Projet n°8 : Déployez un modèle dans le cloud

Parcours Data Scientist - OpenClassrooms

Jérémy Vangansberg - octobre 2021

Introduction (1)

Fruits!

- La start-up **Fruits!** cherche à proposer des solutions innovantes pour la récolte de fruits.
- L'entreprise souhaite dans un premier temps se faire connaître en mettant à disposition du grand public une application qui pourra permettre de prendre en photo un fruit et avoir des informations sur celui-ci. Cette application implique de mettre en place un **moteur de classification d'image**.
- L'application doit être conçu dans un environnement Big Data pour faire face à une potentielle évolution du nombre d'utilisateurs.

Introduction (2)

Fruits!

- Le but du projet est de mettre en place le pipeline de traitement des images.
- Il n’est pas **demandé d’effectuer la modélisation.**

Sommaire

Fruits!

A. Les données
B. Le Big Data et son architecture
C. Les solutions envisagées
D. Databricks : fonctionnement et paramétrage
E. La chaîne de traitement
F. Conclusion

Les données

Le jeu de données brut

Fruits!

- Données d’entraînement : 67692 images (un fruit ou un légume par image)
- Données de test : 22688 images (un fruit ou un légume par image)
- Nombre de classes : 131 (fruits ou légumes)
- Taille des images : 100 x 100 pixels

Le jeu de données

Fruits!

- Parmi ce jeu de données, j’ai sélectionné **8 images** pour faire une démonstration du pipeline.
- Classes sélectionnées :
  - **Banane**
  - **Cerise**
  - **Aubergine**
  - **Kiwi**
- Note : je propose également un pipeline alternatif où je traite l’intégralité des images.

Le Big Data et son architecture

Le Big Data

Fruits!

- **Quelques chiffres en 2020**:
  - 40 zétaoctets (10^{21} octets) de données générées.
  - 150 millions d'e-mails envoyés par minute.
- Qu'est ce que le Big Data ?
  - C'est lorsque le **volume de données est trop important** pour utiliser des **méthodes traditionnelles.**
- Quand commence le Big Data ?
  - **Ça dépend**, par exemple 100 Go de données à traiter peut paraître être un volume important mais à l'échelle de l'un des acteurs des GAFAM, c'est une goutte d'eau. D'une manière générale, on retient que lorsque les données **sont trop grandes pour être stockées en RAM**, on est confronté à un problème de Big Data.

Les 3V du Big Data

Fruits!

- J’ai évoqué précédemment que le **volume** de données est ce qui caractérise le Big Data. En effet, il est l’un des 3 V du Big Data mais il y a également :

- La **vélocité**, il faut être capable de traiter les données dans des délais acceptables. Par exemple, imaginons qu’une société mette une heure à traiter 1 Go de données de log pour faire un rapport journalier. Si la société produit plus de 24 Go de données par jour, elle n’est déjà plus capable de traiter les données de manière journalière.

- La **variété**, le big data se caractérise aussi par l’hétérogénéité des formats. Ils peuvent être structurés (JSON), semi-structurés (logs), non-structurés (texte et image).

Le cloud : l’émergence du Big Data

- **Avant l’ère du cloud**, les entreprises devaient s’affranchir de frais considérables pour acheter et maintenir des serveurs.

- Désormais avec des acteurs comme **AWS**, **Microsoft Azure et Google Cloud** des capacités de calcul importantes sont à la disposition de tout le monde à des coûts acceptables.

3 briques du Big Data

Fruits!

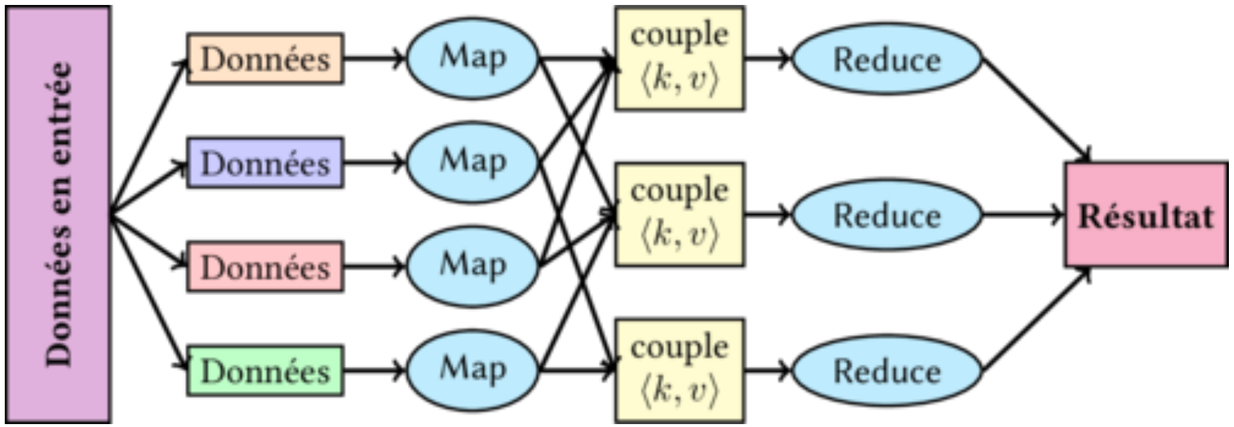
- Map Reduce
- Hadoop Map Reduce
- Spark

Map Reduce

Fruits!

- L’algorithme de Map Reduce, est le composant fondamental du Big Data. Il permet de **paralléliser** et **distribuer** des calculs.

ARCHITECTURE MAP REDUCE



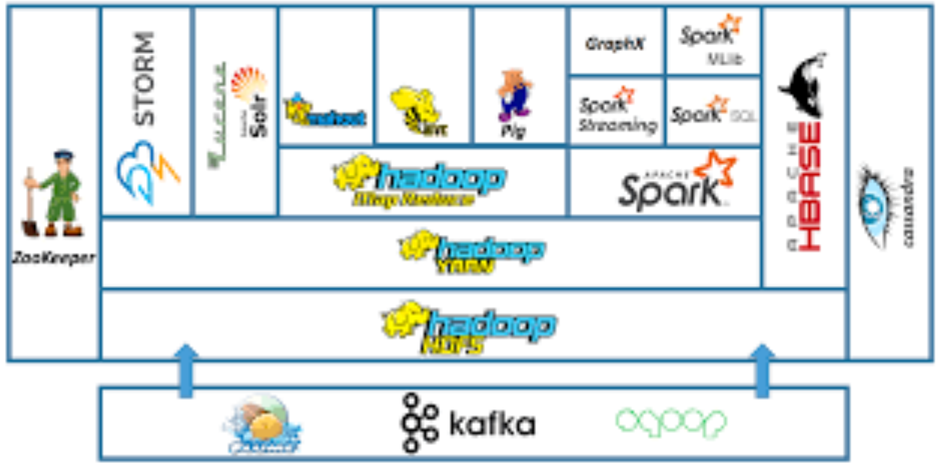
- Cette architecture **développée par Google** au début des années 2000 s’articule autour de deux opérations : Map et Reduce. La première permet **d’appliquer une fonction** à des données qui ont été au préalable partitionnées. La seconde permet d’**agréger les résultats**.

Fruits!

- C’est un **projet open-source** inspiré des travaux de Google qui permet d’appliquer l’architecture Map Reduce de manière **générique**. Le projet a rejoint la fondation Apache en 2008.

- Aujourd’hui tout un Écosystème existe autour d’Hadoop.

ÉCOSYSTÈME HADOOP



Fruits!

- L’un des inconvénients majeurs lorsque l’on veut travailler avec Hadoop MapReduce, est que l’on doit **traduire un problème en un enchaînement d’opération *map* et *reduce***. De plus Hadoop MapReduce écrit les données sur disque entre chaque opération.

- Spark permet de **s’abstraire** de MapReduce car il s’utilise comme un langage de haut niveau.

- Spark va stocker les données en **RAM**, ce qui peut accélérer le traitement des données par un facteur entre 10 et 100.

Architecture Spark : les composantes

- Spark dépend de plusieurs éléments :
  - Hadoop
  - Java
  - Scala
- Pour pouvoir utiliser le langage python avec spark, il faut se servir de la libraire **Pyspark.**
- De plus, pour pouvoir mettre en place le pipeline de traitement des images, j'aurai également besoin de :
  - pandas, PIL, tensorflow, os, io, numpy, boto3

Les solutions envisagées

Les solutions envisagées

Fruits!

Service web qui propose des capacités de calcul dans le cloud

Service clé en main qui permet de faire du machine learning

Semblable à EC2 mais spécialisé dans le traitement de big data

Entreprise qui travaille avec AWS. Elle permet de lancer des clusters et des notebooks pré-configurés pour le big data

Difficultés rencontrées

Fruits!

- La principale difficulté a été liée aux conflits entre les différentes versions.
- En effet les **versions évoluent rapidement** et les tutoriels deviennent rapidement obsolètes.
- Pour pouvoir me concentrer sur le pipeline de traitement des données et de résoudre les problèmes liés à la configuration, j’ai décidé de me tourner vers **Databricks**.

Databricks

Databricks

Fruits!

Avantage :

- Databricks permet de réduire le temps de déploiement d’un modèle de big data

Inconvénient :

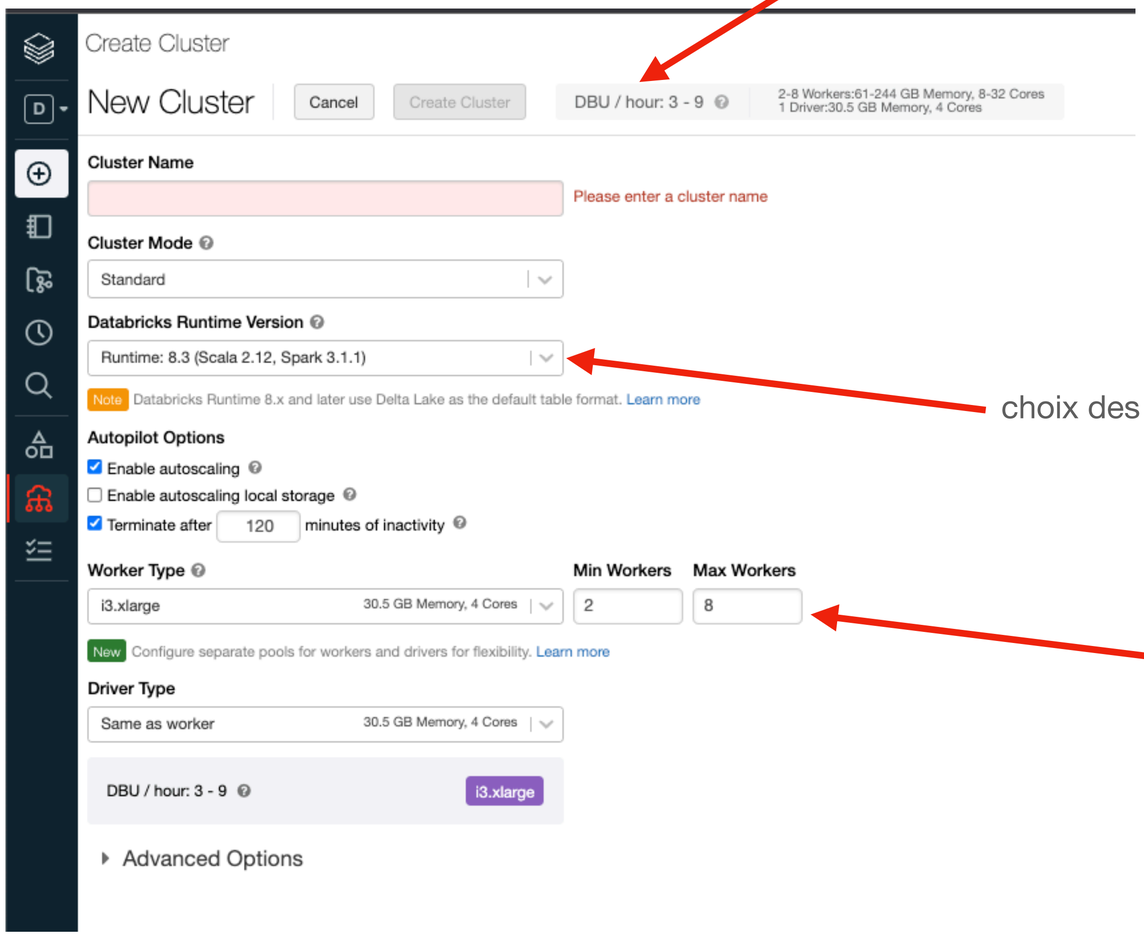
- Solution payante

Databricks - marche à suivre

Fruits!

1. Créer un compte Databricks et profiter des 14 jours d'essai gratuit
2. Associer son compte avec AWS en mode *Cross Account Role*
   - https://docs.databricks.com/administration-guide/account-api/iam-role.html
3. Définir un *bucket* S3 où stocker les informations de Databricks
4. Paramétrer les politiques de sécurité du *bucket* S3 où le jeu de données est stocké
   - https://docs.databricks.com/administration-guide/cloud-configurations/aws/instance-profiles.html
5. Déployer un *cluster* (avec *l'instance profile* défini au point précédent)
6. Lancer un *notebook* associé à ce *cluster*

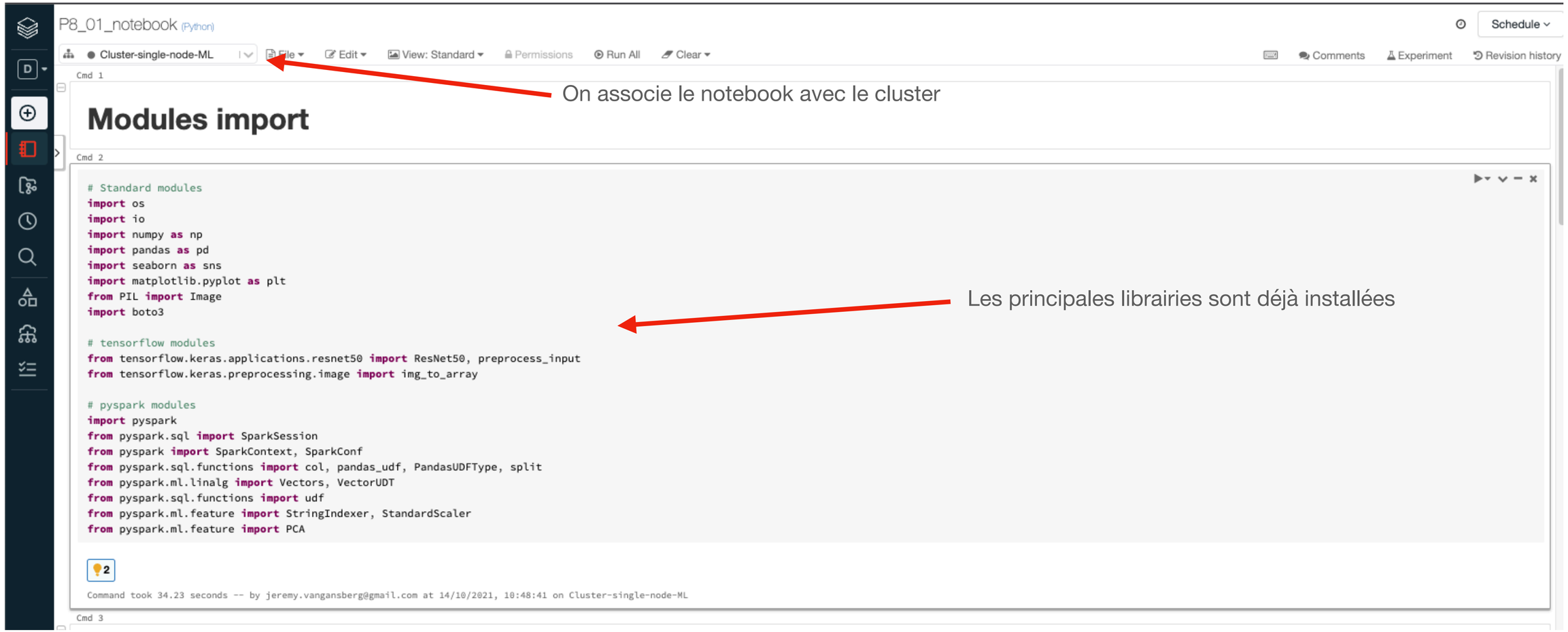
Databricks : cluster



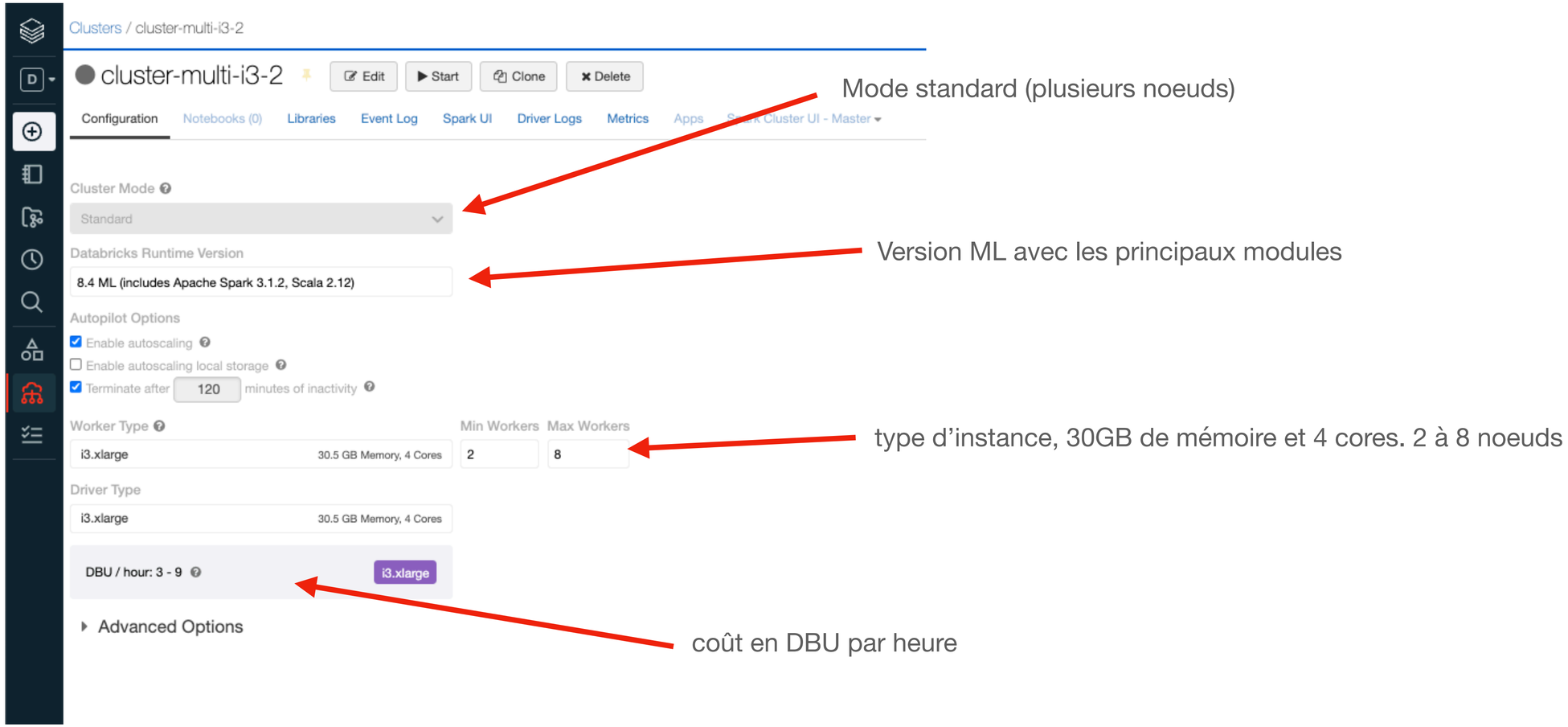
- Databricks nous permet de lancer un cluster qui va lancer une ou plusieurs instances ec2

Databricks : notebook

Fruits!



Les paramètres du cluster



Mode standard (plusieurs noeuds)

Version ML avec les principaux modules

type d'instance, 30GB de mémoire et 4 cores. 2 à 8 noeuds

coût en DBU par heure

La chaîne de traitement

Stratégie

Fruits!

- Afin de réduire les coûts associés aux clusters, j'ai décidé de limiter le jeu de données à **8 images** de 4 classes différentes.
- Pour extraire les features des images, j'ai effectué du *transfer learning* en utilisant le modèle **ResNet50.**
- Utiliser une série de fonctions pour transformer les données binaires en une liste de vecteur.
- Réduire la dimension de ce vecteur à 16 dimensions grâce à une PCA.

Étapes (1)

Fruits!

1. Initialisation d'une session spark

Cmd 4

```
# Instanciation of spark with credientials

spark = (SparkSession.builder
         .appName('jv-p8')
         .config('spark.hadoop.fs.s3a.access.key', [redacted])
         .config('spark.hadoop.fs.s3a.secret.key', '[redacted]')
         .config('spark.hadoop.fs.s3a.impl', 'org.apache.hadoop.fs.s3a.S3AFileSystem')
         .getOrCreate()
        )

sc = spark.sparkContext
sc.setSystemProperty('com.amazonaws.services.s3a.enableV4', 'true')
sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint", "s3.eu-west-1.amazonaws.com")
```

```
Command took 0.08 seconds -- by jeremy.vangansberg@gmail.com at 14/10/2021, 10:48:41 on Cluster-single-node-ML
```

2. Chargement des données

Cmd 7

```
# Loading the data

s3_url = "s3a://s3-jv-ocr/reduced_dataset/*"
image_df = image_df = spark.read.format("binaryfile").load(s3_url)
image_df.printSchema()
```

```
▸ image_df: pyspark.sql.dataframe.DataFrame = [path: string, modificationTime: timestamp ... 2 more fields]
root
 |-- path: string (nullable = true)
 |-- modificationTime: timestamp (nullable = true)
 |-- length: long (nullable = true)
 |-- content: binary (nullable = true)
```

```
Command took 2.36 seconds -- by jeremy.vangansberg@gmail.com at 14/10/2021, 10:48:41 on Cluster-single-node-ML
```

Étapes (2)

Fruits!

3. Modification du jeu de données

```
# Data restructuring

image_df = image_df.withColumn('label', split(col('path'), '/').getItem(4))
image_df = image_df.select('path', 'content', 'label')
image_df.show()
```

▸ (2) Spark Jobs

▸ image_df: pyspark.sql.dataframe.DataFrame = [path: string, content: binary ... 1 more field]

```
+--------------------+--------------------+--------+
|                path|             content|   label|
+--------------------+--------------------+--------+
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|    Kiwi|
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|    Kiwi|
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|Cherry_1|
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|Cherry_1|
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|  Banana|
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|  Banana|
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|Eggplant|
|s3a://s3-jv-ocr/r...|[FF D8 FF E0 00 1...|Eggplant|
+--------------------+--------------------+--------+
```

Command took 2.57 seconds -- by jeremy.vangansberg@gmail.com at 14/10/2021, 10:48:41 on Cluster-single-node-ML

Étapes (3)

Fruits!

4. Chargement du modèle

Cmd 13

```
# Deeplearning model instanciation

model = ResNet50(include_top=False)
model.summary()  # verify that the top layer is removed
```

```
conv5_block3_1_relu (Activation (None, None, None, 5 0           conv5_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_2_conv (Conv2D)    (None, None, None, 5 2359808     conv5_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, None, None, 5 2048        conv5_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, None, None, 5 0           conv5_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, None, None, 2 1050624     conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, None, None, 2 8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, None, None, 2 0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, None, None, 2 0           conv5_block3_add[0][0]
==================================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
__________________________________________________________________________________________________
```

Command took 5.41 seconds -- by jeremy.vangansberg@gmail.com at 14/10/2021, 10:48:41 on Cluster-single-node-ML

Étapes (4)

Fruits!

5. Définir une série de fonction pour transformer les données en vecteur

```
bc_model_weights = spark.sparkContext.broadcast(model.get_weights())
def model_fn():
    """
    Returns a ResNet50 model with top layer removed and broadcasted pretrained weights.
    """
    model = ResNet50(weights=None, include_top=False)
    model.set_weights(bc_model_weights.value)
    return model

def preprocess(content):
    """
    Preprocesses raw image bytes for prediction.
    """
    img = Image.open(io.BytesIO(content)).resize([50, 50])
    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)
    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.
    '''
    model = model_fn()
    for content_series in content_series_iter:
        yield featurize_series(model, content_series)
```

Étapes (5)

Fruits!

6. Réduction de la dimension

Cmd 18

Dimensions reduction with a PCA

Cmd 19

```
# standard scaler before doing the PCA

scaler = StandardScaler(withMean=True, withStd=True,
                        inputCol='features',
                        outputCol='features_scaled')
scaler_fitted = scaler.fit(features_df)
features_scaled_df = scaler_fitted.transform(features_df)
```

▸ (2) Spark Jobs

▸ features_scaled_df: pyspark.sql.dataframe.DataFrame = [path: string, label: string ... 2 more fields]

Command took 8.99 seconds -- by jeremy.vangansberg@gmail.com at 19/10/2021, 13:36:10 on cluster-multi-i3-2

Cmd 20

```
# I choose a number of dimensions pretty low because the cluster is on a single node to reduce the

pca = PCA(k=16, inputCol="features_scaled", outputCol="features_pca")
model = pca.fit(features_scaled_df)
df_transformed = model.transform(features_scaled_df)
```

▸ (5) Spark Jobs

▸ df_transformed: pyspark.sql.dataframe.DataFrame = [path: string, label: string ... 3 more fields]

Command took 34.16 minutes -- by jeremy.vangansberg@gmail.com at 19/10/2021, 13:36:10 on cluster-multi-i3-2

Étapes (6)

Fruits!

7. Export des données vers S3

Cmd 22

Export the data scaled to S3

Cmd 23

```
# Data selection and transformation to pandas

df_pandas = df_transformed.select('path', 'label', 'features_pca').toPandas()
```

▸ (1) Spark Jobs

```
/databricks/spark/python/pyspark/sql/pandas/conversion.py:92: UserWarning: toPandas attempted Arrow optim
  Unable to convert the field features_pca. If this column is not necessary, you may consider dropping it
Direct cause: Unsupported type in conversion to Arrow: VectorUDT
Attempting non-optimization as 'spark.sql.execution.arrow.pyspark.fallback.enabled' is set to true.
  warnings.warn(msg)
```

Command took 7.16 seconds -- by jeremy.vangansberg@gmail.com at 19/10/2021, 13:36:10 on cluster-multi-i3-2

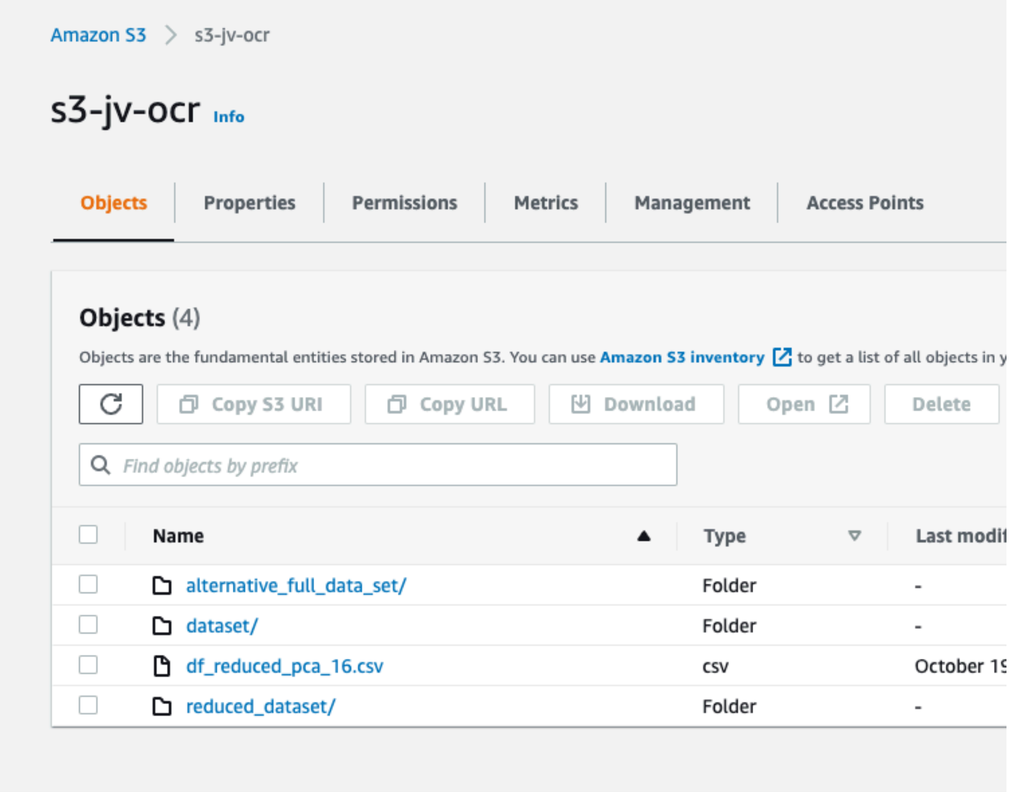
Cmd 24

```
# Export to S3 bucket

from io import StringIO

bucket = "s3-jv-ocr"

csv_buffer = StringIO()
df_pandas.to_csv(csv_buffer)

s3_resource = boto3.resource('s3')

s3_resource.Object(bucket, 'df_reduced_pca_16.csv').put(Body=csv_buffer.getvalue())
```

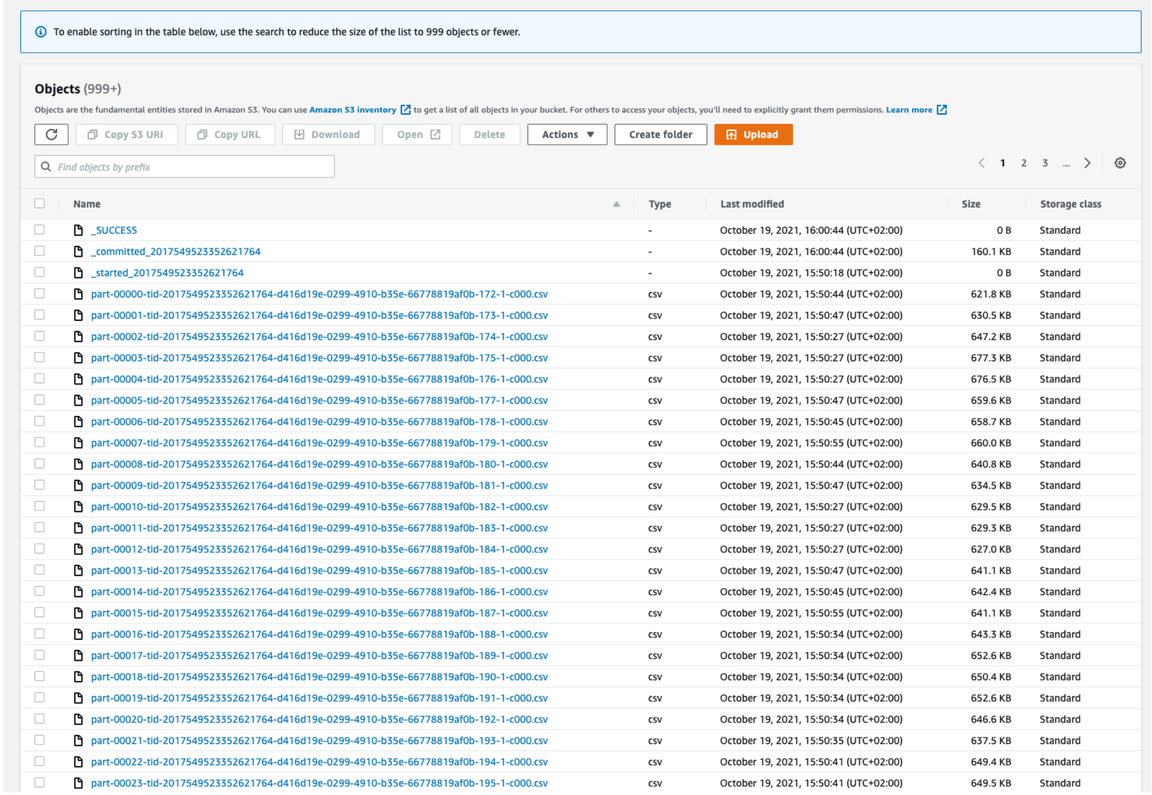
Les données dans le S3

Fruits!

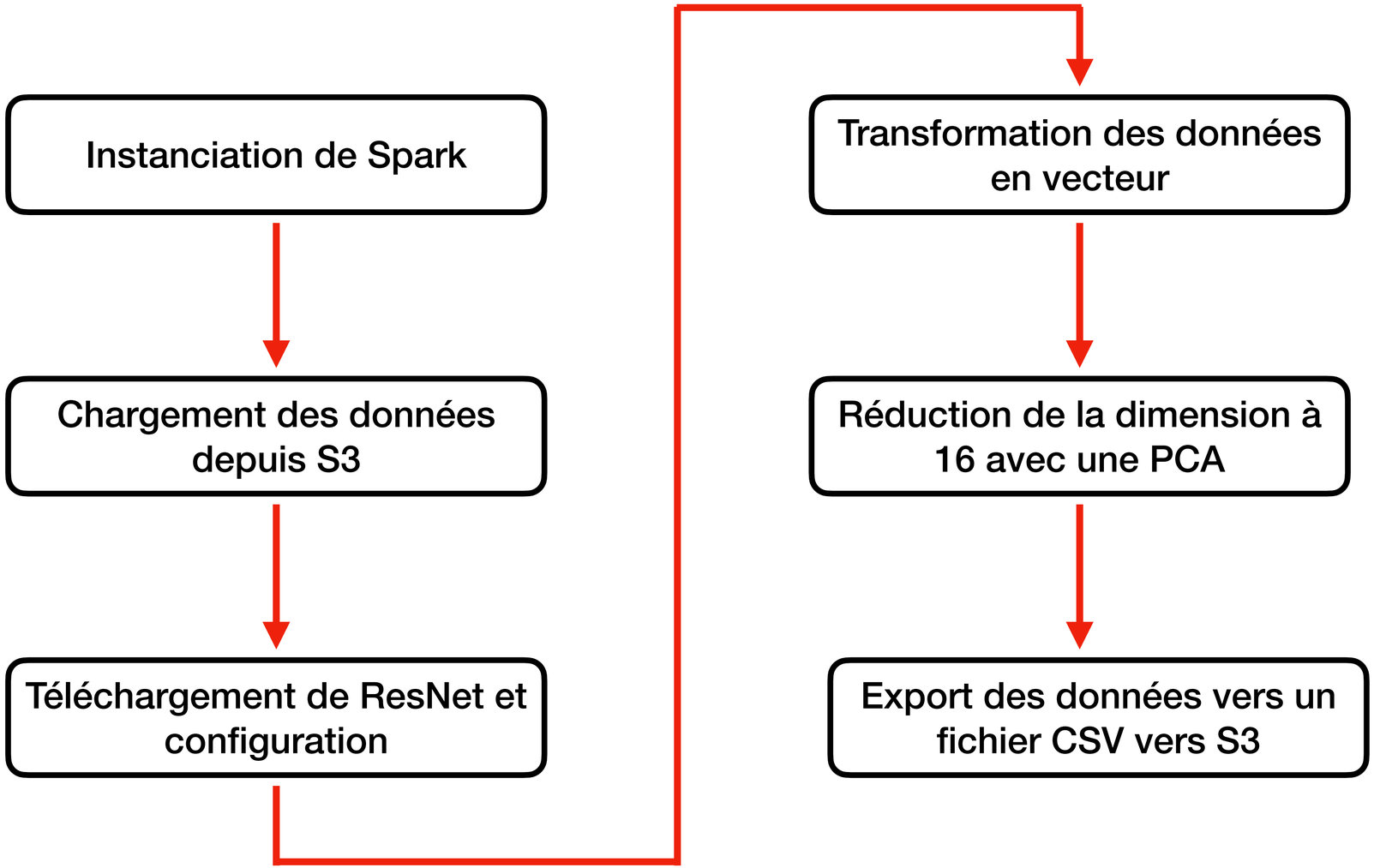
Bucket s3-jv-ocr



Les partitions de fichiers CSV



Récapitulatif de la chaîne de traitement



Version alternative (1)

Fruits!

- **Le projet imposait d'effectuer une PCA pour réduire la dimension**, cependant cette méthode a été une contrainte à plusieurs niveaux :
- Premièrement, j'ai dû réduire considérablement la taille du jeu de données de 67000 images à 8
- La PCA met environ 35 minutes à s'exécuter
- En prenant du recul, je suis arrivé à la conclusion que pour réduire la dimension des images, il était plus efficient d'utiliser une fonction *resize* de PIL

Version alternative (2)

- La fonction *resize* permet de **redimensionner** une image avec la méthode *Nearest-neighbor interpolation.*
- En procédant de cette manière, j’ai réussi à réduire les dimensions des 67 000 images en 20 x 20 en quelques secondes.

Conclusion

Fruits!

- Configurer un serveur manuellement pour utiliser Spark est un **challenge** important à cause de l'évolution rapide des versions.
- Les solutions comme Databricks permettent de déployer des modèles plus rapidement.
- La réduction de dimension avec une **PCA est gourmande en ressources de calcul**, même en distribuant les calculs.
- En utilisant une fonction de resizing uniquement, il est possible de traiter l'intégralité du jeu de données en quelques secondes.

Recommandations : Data Architect vs Data Scientist

Fruits!

Selon moi :

- Le paramétrage d'une instance pour utiliser Spark est le **coeur d'activité d'un poste de Data Architect**.

- Les Data Scientists doivent plutôt s'orienter vers des solutions comme Databricks pour **gagner du temps sur le déploiement** de solution Big Data afin de **se concentrer sur le pipeline** de traitement des données.