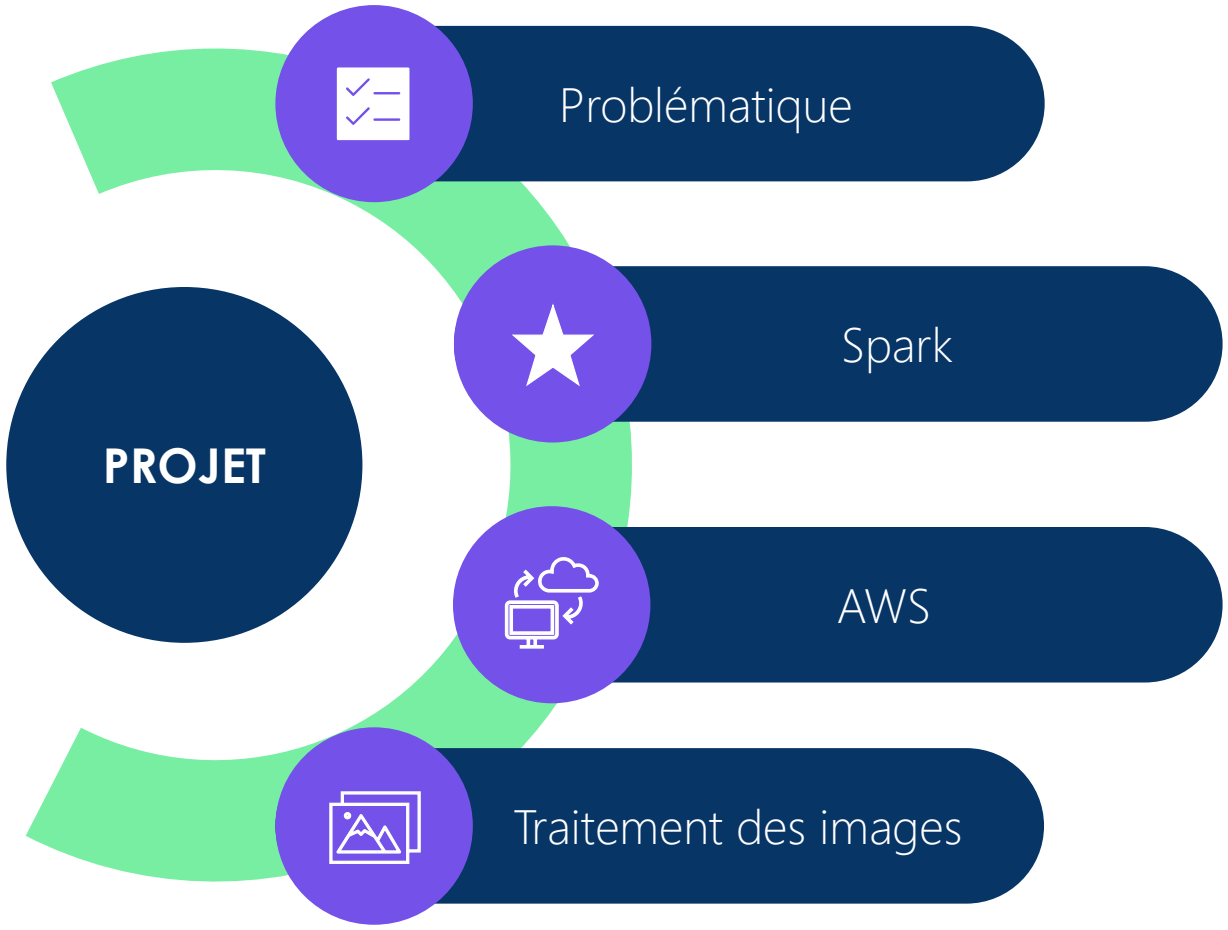
Projet N°8 :
Déployez un modèle dans le cloud

Agustin Bunader (autofinancé)
Soutenance de Projet
Octobre 2021

Programme



Problématique – Présentation

Contexte :

- L'entreprise souhaite développer une application mobile qui permettrait aux utilisateurs de prendre en photo un fruit et d'obtenir des informations sur ce fruit
- Préserver la biodiversité des fruits en permettant des traitements spécifiques pour chaque espèce de fruits en développant des robots cueilleurs intelligents

Objectifs :

- 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.
- Construire une première version de l'architecture Big Data nécessaire

Mission :

- Développer dans un environnement Big Data une première chaîne de traitement des données qui comprendra le pre-processing et une étape de réduction de dimension

Sources :

- Jeux des images contentant 131 fruits avec 67692 photos dans le set d'entraînement et 22688 photos dans le set de test, toutes les images sont de taille 100x100 pixels

Contraintes :

- Utilisation des services avec une puissance de calcul très faible (même si elle est supérieure à celle proposée gratuitement)
- Utilisation de Spark et les services EC2 et S3 d'AWS

Problématique – Données

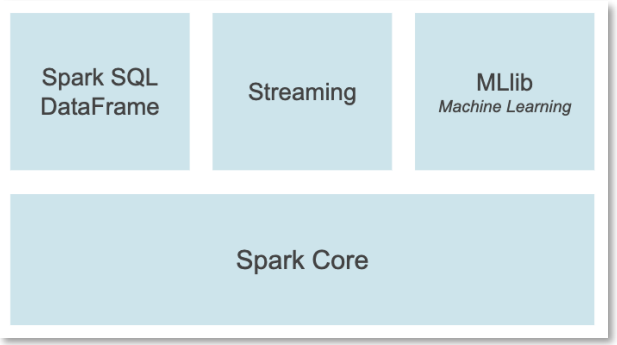
Détails :

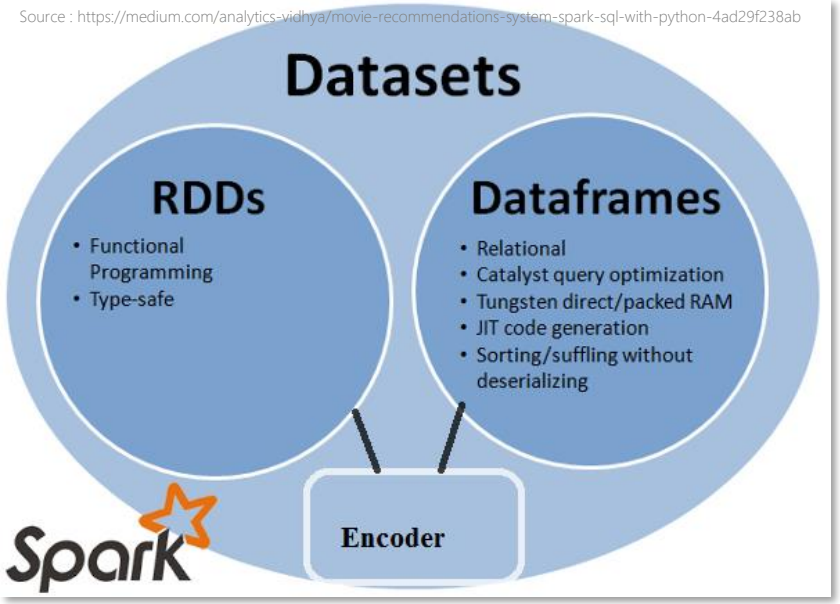
- 131 fruits
- Chaque image a une taille de 100x100 pixels
- Vue 360° de chaque fruit sur fond blanc et en format jpg

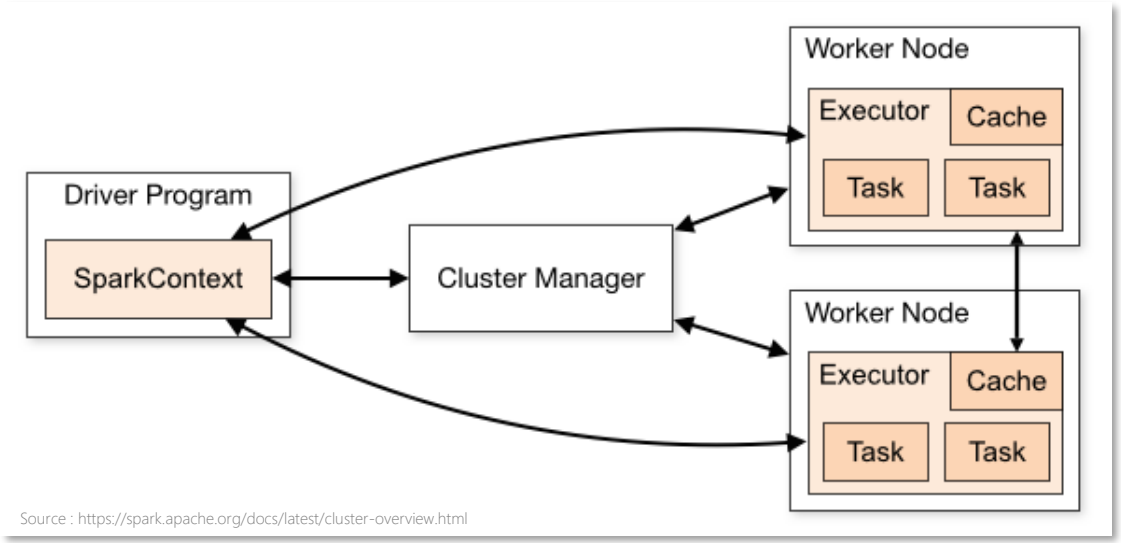
Spark – PySpark

Qu'est-ce que PySpark ?

- Il s'agit d'une API de Python pour Spark car Spark est développé en Scala
- Elle permet l'interaction avec les Resilient Distributed Datasets (RDDs) de Spark avec Python en utilisant la librairie Py4J qui permet l'interaction dynamique avec des objets JVM (Java Virtual Machine)
- Elle permet l'utilisation de Dataframes, ce qui permet à Spark d'exécuter des optimisations sur des requêtes avec un code plus simplifié sans besoin de traiter les RDD avec SQL



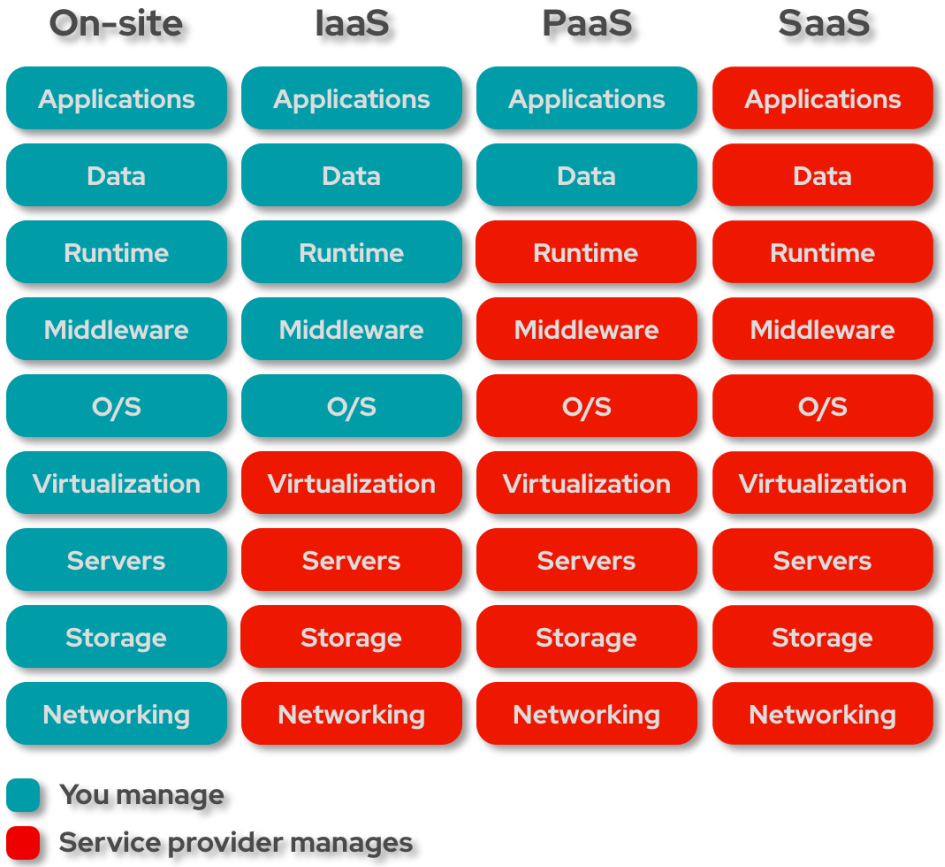




AWS – Qu'est que c'est ?

Introduction :

- Service de cloud computing à la demande, les plus populaires sont Elastic Compute Cloud (EC2) et Simple Storage Service (S3)
- EC2 est une Infrastructure as a Service (Iaas), car Amazon fournit un accès à une partie de ses serveurs mais c'est à l'utilisateur de gérer le système opératif, runtime et data
- Connexion avec S3 utilisant la librairie boto3 (AWS SDK pour Python)

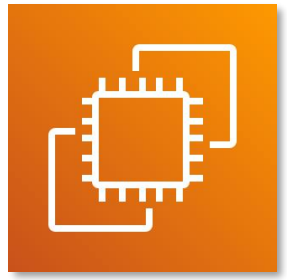


Source : https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas

AWS – EC2 Choix

T2 Instance :

- Instance à usage général à faible coût avec la possibilité de booster en cas de besoin
- Idéal pour les bases de données, environnements de développement, code repositories et prototypage de produits



Name	vCPUs	RAM (GiB)	CPU Credits/hr	On-Demand Price/hr*	1-yr Reserved Instance Effective Hourly*	3-yr Reserved Instance Effective Hourly*
t2.nano	1	0.5	3	$0.0058	$0.003	$0.002
t2.micro	1	1.0	6	$0.0116	$0.007	$0.005
t2.small	1	2.0	12	$0.023	$0.014	$0.009
t2.medium	2	4.0	24	$0.0464	$0.031	$0.021
t2.large	2	8.0	36	$0.0928	$0.055	$0.037
t2.xlarge	4	16.0	54	$0.1856	$0.110	$0.074
t2.2xlarge	8	32.0	81	$0.3712	$0.219	$0.148

Source : https://aws.amazon.com/ec2/instance-types/t2/

- t2.large : 2 vCPU avec 8Go RAM et 30Go de stockage
- Security group : All traffic
- Connection SSH avec PuTTY utilisant key pair
- Elastic IP addresse

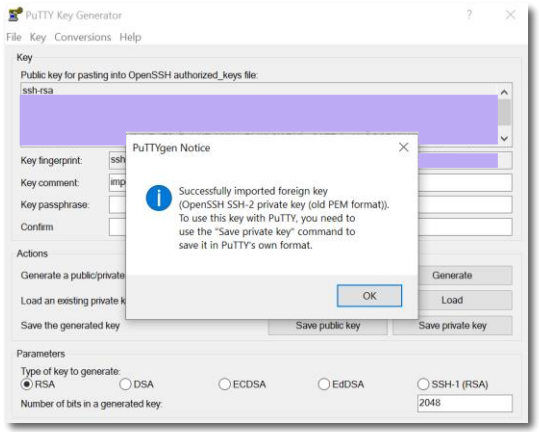
- Ubuntu Server 18.04 64bits
- Python 3.8.8
- Java 1.8.0_292
- Conda 4.10.3
- Spark 3.1.2
- Hadoop 2.7
- TensorFlow 2.3

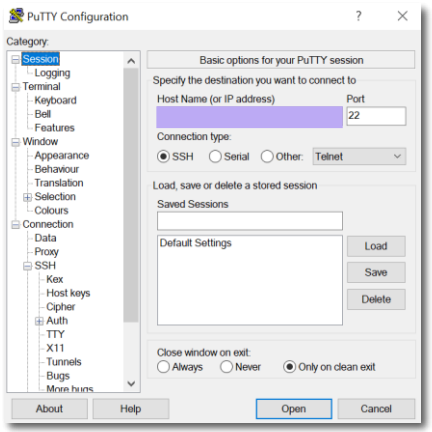
```
(base) ubuntu@ip-172-31-40-178:~$ spark-shell --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Scala version 2.12.10, OpenJDK 64-Bit Server VM, 1.8.0_292
Branch HEAD
Compiled by user centos on 2021-05-24T04:46:13Z
Revision de351e30a90dd988b133b3d00fa6218bfcaba8b8
Url https://github.com/apache/spark
```

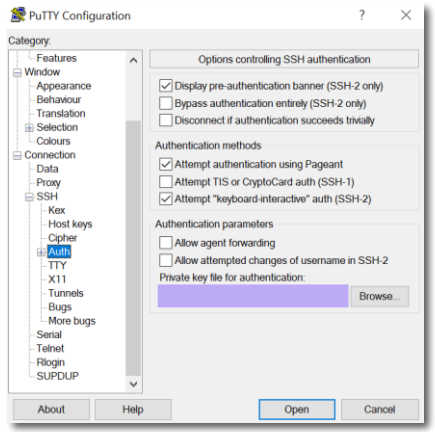
AWS – PuTTY

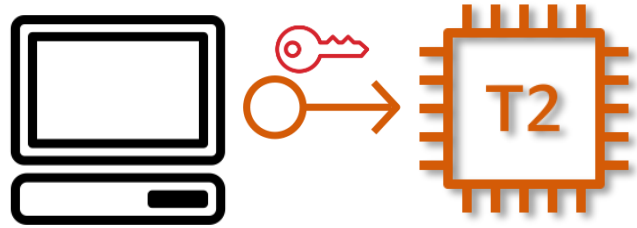
PuTTY :

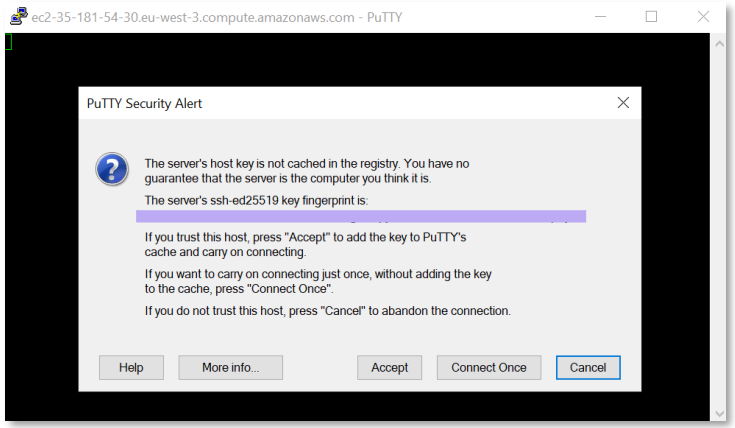
- Emulateur de terminal doublé d'un client pour les protocoles SSH, Telnet, rlogin et TCP ainsi que les liaisons RS-232
- Distribué selon les termes de la licence MIT (logiciel libre et open-source)
- Utilise une clé .ppk car le créateur de PuTTY à décidé de pas importer des foreign keys
- PuTTY Key Generator convertit la clé proportionnée par AWS EC2 du format .pem en .ppk

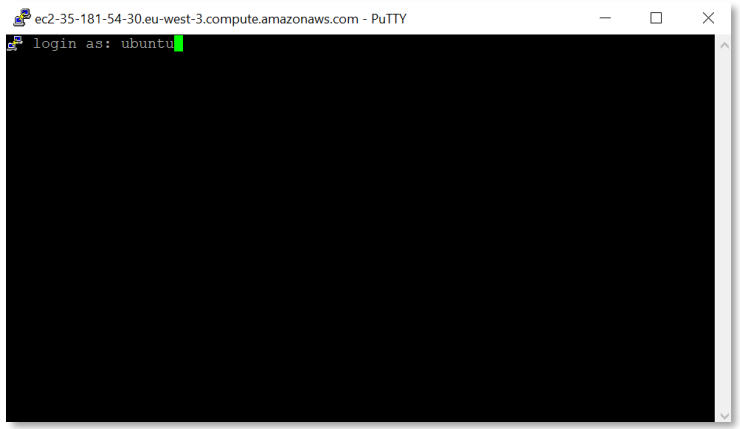


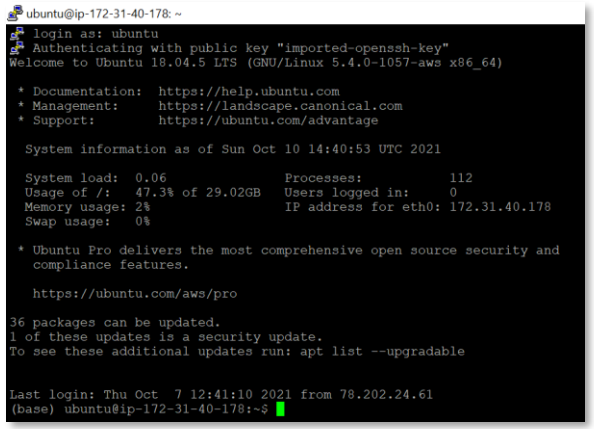












AWS – Simple Storage Service

S3 Bucket :

- Service de stockage et distribution de fichiers
- Les buckets sont comme des dossiers mais le nom doit être unique parmi tous les comptes AWS
- Capacité de static web hosting pour des pure JS or Single Page serverless applications
- Contrôle d'accès aux sous-dossiers et fichiers par utilisateur, group d'utilisateur ou globale avec IAM ou ACL
- Accès avec la librairie boto3 (AWS SDK pour Python)

Access control list (ACL)

Edit

Grant basic read/write permissions to other AWS accounts. Learn more

The console displays combined access grants for duplicate grantees
To see the full list of ACLs, use the Amazon S3 REST API, AWS CLI, or AWS SDKs.

Grantee	Objects	Bucket ACL
Bucket owner (your AWS account) Canonical ID: [redacted]	List, Write	Read, Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	-	-
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-
S3 log delivery group Group: http://acs.amazonaws.com/groups/s3/LogDelivery	-	-

```
s3 = boto3.resource('s3',
                    REGION_NAME,
                    aws_access_key_id=AWS_ACCESS_KEY_ID,
                    aws_secret_access_key=AWS_SECRET_ACCESS_KEY)
bucket = s3.Bucket(BUCKET_NAME)
```

AWS – Session Spark

Etapes :

1. Localiser Spark dans l'EC2 avec findspark
2. Communication EC2-S3 avec boto3
3. Import de la liste des fichiers dans le dataset sur S3
4. Configuration de la session Spark et le client S3A
5. Création du contexte Spark capable de communiquer avec S3
6. Spark Dataframe contenant la liste des fichiers dans le dataset choisi

1
```
import os
os.environ['SPARK_HOME'] = '/home/ubuntu/spark-3.1.2-bin-hadoop2.7/'
os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages com.amazonaws:aws-java-sdk-pom:1.10.34,org.apache.hadoop:hadoop-aws:2.7.2 pyspark-shell'

import findspark
findspark.init()
```

2 3
```
dataset = 'Training/' #Test or Training
dataset_path = 's3://' + BUCKET_NAME + '/' + FRUITS_PATH + dataset
prefix_path = FRUITS_PATH + dataset

session = boto3.session.Session(aws_access_key_id=AWS_ACCESS_KEY_ID,
                                aws_secret_access_key=AWS_SECRET_ACCESS_KEY)
s3_client = session.client(service_name='s3', region_name=REGION_NAME)
ld_set_folders = s3_client.list_objects_v2(Bucket=BUCKET_NAME,
                                           Prefix=prefix_path) #list of all the photos in the dataset
```

4 5
```
spark = (SparkSession
             .builder.master('local[*]')
             .appName('p8')
             .config('spark.hadoop.fs.s3a.access.key', AWS_ACCESS_KEY_ID)
             .config('spark.hadoop.fs.s3a.secret.key', AWS_SECRET_ACCESS_KEY)
             .config('spark.hadoop.fs.s3a.impl', 'org.apache.hadoop.fs.s3a.S3AFileSystem')
             .getOrCreate()
            )
sc = SparkContext.getOrCreate()
sc.setSystemProperty('com.amazonaws.services.s3.enableV4', 'true')
sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint", "s3.eu-west-3.amazonaws.com")
sc.setLogLevel('WARN')
```

6
```
lst_fruit = []

for photo in ld_set_folders['Contents']:
    file = photo['Key']
    lst_fruit.append(prefix_path+file)
```

```
zipped = zip(lst_fruit)
cols = ['image']
image_sdf = spark.createDataFrame(zipped, cols)
```

```
image_sdf.show(5)
```

```
+--------------------+
|               image|
+--------------------+
|data_sample/fruit...|
|data_sample/fruit...|
|data_sample/fruit...|
|data_sample/fruit...|
|data_sample/fruit...|
+--------------------+
only showing top 5 rows
```

AWS – Session Spark

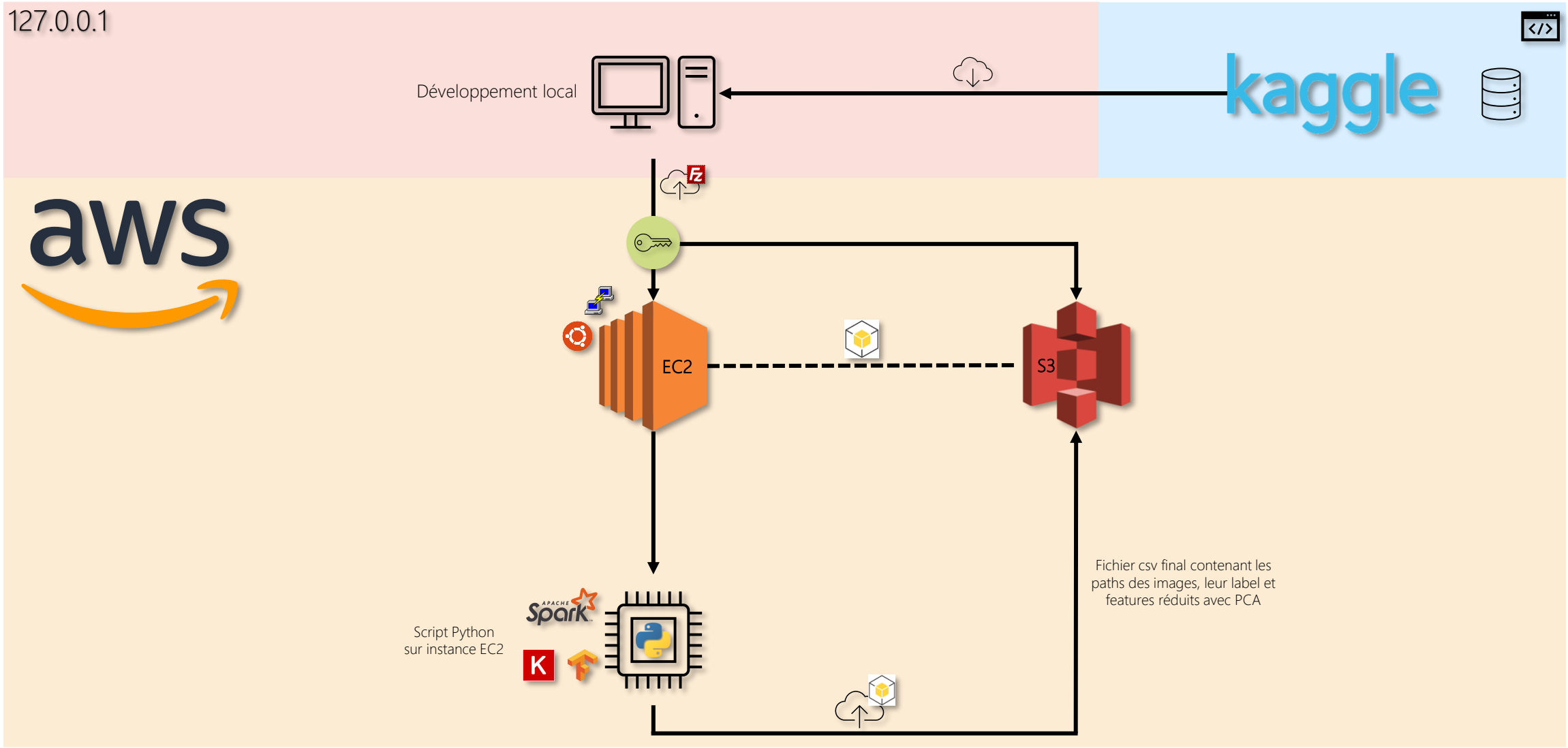
Etapes :

7. Importer une image comme test de fonctionnement

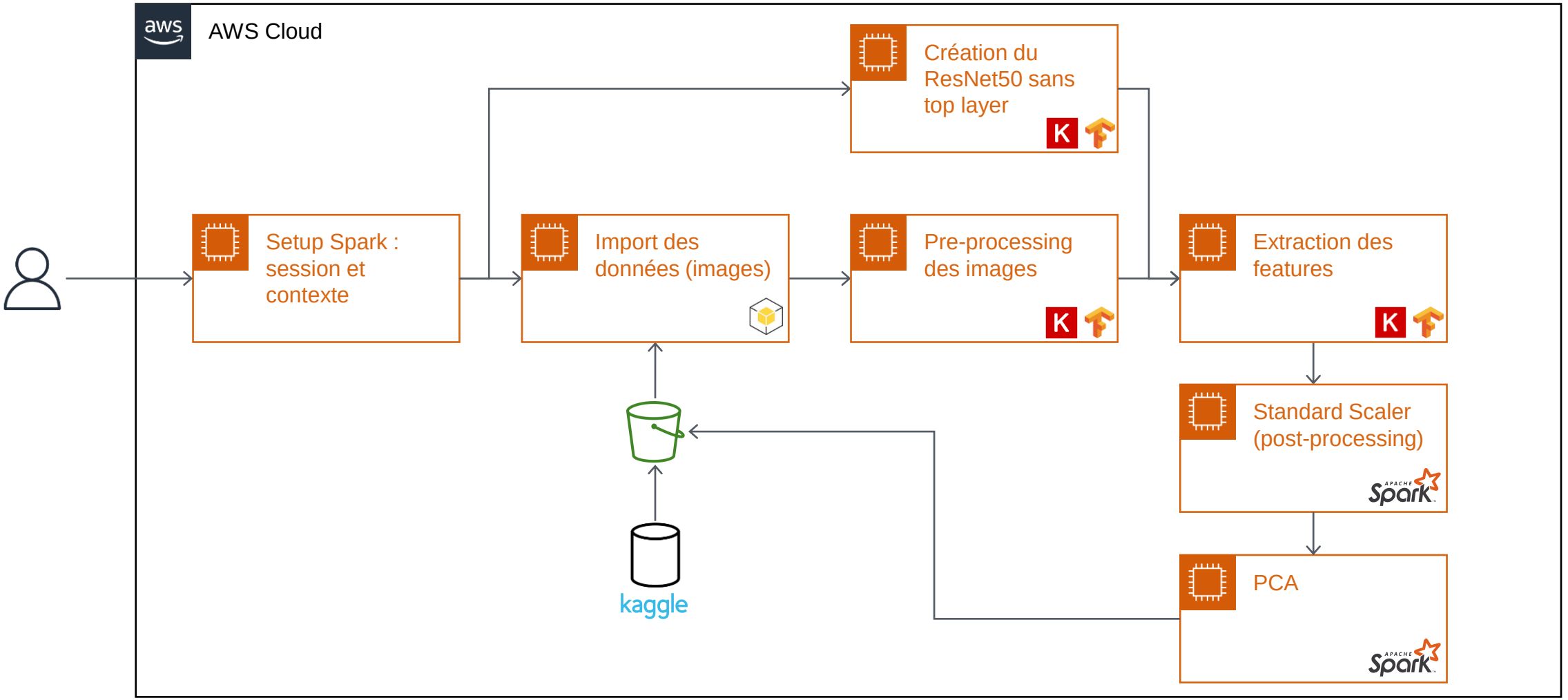
```
photo_test = 'data_sample/fruits_360/Training/Apple Braeburn/0_100.jpg'

response = s3_client.generate_presigned_url('get_object',
                                            Params={'Bucket': BUCKET_NAME,
                                                    'Key': photo_test},
                                            ExpiresIn=604800)
resp = urllib.request.urlopen(response)
# img_bytes = Image.open(io.BytesIO(resp.read())).convert('L').resize((224, 224)) #for grayscale
img_bytes = Image.open(io.BytesIO(resp.read())).resize((224, 224)) #rgb colors
```

```
img_bytes
```

AWS – Environnement



Traitement des images – Processus



Traitement des images – Pre, Post Processing et PCA

Extraction des features :

- Load ResNet50
- Re-shape des inputs : 100x100 px vers 224x224
- Expansion d'une dimension
- Prédiction des features

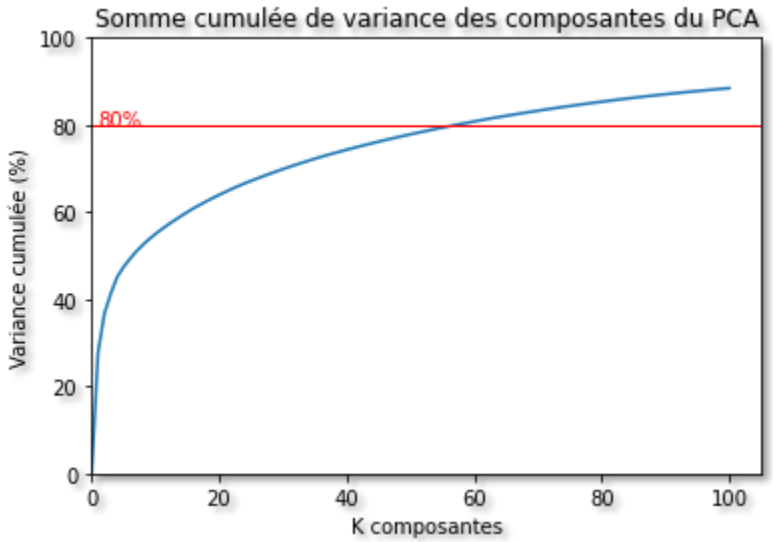
```
s3 = boto3.resource('s3',
                    REGION_NAME,
                    aws_access_key_id=AWS_ACCESS_KEY_ID,
                    aws_secret_access_key=AWS_SECRET_ACCESS_KEY)
bucket = s3.Bucket(BUCKET_NAME)

resnet_features=[]

for photo in ld_set_folders['Contents']:
    file = photo['Key']
    obj = bucket.Object(file)
    img_body = obj.get()['Body']

    img = Image.open(img_body).resize((224, 224))
    img_arr = np.expand_dims(img, axis=0)
    img_pss = preprocess_input(img_arr)
    resnet_feature = model.predict(img_pss).ravel().tolist()
    resnet_features.append(resnet_feature)
```

```
#https://stackoverflow.com/questions/48164206/pyspark-adding-a-column-from-a-list-of-values-using-a-udf
temp_sdf = spark.createDataFrame([(f,) for f in resnet_features], ['features'])
image_sdf = image_sdf.withColumn('row_index', row_number().over(Window.orderBy(monotonically_increasing_id())))
temp_sdf = temp_sdf.withColumn('row_index', row_number().over(Window.orderBy(monotonically_increasing_id())))

image_sdf = image_sdf.join(temp_sdf, image_sdf.row_index == temp_sdf.row_index).drop('row_index')
```

Formater les features pre PCA :

- Conversion de la liste contenant les features de chaque image en Spark Dataframe
- Vector Dense pour assurer la compatibilité des données avec MLlib
- Standar Scaler pour standardiser les features en supprimant la moyenne et en mettant à l'échelle la variance unitaire à l'aide des statistiques sur l'échantillon d'entraînement

```
standardizer = StandardScaler(withMean=True, withStd=True,
                              inputCol='features_vec',
                              outputCol='features_scaled')
std = standardizer.fit(sdf_)
sdf_ = std.transform(sdf_)
sdf_ = sdf_.select('image', 'label', 'features_scaled')
```




```
dim_pca = 100
pca = PCA(k=dim_pca,
          inputCol='features_scaled',
          outputCol='features_pca',
         )

pca_model = pca.fit(sdf_)
pca_sdf = pca_model.transform(sdf_)

pca_sdf = pca_sdf.select('image', 'label', 'features_pca')
```

Conclusions – Aller plus loin

Big Data solution dans le cloud :

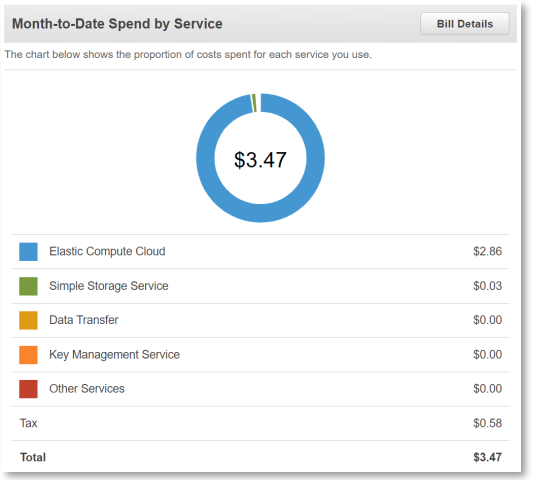
- Mise en place d'une instance EC2 et d'un Bucket S3
- Gestion des droits sur S3
- Administration d'un serveur Linux (Ubuntu Server) par SSH
- Configuration de session et contexte Spark

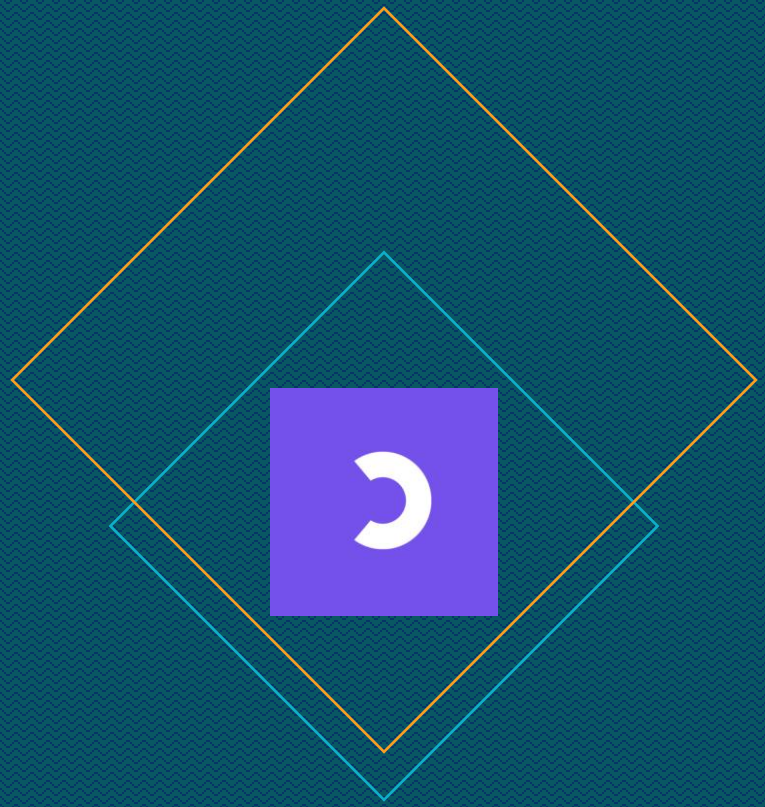
Difficultés :

- Mise en place d'un environnement Spark fonctionnel
- Choix complexes, nombreuses combinaisons techniques possibles
- Debugging compliqué dû à des erreurs explicites (superposition Spark/Java)

Amélioration :

- Passer à l'échelle, augmentation du nombre d'instances esclaves (nœuds) sans coupure
- Déployer le modèle en prod
- Surveillance des tâches via SparkUI sur l'instance EC2 avec AWS Glue



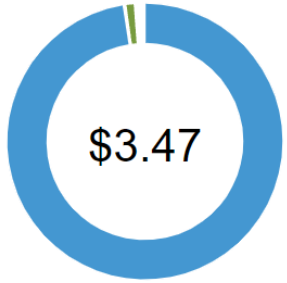


Annexe – Budget

Month-to-Date Spend by Service

Bill Details

The chart below shows the proportion of costs spent for each service you use.



Elastic Compute Cloud	$2.86
Simple Storage Service	$0.03
Data Transfer	$0.00
Key Management Service	$0.00
Other Services	$0.00
Tax	$0.58
Total	**$3.47**

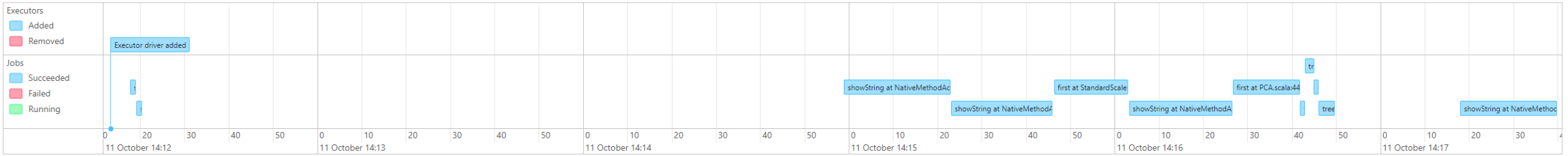
AWS Service Charges		**$3.47**
▸ Data Transfer		$0.00
▾ Elastic Compute Cloud		$2.86
▾ **EU (Paris)**		**$2.86**
Amazon Elastic Compute Cloud running Linux/UNIX		$1.73
$0.1056 per On Demand Linux t2.large Instance Hour	16.386 Hrs	$1.73
EBS		$1.13
$0.116 per GB-month of General Purpose SSD (gp2) provisioned storage	9.770 GB-Mo	$1.13
▾ Key Management Service		$0.00
▾ **EU (Paris)**		**$0.00**
AWS Key Management Service eu-west-3-KMS-Requests		$0.00
$0.00 per request - Monthly Global Free Tier for KMS requests	13.000 Requests	$0.00
▸ Secrets Manager		$0.00
▸ Simple Queue Service		$0.00
▾ Simple Storage Service		$0.03
▾ **EU (Paris)**		**$0.03**
Amazon Simple Storage Service EUW3-Requests-Tier1		$0.01
$0.0053 per 1,000 PUT, COPY, POST, or LIST requests	2,091.000 Requests	$0.01
Amazon Simple Storage Service EUW3-Requests-Tier2		$0.01
$0.0042 per 10,000 GET and all other requests	12,483.000 Requests	$0.01
Amazon Simple Storage Service EUW3-TimedStorage-ByteHrs		$0.01
$0.024 per GB - first 50 TB / month of storage used	0.336 GB-Mo	$0.01
▸ **US East (N. Virginia)**		**$0.00**
Taxes		
VAT to be collected		$0.58

Annexe – Spark UI Local

Spark Jobs (?)

User: pipoa
Total Uptime: 13 min
Scheduling Mode: FIFO
Completed Jobs: 12

Event Timeline
Enable zooming



Completed Jobs (12)

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2021/10/11 14:17:17	22 s	3/3	25/25
10	treeAggregate at RowMatrix.scala:156 treeAggregate at RowMatrix.scala:156	2021/10/11 14:16:45	4 s	1/1 (2 skipped)	1/1 (24 skipped)
9	first at RowMatrix.scala:442 first at RowMatrix.scala:442	2021/10/11 14:16:44	1 s	1/1 (2 skipped)	1/1 (24 skipped)
8	treeAggregate at Statistics.scala:58 treeAggregate at Statistics.scala:58	2021/10/11 14:16:42	2 s	1/1 (2 skipped)	1/1 (24 skipped)
7	first at RowMatrix.scala:62 first at RowMatrix.scala:62	2021/10/11 14:16:41	1 s	1/1 (2 skipped)	1/1 (24 skipped)
6	first at PCA.scala:44 first at PCA.scala:44	2021/10/11 14:16:26	15 s	3/3	25/25
5	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2021/10/11 14:16:03	23 s	3/3	25/25
4	first at StandardScaler.scala:113 first at StandardScaler.scala:113	2021/10/11 14:15:46	17 s	4/4	26/26
3	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2021/10/11 14:15:23	23 s	3/3	25/25
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2021/10/11 14:14:58	24 s	3/3	25/25
1	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2021/10/11 14:12:19	1 s	1/1	1/1
0	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2021/10/11 14:12:17	1 s	1/1	1/1

Annexe – Big Data I

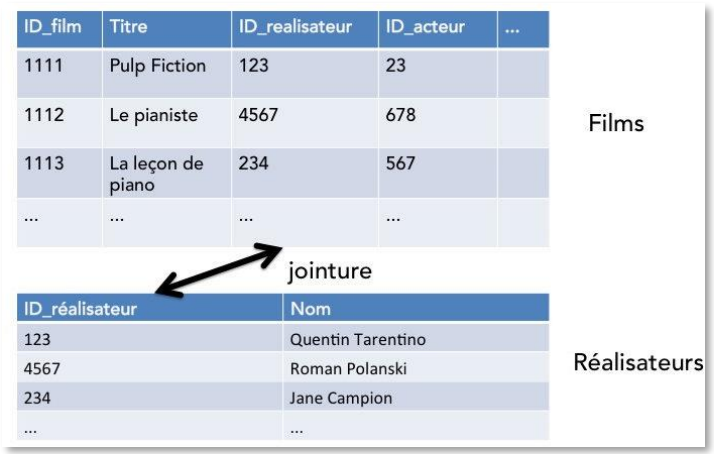
Les quatre « V » :

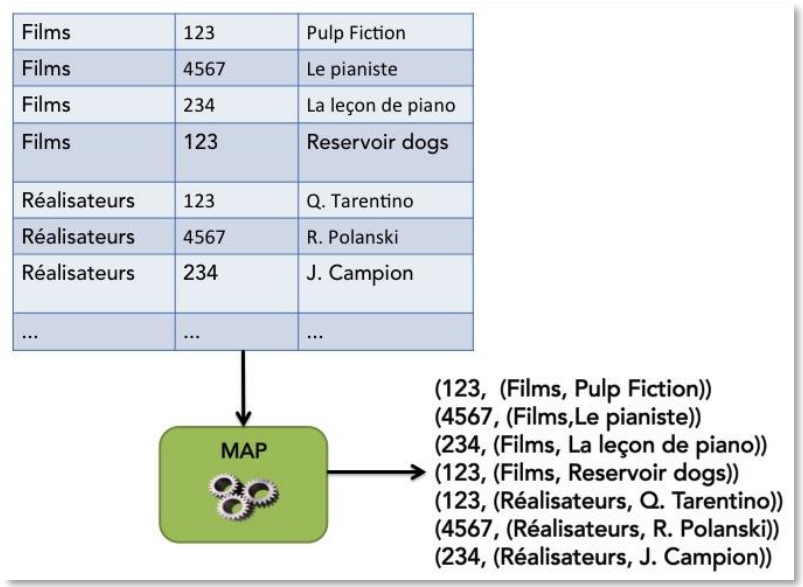
- Volume : quantité de données générées et stockées
- Vitesse : à laquelle les données sont générées et traitées
- Variété : type et nature des données, lié à la diversification des usages du numérique
- Véracité : ou fiabilité, qui fait référence à la qualité et à la valeur des données

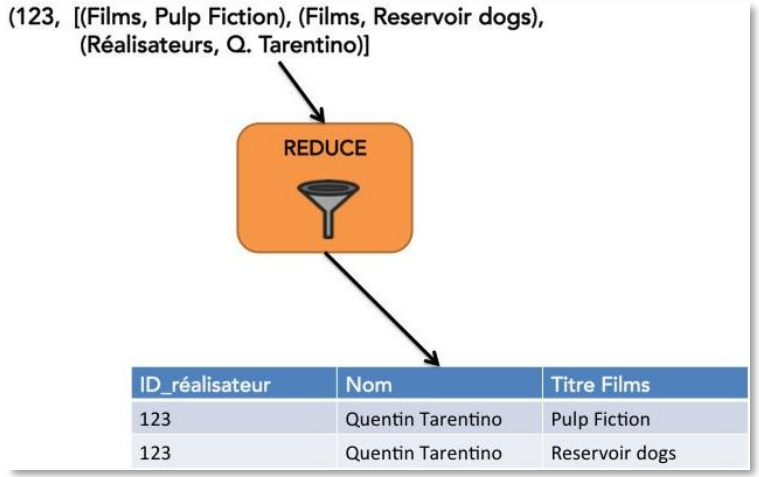
Comment traiter les mégadonnées?

Traitement par calculs distribués (MapReduce) divisant les opérations en micro-opérations distribuables entre différentes machines, réalisables en parallèle et agrégeant les résultats sur une même machine

Source : https://openclassrooms.com/en/courses/4297166-realisez-des-calculs-distribues-sur-des-donnees-massives/4308631-parcourez-les-principaux-algorithmes-mapreduce



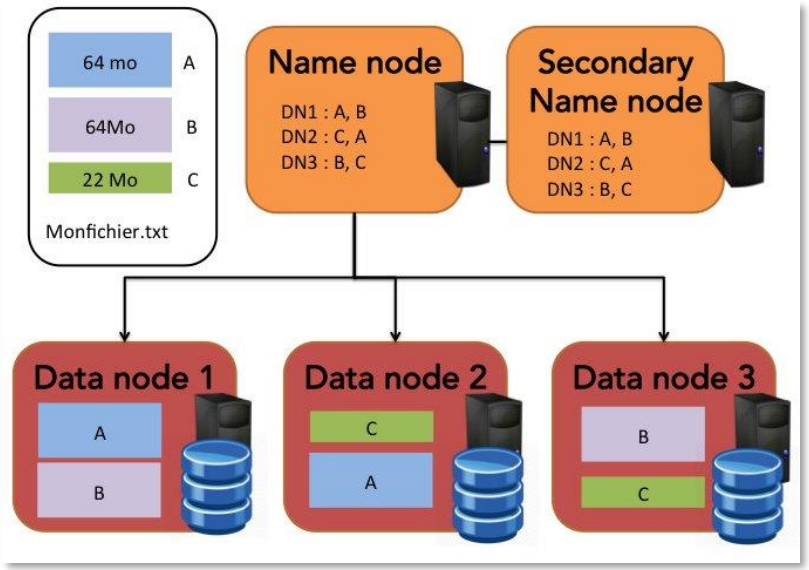


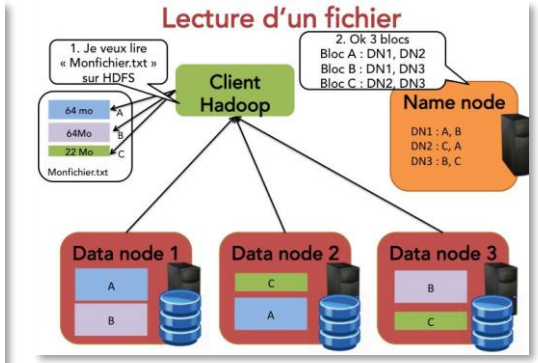


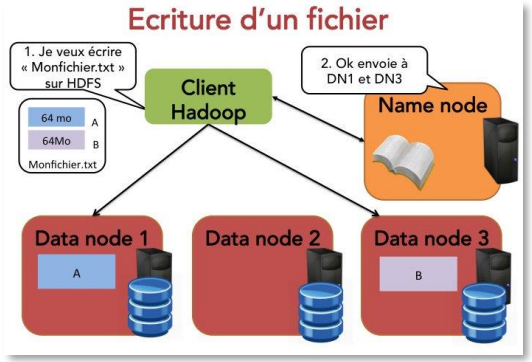
Annexe – Big Data II

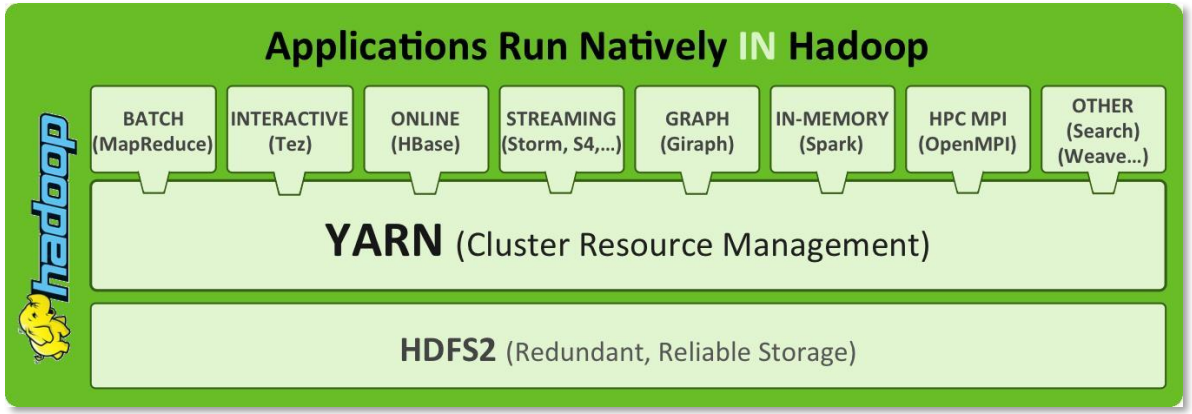
Système de fichiers HDFS :

- Hadoop Distributed File System est un système de fichiers distribué et la couche native de stockage et d'accès à des données d'Hadoop
- Les fichiers sont découpés en blocs d'octets de grande taille (64 Mo) pour optimiser les temps
- Ces blocs sont répartis sur plusieurs machines, permettant le traitement du même fichier en parallèle
- Les blocs sont répliqués sur plusieurs machines pour garantir la tolérance aux pannes









Source : https://openclassrooms.com/en/courses/4297166-realisez-des-calculs-distribues-sur-des-donnees-massives/4308656-familiarisez-vous-avec-hadoop

Annexe – Spark

Hadoop :

- Logiciel open-source qui permet gérer des ensembles de données volumineux (giga-octets ou plus) en permettant à un réseau de nœuds (ordinateurs) de résoudre des problèmes
- C'est une solution scalable et rentable qui stocke et traite des données structurées, semi et non structurées (logs, capteurs IoT, clickstream records, etc)
- Les principaux avantages d'Hadoop sont la protection des données en cas de panne de hardware, sa scalabilité d'un seul nœud à des réseaux de serveurs et la capacité d'analyse en temps réel pour la prise de décision

Spark :

- Moteur de traitement de données open-source pour les grands ensembles de données
- Comme Hadoop, Spark distribue les tâches volumineuses sur différents nœuds. Cependant, il performe plus rapidement qu'Hadoop car il utilise la RAM pour caching et traitement des données
- Le framework Spark inclut un moteur qui prend en charge des requêtes SQL, data streaming, machine learning et traitement des graphes

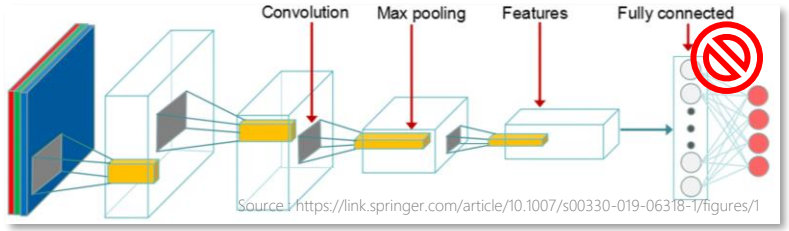
Comparaison :

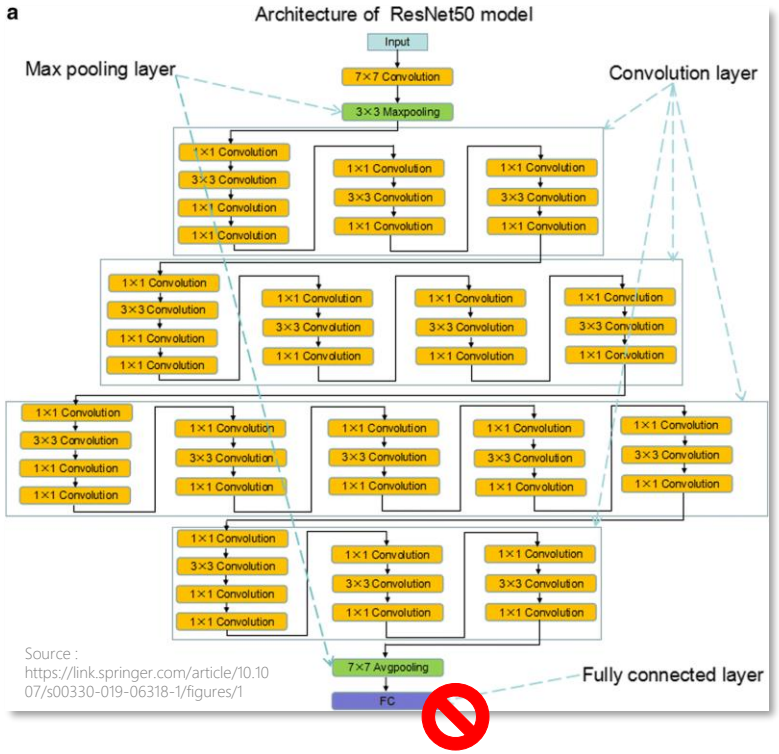
- Spark utilise RAM. Par contre, Hadoop stocke les données sur plusieurs sources (n'importe quel type de stockage) et les traite par lots via MapReduce
- Spark coûte plus cher car il repose sur des calculs en mémoire pour le traitement des données en temps réel, ce qui l'oblige à utiliser de grandes quantités de RAM pour faire tourner les nœuds
- Spark inclut MLlib qui effectue des calculs ML itératifs en mémoire et des outils qui effectuent des régressions, classifications, construction du pipelines, évaluation, etc.

Annexe – ResNet50

Residual Network :

- Modèle pré-entraîné utilisé pour la classification des images
- 50 car il a 50 couches de profondeur
- L'option include_top=False permet l'extraction des features car elle enlève la dernière couche du network
- Aussi, enlever la dernière couche permet d'utiliser des images sans taille fixe et d'éviter l'erreur de taille de matrice sur la forme des inputs
- Le 3 sur la forme d'input signifie que nous allons utiliser des images en RGB






```
model = ResNet50(
    include_top=False,
    weights=None,
    pooling='max',
    input_shape=(224, 224, 3),
)
model.summary()
```

```
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0          conv2_block2_1_bn[0][0]
__________________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928      conv2_block2_1_relu[0][0]
__________________________________________________________________________________________
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256        conv2_block2_2_conv[0][0]
__________________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0          conv2_block2_2_bn[0][0]
__________________________________________________________________________________________
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640      conv2_block2_2_relu[0][0]
__________________________________________________________________________________________
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024       conv2_block2_3_conv[0][0]
```