

Projet 8 : Déployez un modèle dans le cloud

Sommaire

- 3 – Présentation du projet
 - 4 – Présentation des données
 - 5 – Pourquoi un environnement Big Data
 - 6 – MapReduce
 - 7 – Amazon Web Services
 - 10 – Déroulement de l'opération
 - 17 – Conclusion
-

Présentation du projet

- L'objectif de « Fruits! » souhaite se faire connaître en mettant à disposition du grand public une application mobile permettant aux utilisateurs de prendre en photo un fruit et d'obtenir des informations sur ce fruit.
 - Cette application permettrait de sensibiliser le grand public sur la diversité des fruits, une architecture Big Data est nécessaire.
 - Pour cela, il est impératif de développer des scripts en Pyspark et d'utiliser le cloud AWS pour profiter d'une architecture Big Data (EMR, S3, IAM).
 - Bien respecter les contraintes RGPD : serveurs situés sur le territoire européen.
-

Présentation des données

90 423 images et 131 classes

2 jeux de données training
(67692) et test (22688)

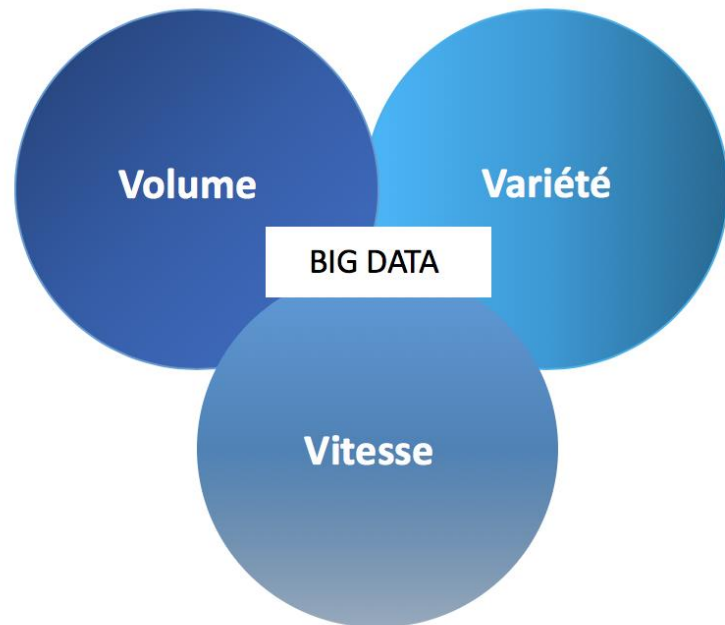
131 dossiers :

- Représente un fruit ou un légume
- Image avec fond blanc et sous 3 axes
- Taille de 100 x 100 pixels en JPG RGB
- Plusieurs variétés pour certains fruits



Pourquoi un environnement BIG DATA ?

les "3 V" sont des caractéristiques qui définissent les enjeux et les avantages des données à grande échelle, communément appelées Big Data.



1. Volume:

1. L'énorme quantité de données manipuler dépasse la capacité des systèmes de gestion de bases de données traditionnels.
2. Amazon S3, par exemple, peut stocker des pétaoctets de données.

2. Vitesse:

1. Les données sont générées et collectées à une vitesse très élevée, ce qui nécessite un traitement en temps quasi-réel.

3. Variété:

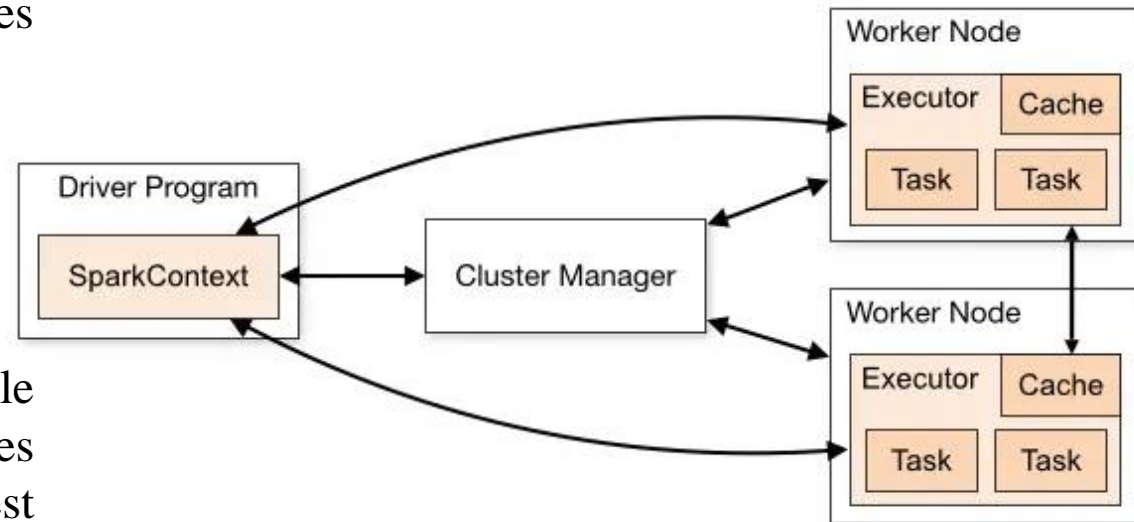
1. Les données proviennent de différentes sources et sont présentées sous différents formats (texte, image, log, vidéo, etc.).
 2. Les services comme Amazon EMR peuvent traiter différentes formes de données de manière efficace.
-

Traitement par calculs distribués (MapReduce)

La distribution des tâches sous forme de "micro-opérations" permet de traiter efficacement de grandes quantités de données sur plusieurs machines en parallèle.

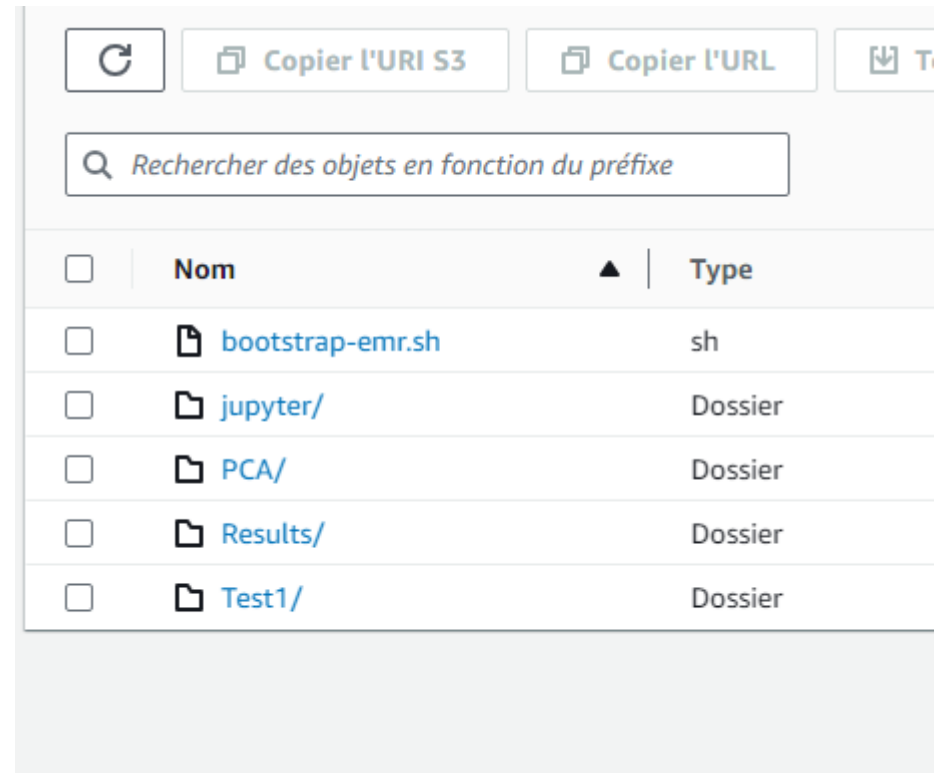


Les résultats sont ensuite agrégés pour former le résultat final. Cette approche est au cœur des systèmes de traitement de données distribuées et est la raison pour laquelle MapReduce est si efficace pour traiter de grands volumes de données.



Amazon Web Services – S3

Création d'un bucket et
envoi des données sur le
cloud (S3)



Amazon Web Services - EMR

Configuration de l'environnement

```
#!/bin/bash
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 pandas
sudo python3 -m pip install pyarrow
sudo python3 -m pip install boto3
sudo python3 -m pip install s3fs
sudo python3 -m pip install fsspec
sudo python3 -m pip install tensorflow
sudo python3 -m pip install matplotlib
```

Amorçage

Version Amazon EMR [Info](#)

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

emr-6.6.0

Offre d'applications



- | | | |
|---|---|--|
| <input type="checkbox"/> Flink 1.14.2 | <input type="checkbox"/> Ganglia 3.7.2 | <input type="checkbox"/> HBase 2.4.4 |
| <input type="checkbox"/> HCatalog 3.1.2 | <input checked="" type="checkbox"/> Hadoop 3.2.1 | <input type="checkbox"/> Hive 3.1.2 |
| <input type="checkbox"/> Hue 4.10.0 | <input type="checkbox"/> JupyterEnterpriseGateway 2.1.0 | <input checked="" type="checkbox"/> JupyterHub 1.4.1 |
| <input type="checkbox"/> Livy 0.7.1 | <input type="checkbox"/> MXNet 1.8.0 | <input type="checkbox"/> Oozie 5.2.1 |
| <input type="checkbox"/> Phoenix 5.1.2 | <input type="checkbox"/> Pig 0.17.0 | <input type="checkbox"/> Presto 0.267 |
| <input checked="" type="checkbox"/> Spark 3.2.0 | <input type="checkbox"/> Sqoop 1.4.7 | <input type="checkbox"/> TensorFlow 2.4.1 |
| <input type="checkbox"/> Tez 0.9.2 | <input type="checkbox"/> Trino 367 | <input type="checkbox"/> Zeppelin 0.10.0 |
| <input type="checkbox"/> ZooKeeper 3.5.7 | | |

Paramètres du catalogue de données AWS Glue

Utilisez le catalogue de données AWS Glue pour fournir un metastore externe à votre application.

- ☐ Utiliser pour les métadonnées de table Spark

Options du système d'exploitation [Info](#)

- ☒ Version Amazon Linux :
- ☐ Amazon Machine Image (AMI) personnalisée
- ☒ Appliquez automatiquement les dernières mises à jour Amazon Linux

Instances

Unité principale

Choisir un type d'instance EC2

m5.xlarge

4 vCore 16 GiB mémoire EBS uniquement stockage
Prix à la demande : 0.214 USD par instance/heure
Prix Spot le plus bas : \$0.081 (eu-west-1b)

► Configuration de nœud - facultatif

Tâche 1 sur 1

Nom

Tâche - 1

Choisir un type d'instance EC2

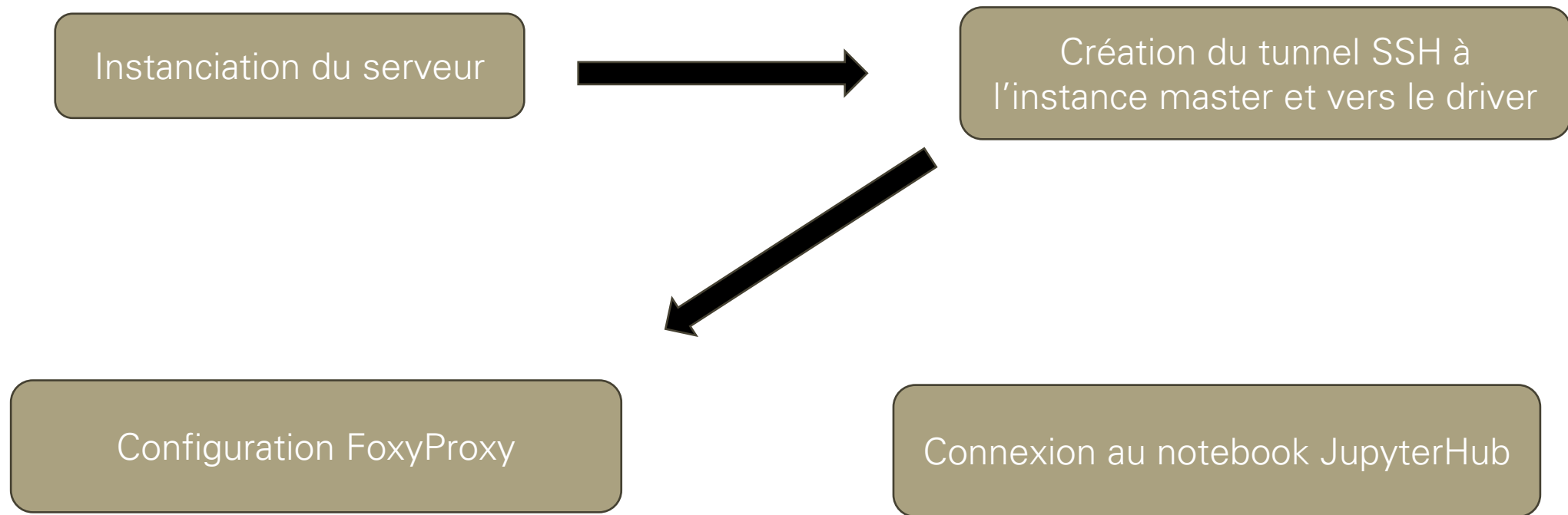
m5.xlarge

4 vCore 16 GiB mémoire EBS uniquement stockage
Prix à la demande : 0.214 USD par instance/heure
Prix Spot le plus bas : \$0.081 (eu-west-1b)

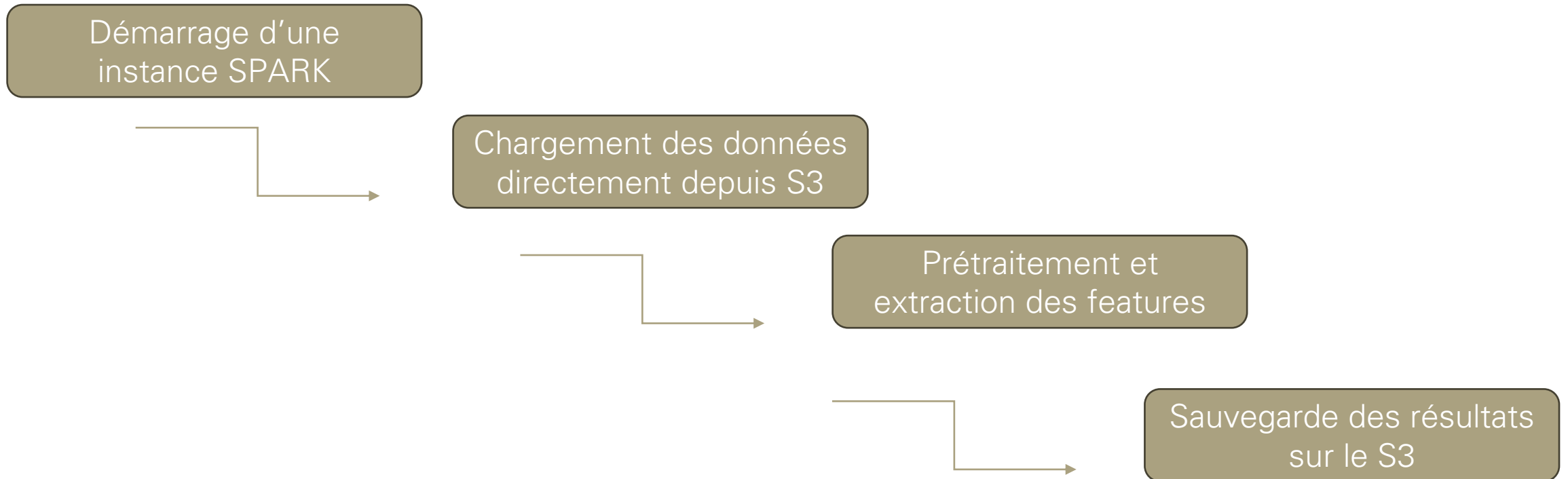
Actions ▼

► Configuration de nœud - facultatif

Lancement de jupyter HUB



Déroulement de l'opération



Déroulement de l'opération

Démarrage d'une
instance SPARK

Entrée [1]: *# L'exécution de cette cellule démarre l'application Spark*

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	User	Current session?
0	application_1697504556601_0001	pyspark	idle	Link	Link	None	✓

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

SparkSession available as 'spark'.

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

Déroulement de l'opération

Chargement des données
directement depuis S3

```
: PATH = 's3://yb-p8-data'
PATH_Data = PATH+'/Test1'
PATH_Result = PATH+'/Results'
PATH_PCA = PATH+'/PCA'

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

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=La

```
PATH:          s3://yb-p8-data
PATH_Data:     s3://yb-p8-data/Test1
PATH_Result:   s3://yb-p8-data/Results
PATH_PCA:      s3://yb-p8-data/PCA
```

```
: [8]: images_sample = spark.read.format("binaryFile") \
      .option("pathGlobFilter", "*.jpg") \
      .option("recursiveFileLookup", "true") \
      .load(PATH_Data)
```

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

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

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

root

Déroulement de l'opération

Prétraitement et
extraction des features

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

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(he
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/m
ering_tf_kernels_1.0_224.h5
14536120/14536120 [=====] - 1s 0us/step

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

```
: 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)
```

Déroulement de l'opération

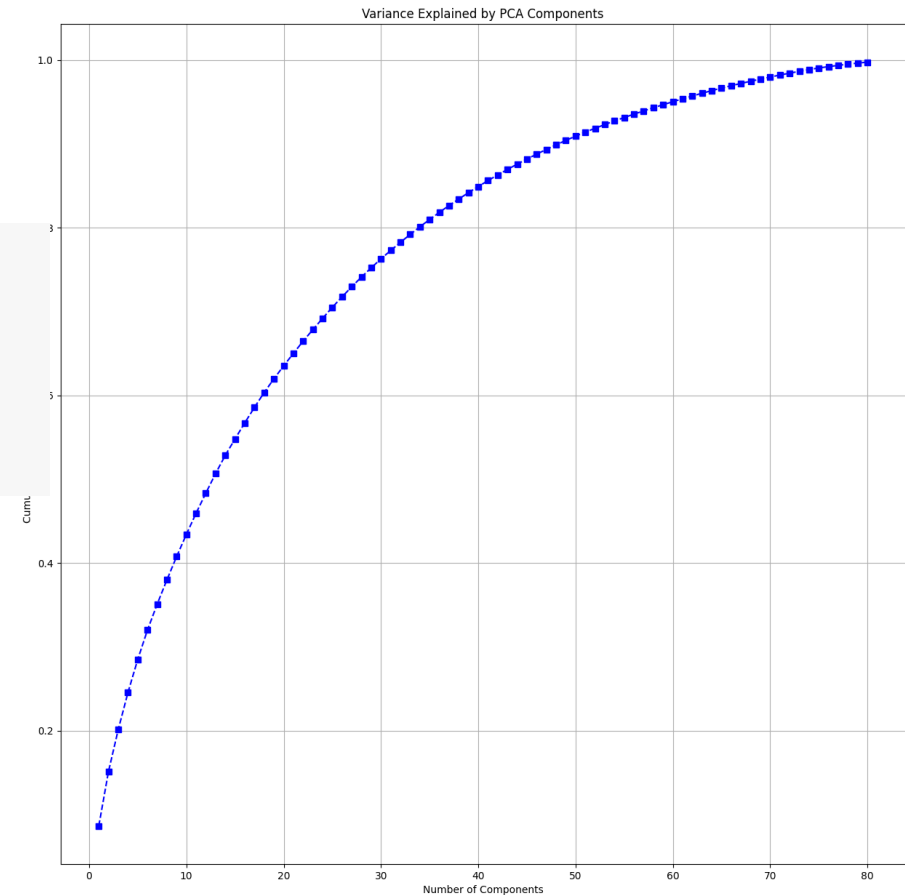
Prétraitement et extraction des features

```
# Initialisation du StandardScaler
scaler = StandardScaler(inputCol="features", outputCol="scaled_features", withStd=True, withMean=True)

# Calcul des statistiques de standardisation et transformation des données
scaler_model = scaler.fit(features_df)
scaled_features_df = scaler_model.transform(features_df)

# Affichage des données standardisées
scaled_features_df.show()
```

```
: from pyspark.ml.feature import PCA
   components = 65
   pca = PCA(k=components, inputCol="features", outputCol="pca_features")
   pca_model = pca.fit(features_df)
   pca_result = pca_model.transform(features_df)
   pca_result.select("label", "pca_features").show()
```



Déroulement de l'opération

Sauvegarde des résultats
sur le S3

```
(pca_result.select("pca_features")).write.mode("overwrite").parquet(PATH_PCA)
```






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

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













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

Déroulement de l'opération

Sauvegarde des résultats
sur le S3

<input type="checkbox"/>	Nom	Type
<input type="checkbox"/>	 bootstrap-emr.sh	sh
<input type="checkbox"/>	 jupyter/	Dossier
<input type="checkbox"/>	 <u>PCA/</u>	Dossier
<input type="checkbox"/>	 Results/	Dossier
<input type="checkbox"/>	 Test1/	Dossier

Objets (25)
Les objets sont les entités fondamentales stockées dans Amazon S3. Vous pouvez utiliser l'[inventaire Amazon S3](#) pour obtenir une liste de tous les objets de votre compartiment. Pour que d'autres personnes puissent accéder à des objets, vous devez leur accorder explicitement des autorisations. [En savoir plus](#)

<input type="checkbox"/>	Nom	Type	Dernière modification	Taille	Classe de stockage
<input type="checkbox"/>	 _SUCCESS	-	17 Oct 2023 03:19:24 AM CEST	0 o	Standard
<input type="checkbox"/>	 part-00000-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:11 AM CEST	3.1 Ko	Standard
<input type="checkbox"/>	 part-00001-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:13 AM CEST	3.6 Ko	Standard
<input type="checkbox"/>	 part-00002-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:11 AM CEST	3.6 Ko	Standard
<input type="checkbox"/>	 part-00003-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:12 AM CEST	3.1 Ko	Standard
<input type="checkbox"/>	 part-00004-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:14 AM CEST	3.1 Ko	Standard
<input type="checkbox"/>	 part-00005-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:15 AM CEST	3.6 Ko	Standard
<input type="checkbox"/>	 part-00006-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:14 AM CEST	3.6 Ko	Standard
<input type="checkbox"/>	 part-00007-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:16 AM CEST	3.1 Ko	Standard
<input type="checkbox"/>	 part-00008-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:16 AM CEST	3.1 Ko	Standard
<input type="checkbox"/>	 part-00009-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:17 AM CEST	3.1 Ko	Standard
<input type="checkbox"/>	 part-00010-c12ca0ad-4165-46d2-9adc-12038c997070-c000.snappy.parquet	parquet	17 Oct 2023 03:19:17 AM CEST	3.6 Ko	Standard

Conclusion

- Mise en place d'un environnement Big Data

AWS, EC2, EMR SPARK

- Passage à l'échelle pour le développement

Stockage S3 illimité

Ajout d'instance pour le calcul et la performance
