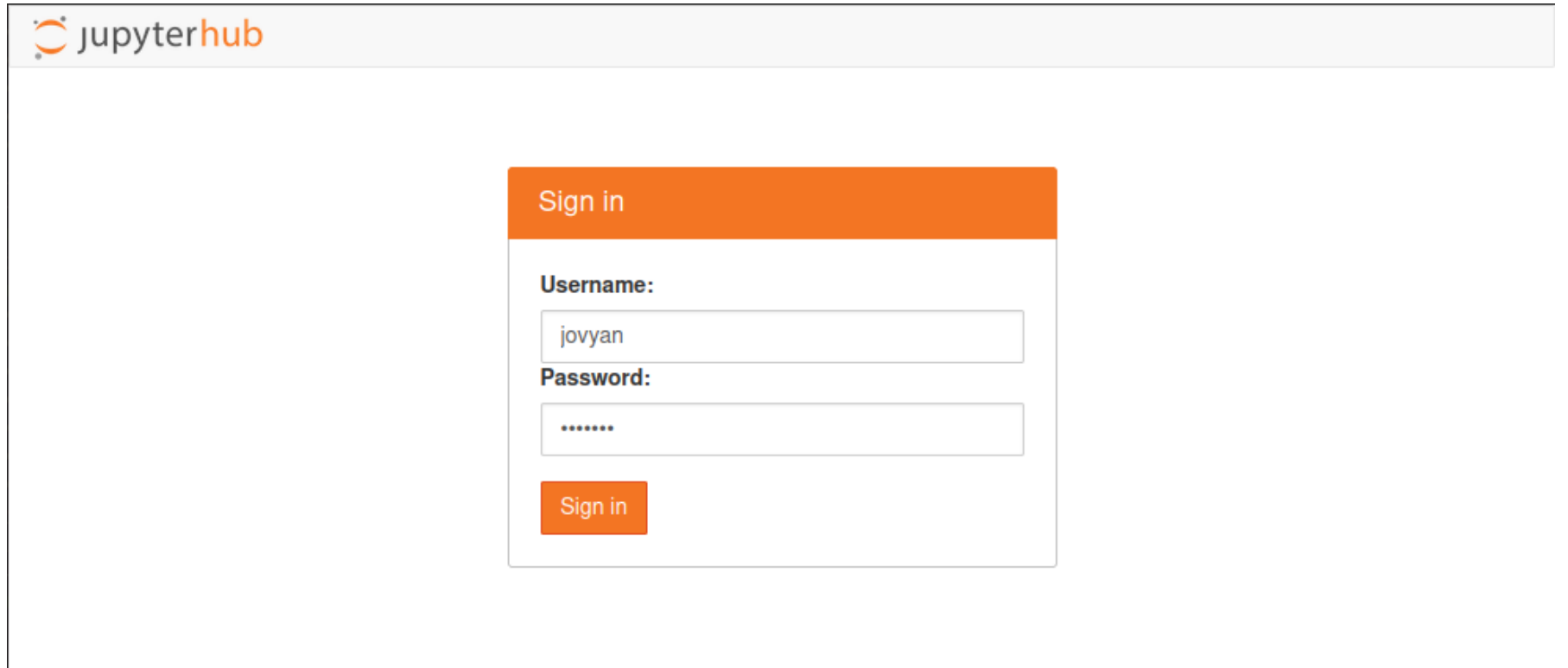
Port ★

5555

I) Création d'un environnement Big Data

2) Calcul distribué : EMR

- Accès à Jupyter Hub !
- Kernel PySpark utilisé



The image shows a screenshot of the JupyterHub login interface. At the top, there is a header bar with the JupyterHub logo and the text "jupyterhub". Below the header, there is a central login form. The form has an orange header with the text "Sign in". Inside the form, there are two input fields: "Username:" with the value "jovyan" and "Password:" with a masked password "*****". Below the password field, there is an orange button labeled "Sign in".

I) Création d'un environnement Big Data


2) Calcul distribué : EMR

- Terminer le cluster
- Cloner si besoin


Cluster: P8_Fruits Terminated Terminated by user request

- Summary
- Application user interfaces
- Monitoring
- Hardware
- Configurations
- Events
- Steps
- Bootstrap actions

Summary

ID: j-RIHGMD6VI6SP
Creation date: 2023-03-01 20:51 (UTC+1)
End date: 2023-03-01 21:51 (UTC+1)
Elapsed time: 59 minutes
After last step completes: Cluster waits
Termination protection: Off
Tags: --
Master public DNS:
ec2-34-254-96-47.eu-west-1.compute.amazonaws.com 
[Connect to the Master Node Using SSH](#)


Application user interfaces

Persistent user interfaces : [Spark history server, YARN timeline server](#)
On-cluster user -- interfaces 

Configuration details

Release label: emr-6.3.0
Hadoop distribution: Amazon
Applications: JupyterHub 1.2.0, TensorFlow 2.4.1, Spark 3.1.1
Log URI: s3://aws-logs-815565965465-eu-west-1/elasticmapreduce/ 
EMRFS consistent view: Disabled
Custom AMI ID: --

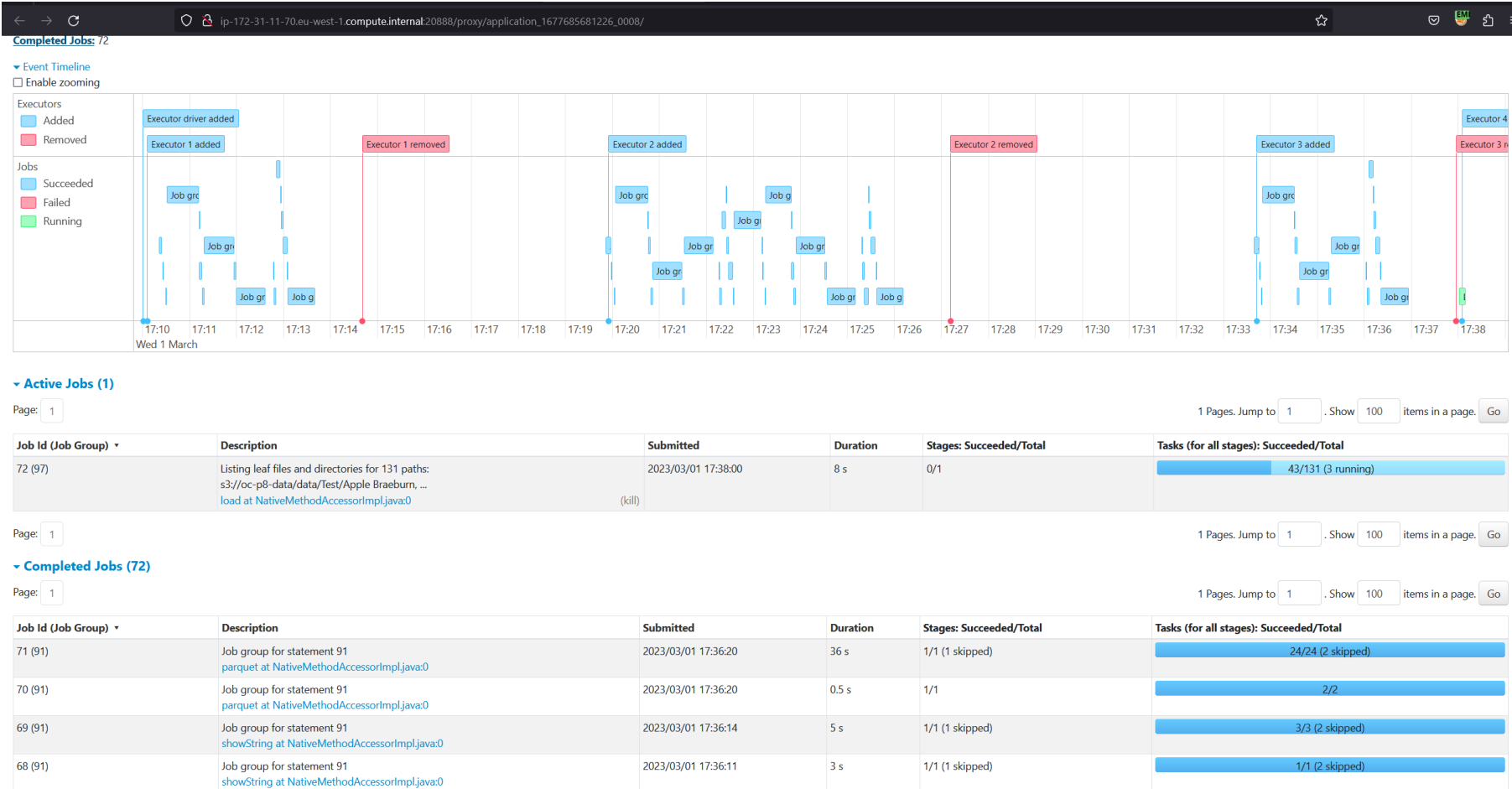
Network and hardware

Availability zone: eu-west-1b
Subnet ID: [subnet-04546476ea2ee5783](#) 
Master: Terminated 1 m5.xlarge
Core: Terminated 2 m5.xlarge
Task: --
Cluster scaling: Not enabled

I) Création d'un environnement Big Data

2) Calcul distribué : EMR

- Suivi des jobs Spark



II) Chaîne traitement des images

Transfert Learning



ACP



II) Chaîne traitement des images

1) Transfert Learning

Accès aux images dans le bucket grâce à la persistance (congif json)

Création automatique du dossier « Results »

1.4 Définition des PATH pour charger les images et enregistrer les résultats

Nous accédons directement à nos **données sur S3** comme si elles étaient **stockées localement**.

```
: PATH = 's3://oc-p8-data/data'
print("PATH :", PATH)

PATH_Data = PATH + '/Test'
PATH_Result = PATH + '/Results'

print('PATH:      ' +
      PATH + '\nPATH_Data:  ' +
      PATH_Data + '\nPATH_Result: ' + PATH_Result)
```

```
PATH : s3://oc-p8-data/data
PATH:      s3://oc-p8-data/data
PATH_Data:  s3://oc-p8-data/data/Test
PATH_Result: s3://oc-p8-data/data/Results
```

II) Chaîne traitement des images

1) Transfert Learning

1.5.1 Chargement des données

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

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

```
+-----+-----+-----+-----+
|      path      | modificationTime | length | content |
+-----+-----+-----+-----+
|s3://oc-p8-data/d...| 2023-02-28 16:01:17 | 7353 | [FF D8 FF E0 00 1... |
|s3://oc-p8-data/d...| 2023-02-28 16:01:18 | 7350 | [FF D8 FF E0 00 1... |
|s3://oc-p8-data/d...| 2023-02-28 16:01:17 | 7349 | [FF D8 FF E0 00 1... |
|s3://oc-p8-data/d...| 2023-02-28 16:01:18 | 7348 | [FF D8 FF E0 00 1... |
|s3://oc-p8-data/d...| 2023-02-28 16:01:18 | 7328 | [FF D8 FF E0 00 1... |
+-----+-----+-----+-----+
only showing top 5 rows
```

Lecture des images en Spark

Je ne conserve que le **path** de l'image et j'ajoute une colonne contenant les **labels** de chaque image :

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

```
root
|-- path: string (nullable = true)
|-- modificationTime: timestamp (nullable = true)
|-- length: long (nullable = true)
|-- content: binary (nullable = true)
|-- label: string (nullable = true)
```

None

```
+-----+-----+
|path                                     | label |
+-----+-----+
|s3://oc-p8-data/data/Test/Watermelon/r_106_100.jpg| Watermelon |
|s3://oc-p8-data/data/Test/Watermelon/r_109_100.jpg| Watermelon |
|s3://oc-p8-data/data/Test/Watermelon/r_108_100.jpg| Watermelon |
|s3://oc-p8-data/data/Test/Watermelon/r_107_100.jpg| Watermelon |
|s3://oc-p8-data/data/Test/Watermelon/r_95_100.jpg | Watermelon |
+-----+-----+
only showing top 5 rows
```

II) Chaîne traitement des images

1) Transfert Learning

Reconnaissance d'images

Architecture du Réseau de Neurones Convolutif

MobileNetV2

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	-	-

1.5.2 Préparation du modèle

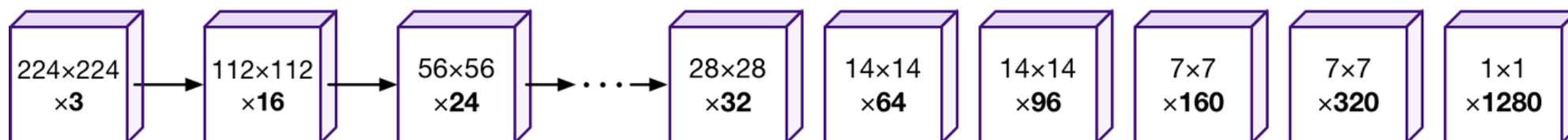
```
model = MobileNetV2(weights='imagenet',
                    include_top=True,
                    input_shape=(224, 224, 3))
```

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

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

```
new_model.summary()
```

```
def model_fn():
    """
    Returns a MobileNetV2 model with top layer removed
    and broadcasted pretrained 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
```



II) Chaîne traitement des images

1) Transfert Learning

1.5.3 Définition du processus de chargement des images et application de leur featurisation à travers l'utilisation de pandas UDF

```
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)
```

Préparation du pre-processing des images pour les adapter au format d'entrée de MobileNetV2

II) Chaîne traitement des images

1) Transfert Learning

1.5.4 Exécutions des actions d'extractions de features

```
print("generating features")
features_df = images.repartition(24).select(col("path"),
                                             col("label"),
                                             featurize_udf("content").alias("features")
                                             )
```

path	label	features
s3://oc-p8-data/d...	Watermelon	[0.8537659, 0.450...
s3://oc-p8-data/d...	Pineapple Mini	[0.0, 4.498071, 0...

only showing top 2 rows

Preprocessing des images et générations des vecteurs de features d'image à l'aide du Transfert Learning

II) Chaîne traitement des images

2) ACP

1.6 PCA

```
def scale_features(df_features):  
    """  
    Returns the DataFrame entered as a parameter scaled using a Standard Scaler  
  
    :param df_features: (pyspark.sql.dataframe.DataFrame)  
    :return:  
    :rtype: pyspark.sql.dataframe.DataFrame  
    """  
  
    # transform array to vector  
    df_features = df_features.withColumn('features', array_to_vector('features'))  
  
    print("Scaling features")  
    # scale data  
    scaler = StandardScaler(  
        inputCol = 'features',  
        outputCol = 'scaled_features',  
        withMean = True,  
        withStd = True  
    ).fit(df_features)  
    df_features_scaled = scaler.transform(df_features)  
  
    return df_features_scaled
```

Réduction de dimension :
Analyse en Composantes Principales

```
def get_pca(df_features_scaled, n_components=40):  
    """  
    make pca on data scaled  
  
    :param df_features_scaled: (pyspark.sql.dataframe.DataFrame)  
    :param n_components: (int) number of components to fit the PCA with  
    """  
    print("Fitting PCA")  
    pca = PCA(  
        k = n_components, # output from Keras model is an array of dim ??  
        inputCol = 'scaled_features',  
        outputCol = 'pca_features'  
    ).fit(df_features_scaled)  
  
    return pca  
  
def get_pca_features(pca_model, df_features_scaled):  
    """  
    :param pca_model:  
    :param df_features_scaled: (pyspark.sql.dataframe.DataFrame)  
    :return:  
    :rtype: pyspark.sql.dataframe.DataFrame  
  
    Getting features based on PCA  
    """  
    df_features_pca = pca_model.transform(df_features_scaled)  
  
    # drop scaled data  
    # transform vector to array for saving  
    df_features_pca = (df_features_pca  
        .drop("scaled_features")  
        .withColumn('features', vector_to_array('features'))  
        .withColumn('pca_features', vector_to_array('pca_features')))  
  
    return df_features_pca
```

II) Chaîne traitement des images

2) ACP

```
def main_pca(df_features):  
    """  
    :param df_features:  
    :return:  
    """  
    # Scaling features  
    print("1-Scaling features")  
    df_features_scaled = scale_features(df_features)  
  
    # Creating pca model fitted on scaled features  
    print("2-Creating pca model fitted on scaled features")  
    pca = get_pca(df_features_scaled)  
  
    # Getting pca features  
    print("3-Getting pca features")  
    df_features_pca = get_pca_features(pca, df_features_scaled)  
  
    return df_features_pca
```

Standardisation : moyenne = 0, écart-type = 1

Application de la PCA en PySpark

```
print("creating PCA")  
features_df_pca = main_pca(features_df)
```

creating PCA
1-Scaling features
Scaling features
2-Creating pca model fitted on scaled features
Fitting PCA
3-Getting pca features

Enregistrement des données traitées au format "parquet" :

```
features_df.show(2)  
features_df_pca.show(2)  
  
features_df_pca.write.mode("overwrite").parquet(PATH_Result)
```

path	label	features
s3://oc-p8-data/d...	Watermelon	[0.8537659, 0.450...
s3://oc-p8-data/d...	Pineapple Mini	[0.0, 4.498071, 0...

only showing top 2 rows

path	label	features	pca_features
s3://oc-p8-data/d...	Watermelon	[0.84154003858566...	[-13.578741245307...
s3://oc-p8-data/d...	Pineapple Mini	[0.0, 4.333771705...	[-5.8529876532093...

only showing top 2 rows

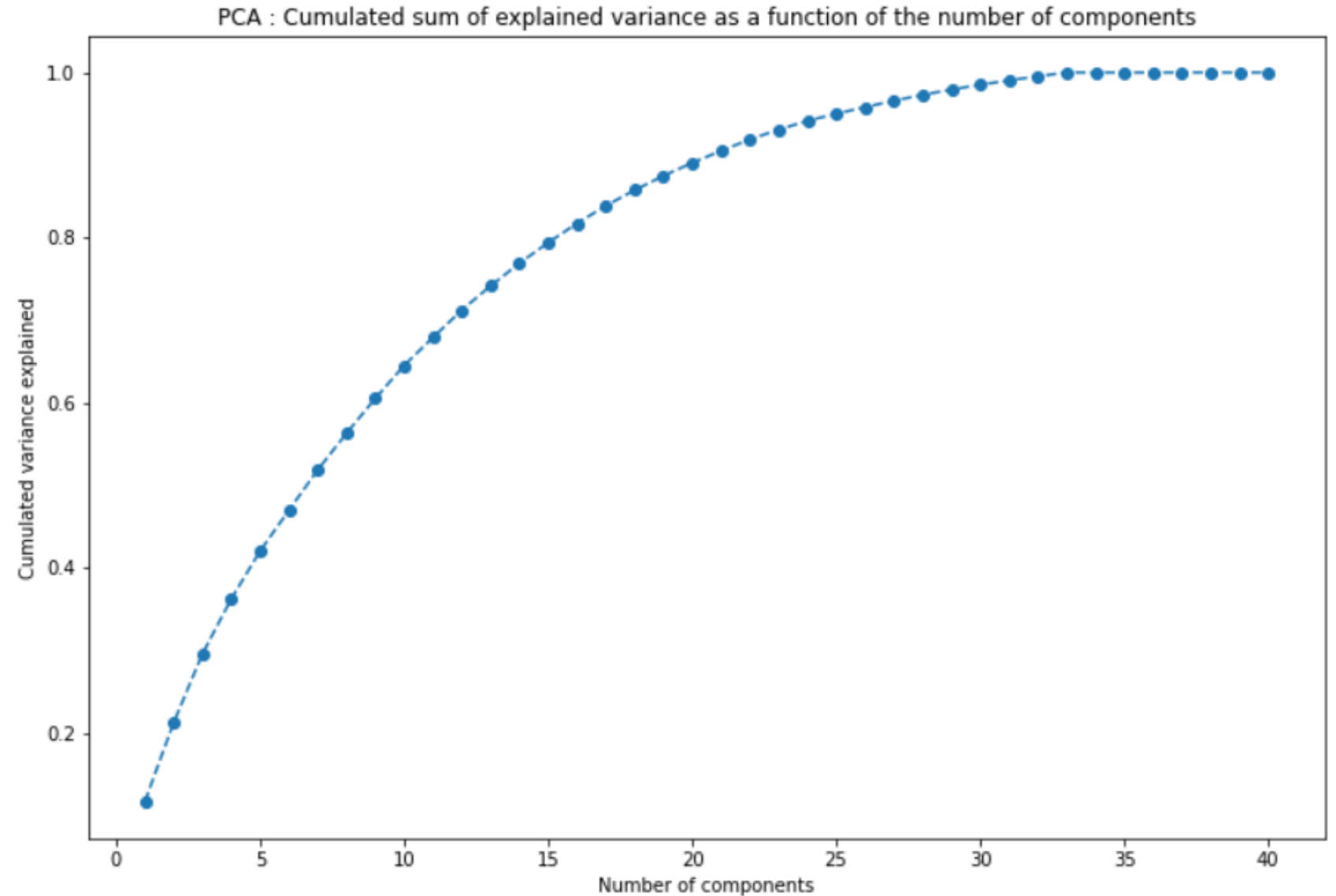
```
print("Computing time : {} seconds".format(time() - t0))
```

Computing time : 2216.4246039390564 seconds

II) Chaîne traitement des images

2) ACP

Avec 35 à 40 composantes, on explique 99% de la variance des données.



II) Chaîne traitement des images

2) ACP

Local

	path	label	features	pca_features
0	file:/drive/My Drive/Al/OC/P8/data/Test1/Apple...	Apple Crimson Snow	[0.0, 0.0, 0.0, 0.0, 0.0, 1.8282229900360107, ...	[7.623144376299502, 4.926324016170813, 4.20514...
1	file:/drive/My Drive/Al/OC/P8/data/Test1/Water...	Watermelon	[0.01282556727528572, 0.4700985848903656, 0.0,...	[-11.049038780238458, -11.189032216304824, 6.0...
2	file:/drive/My Drive/Al/OC/P8/data/Test1/Apple...	Apple Red Yellow 1	[0.37628528475761414, 0.035564910620450974, 0....	[10.306888619480848, -2.384498569050896, -7.86...
3	file:/drive/My Drive/Al/OC/P8/data/Test1/Tomat...	Tomato Yellow	[0.0, 0.7056543827056885, 0.0, 0.0, 0.0, 2.384...	[10.203155492101946, 6.450768106599577, -5.278...
4	file:/drive/My Drive/Al/OC/P8/data/Test1/Apple...	Apple Golden 1	[0.0, 0.003275326220318675, 1.0620614290237427...	[0.19016565809646457, -6.611334741220691, 11.9...

Mon Drive > ... > data > Results_local ▾

Nom ↑	Propriétaire	Dernière modific...	Taille du fichier
 _SUCCESS	moi	27 févr. 2023	0 octet
 ._SUCCESS.crc	moi	27 févr. 2023	8 octets
 .part-00000-2f6729b9-7243-4083-883a-274f2f6afd3b-c000.sna...	moi	27 févr. 2023	120 octets
 .part-00001-2f6729b9-7243-4083-883a-274f2f6afd3b-c000.sna...	moi	27 févr. 2023	72 octets
 .part-00002-2f6729b9-7243-4083-883a-274f2f6afd3b-c000.sna...	moi	27 févr. 2023	72 octets
 .part-00003-2f6729b9-7243-4083-883a-274f2f6afd3b-c000.sna...	moi	27 févr. 2023	76 octets

II) Chaîne traitement des images

2) ACP

EMR et S3

Amazon S3 > Buckets > oc-p8-data > data/ > Results/

Results/ Copy S3 URI

Objects Properties

Objects (25)

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](#)

Refresh Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	_SUCCESS	-	March 1, 2023, 19:53:24 (UTC+01:00)	0 B	Standard
<input type="checkbox"/>	part-00000-fbc6854d-f644-48a0-8b3d-3d6793a459e6-c000.snappy.parquet	parquet	March 1, 2023, 19:51:05 (UTC+01:00)	5.1 MB	Standard
<input type="checkbox"/>	part-00001-fbc6854d-f644-48a0-8b3d-3d6793a459e6-c000.snappy.parquet	parquet	March 1, 2023, 19:51:04 (UTC+01:00)	5.2 MB	Standard
<input type="checkbox"/>	part-00002-fbc6854d-f644-48a0-8b3d-3d6793a459e6-c000.snappy.parquet	parquet	March 1, 2023, 19:51:05 (UTC+01:00)	5.1 MB	Standard
<input type="checkbox"/>	part-00003-fbc6854d-f644-48a0-8b3d-3d6793a459e6-c000.snappy.parquet	parquet	March 1, 2023, 19:51:04 (UTC+01:00)	5.1 MB	Standard
<input type="checkbox"/>	part-00004-fbc6854d-f644-48a0-8b3d-3d6793a459e6-c000.snappy.parquet	parquet	March 1, 2023, 19:51:32 (UTC+01:00)	5.1 MB	Standard
<input type="checkbox"/>	part-00005-fbc6854d-f644-48a0-8b3d-3d6793a459e6-c000.snappy.parquet	parquet	March 1, 2023, 19:51:32 (UTC+01:00)	5.1 MB	Standard

III) Démo

The slide features two horizontal green bars below the title. In the bottom-left corner, there are three overlapping geometric shapes: a teal triangle, a yellow parallelogram, and a green triangle.

Cluster EMR prêt

Lancement du notebook (P8_Notebook_AWS.ipynb) sur Jupyter Hub

Conclusion

- Architecture Big Data
- Architecture basé sur Spark (Hadoop MapReduce)
- Stockage des données sur S3
- Calcul distribué sur un cluster EMR (serveur EC2)
- Featurisation et ACP durent 5 minutes sur 34 images
- Featurisation et ACP dure 36 minutes sur 22668 images
- Sauvegarde au format parquet

- Axes d'améliorations :
- Optimisation des performances PySpark : réduire le temps de calcul des jobs de 5 min
- Coûts de l'infrastructure AWS

Merci !