



PROJET 8 - PARCOURS DATA SCIENTIST :

DÉPLOYEZ UN MODÈLE DANS LE CLOUD

LAURA DAINES

MENTOR :

BENJAMIN TARDY

SOMMAIRE



Rappel de la problématique et présentation du jeu de données



Environnement Big Data dans le cloud



Chaîne de traitement des images



Conclusion et recommandations



Questions - Réponses

RAPPEL DE LA PROBLÉMATIQUE ET PRÉSENTATION DU JEU DE DONNÉES

RAPPEL DE LA PROBLÉMATIQUE - CONTEXTE

« Fruits! », jeune start-up de l'AgriTech, cherche à :

- Développer une application mobile où les utilisateurs prennent en photo un fruit, et obtiennent des informations sur ce fruit
- Préserver la biodiversité des fruits, en proposant des solutions innovantes pour la récolte des fruits, comme des robots cueilleurs intelligents permettant des traitements spécifiques pour chaque espèce de fruits



Fruits!

RAPPEL DE LA PROBLÉMATIQUE - MISSION

Dans un environnement Big Data sur AWS :

- développer une première chaîne de traitement des données qui comprendra le preprocessing et une étape de réduction de dimensions
- mettre en place les briques de traitement et le stockage des données qui serviront lorsqu'il faudra passer à l'échelle en termes de volume de données

JEU DE DONNÉES

Jeu de données constitué d'images de fruits, disponible sur Kaggle : <https://www.kaggle.com/moltean/fruits>

- Nombre total d'images labellisées (un fruit ou légume par image) : 90483
 - Taille de l'ensemble d'entraînement : 67692 images
 - Taille de l'ensemble de test : 22688 images
- Nombre de classes : 131 espèces différentes (fruits et légumes)
- Taille des images : 100 x 100 pixels

ENVIRONNEMENT BIG DATA DANS LE CLOUD

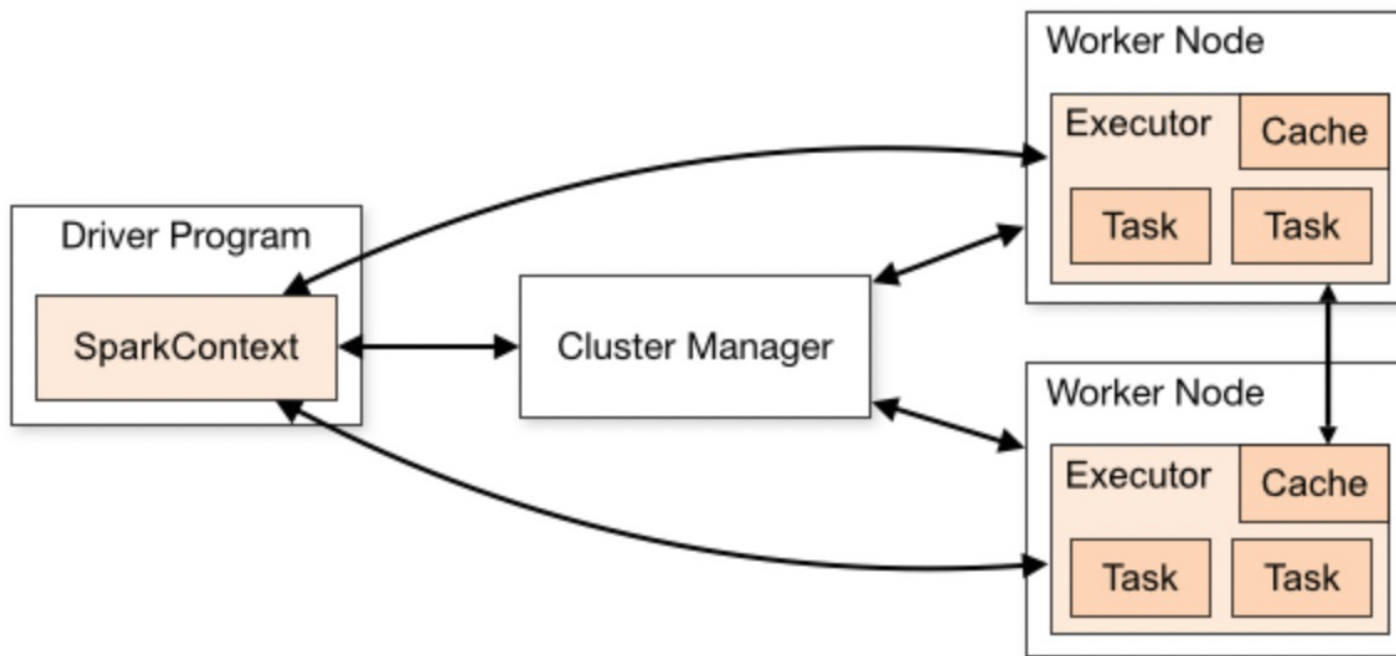
BIG DATA - SPARK

- Le Big Data, ce sont des données massives à forte vélocité, non structurées et hétérogènes, qui sont telles que les solutions classiques de stockage, de gestion et de traitement, ne suffisent plus.
- Comment traiter ces données?
 - Calcul distribué avec Spark
- Apache Spark est un moteur de traitement de données, dédié au Big Data. Il permet d'effectuer des analyses et traitements de larges volumes de données de manière distribuée. Ainsi, Spark coordonne l'exécution de tâches sur des données en les répartissant au sein d'un cluster de machines (cluster computing).
- Spark permet de prototyper des applications en local avant de les envoyer vers un cluster de plusieurs machines pour traiter des données de taille plus conséquente, sans se préoccuper du changement d'architecture.



ARCHITECTURE SPARK

Driver Program : accessible par un point d'entrée appelé SparkSession. Chargé de répartir les tâches sur les différents executors. C'est le driver qui exécute la méthode main() de l'application Spark et qui crée le SparkContext



Un ou plusieurs workers : chaque worker instancie un executor, chargé d'exécuter le code qui lui est assigné par le driver et lui rapporter l'état d'avancement de la tâche.

SparkContext est capable de se connecter à plusieurs types de Cluster Manager comme Mesos, Yarn, Kubernetes ou le cluster manager autonome de Spark.

Cluster Manager : chargé d'instancier les différents workers. Responsable de l'allocation des ressources à travers l'application Spark (programme utilisateur construit sur Spark)

SPARK

L'écosystème des API de Spark est hiérarchisé :

- les APIs bas-niveau, avec les RDDs (Resilient Distributed Dataset)
 - RDD : il s'agit de l'abstraction de données la plus basique dans Spark. RDD est une collection distribuée immuable d'éléments de vos données, partitionnée sur les nœuds de votre cluster qui peut être exploitée en parallèle avec une API de bas niveau qui offre des transformations et actions.
- les APIs de haut niveau, avec les Datasets, DataFrames et SQL
 - Dataframe : collection distribuée de données organisées en colonnes nommées. Conceptuellement équivalent à une table dans une base de données relationnelle ou à un R/Python Dataframe.
 - Datasets : L'API Dataset est une extension de DataFrames qui fournit une interface de programmation orientée objet et sécurisée. Il s'agit d'une collection d'objets fortement typés et immuables qui sont mappés à un schéma relationnel.
- les autres bibliothèques (Structured Streaming, Advanced Analytics, etc.)

AWS – AMAZON WEB SERVICES

- Amazon Web Services (AWS) est une plate-forme de services cloud offrant plus de 200 services complets provenant de centres de données du monde entier. C'est un service qui permet d'utiliser les mécanismes du cloud computing.
- Le cloud computing est la fourniture à la demande de puissance, de stockage de base de données, d'applications et d'autres ressources informatiques.



AWS – IAM IDENTITY AND ACCESS MANAGEMENT

AWS IAM est un service qui offre des mécanismes de contrôle d'accès sécurisés pour tous les services AWS.

IAM dans le cadre du projet :

- Création d'un user
- Configuration des permissions et des rôles
 - Rôle IAM est une identité IAM qui dispose d'autorisations spécifiques. Au lieu d'être uniquement associé à une personne, un rôle est destiné à être assumé par toute personne qui en a besoin.



AWS – EC2 ELASTIC COMPUTE CLOUD

AWS EC2 est un service qui permet de gérer des serveurs sous forme de machines virtuelles dans le cloud. Il compose l'offre de IaaS (Infrastructure as a Service).



EC2 dans le cadre du projet :

- Choix de l'instance
 - OS: Ubuntu Server 18.04, 64-bit / t2.large / 8 GiB RAM / EBS Storage 8 GiB

	Family ▾	Type ▾	vCPUs ⓘ ▾	Memory (GiB) ▾	Instance Storage (GB) ⓘ ▾
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only
<input type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only
<input type="checkbox"/>	t2	t2.small	1	2	EBS only
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only
<input checked="" type="checkbox"/>	t2	t2.large	2	8	EBS only

AWS – EC2 ELASTIC COMPUTE CLOUD

- Groupe de sécurité pour SSH : Protocol TCP, Port range 22
- Paire de clés pour SSH : Public Key / Private Key
- Utilisation d'une Elastic IP
- Attribution d'un rôle IAM pour notre instance EC2


Instance ID

 i-09b8246f56b2e4248 (P8_inst_try6)

IPv6 address

–


Private IPv4 DNS

 ip-172-31-31-114.ec2.internal

VPC ID

 vpc-0ec7c6c989d30bea0 [🔗](#)

Public IPv4 address

 52.45.94.141 (My first Elastic IP) | [open address](#) [🔗](#)

Instance state

⏹ Stopped


Instance type

t2.large


AWS Compute Optimizer finding

ⓘ Opt-in to AWS Compute Optimizer for recommendations. | [Learn more](#) [🔗](#)

Private IPv4 addresses

 172.31.31.114

Public IPv4 DNS

 ec2-52-45-94-141.compute-1.amazonaws.com | [open address](#) [🔗](#)

Elastic IP addresses

 52.45.94.141 (My first Elastic IP) [Public IP]

IAM Role

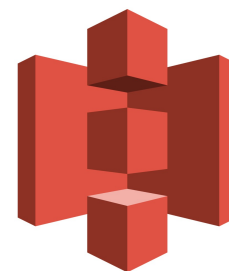
 PolicyforAmesp8 [🔗](#)

AWS – S3 SIMPLE STORAGE SERVICE

Amazon S3 est un service de stockage et de distribution de fichiers. S3 propose de stocker des données dans des buckets. A l'intérieur de chaque bucket, il est possible de déposer des fichiers (on parle d'objets).

S3 dans le cadre du projet :

- Création d'un bucket
- Upload des données dans notre bucket
- Gestion de l'accès aux objets stockés dans le bucket avec une Bucket Policy, écrite en JSON

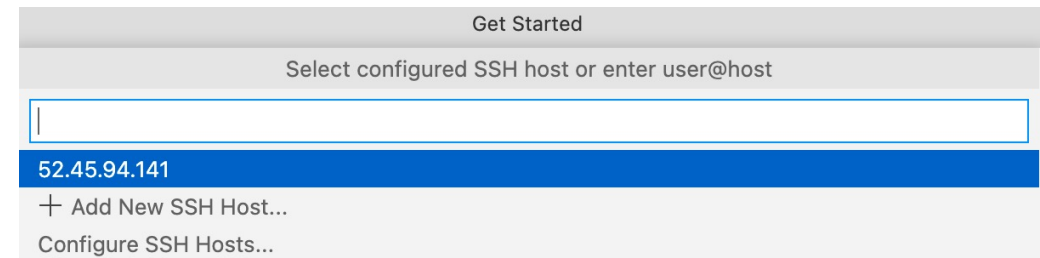
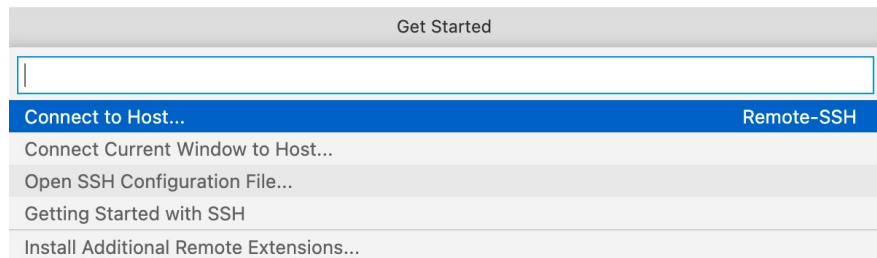


VISUAL STUDIO CODE - ACCÈS SSH

Pour accéder au serveur EC2 via SSH sur Visual Studio Code :

- Fichier de configuration
- Key Pair : fichier .pem

```
config x
Users > la > .ssh > config
1 Host 52.45.94.141
2   HostName 52.45.94.141
3   IdentityFile /Users/la/.ssh/Proj8InstanceEnvTry2Key.pem
4   User ubuntu
5
6 Host 52.45.94.141
7   HostName 52.45.94.141
8   User ubuntu
9
10
```



RÉALISATION DE LA CHAÎNE DE TRAITEMENT DES IMAGES

ENVIRONNEMENT SPARK

- findspark : chargée de localiser la bibliothèque PySpark installée avec Apache Spark
- SparkSession : création d'une spark session et configuration du module hadoop-s3
- SparkContext : création du contexte Spark capable de communiquer avec S3

```
import os

# setting the environment variable PYSPARK_SUBMIT_ARGS :
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:
```

```
import findspark
findspark.init()
```

```
spark = (SparkSession
        .builder.master('local[*]')
        .appName('p8')
        .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.us-east-1.amazonaws.com')
sc.setLogLevel('WARN')
sc
```

SparkContext

Spark UI

Version	v3.1.2
Master	local[*]
AppName	p8

CONNEXION À S3

- Boto3 : ensemble de fonctions Python spécifiques pour interagir avec les services AWS Amazon
- Connexion à S3 et communication entre S3 et EC2
- Import de la liste des fichiers dans le bucket sur S3
 - Les sorties sont renvoyées à l'aide de dictionnaires Python.
 - Nous devons parcourir ces dictionnaires nous-mêmes

```
connexion = boto3.client('s3')  
connexion
```

```
<botocore.client.S3 at 0x7f0194f92130>
```

```
# use connexion.list_objects to get the objects in our desired bucket:  
contents = connexion.list_objects(Bucket='fruit-folders-bucket')['Contents']  
contents[0]
```

```
{'Key': 'my_folder_fruits/Apple Braeburn/109_100.jpg',  
 'LastModified': datetime.datetime(2021, 10, 31, 12, 11, 11, tzinfo=tzlocal()),  
 'ETag': '"562fb71e112bec530ac9d6b9b2b4393d"',  
 'Size': 4750,  
 'StorageClass': 'STANDARD',  
 'Owner': {'DisplayName': 'lauraamydaines+aws',  
           'ID': 'f33c3d09fb3f11b7378e2e5fb876ee2573413479ccf00132c0dd0510993389ea'}}
```

PREPROCESSING – EXTRACTION DE FEATURES AVEC RESNET 50

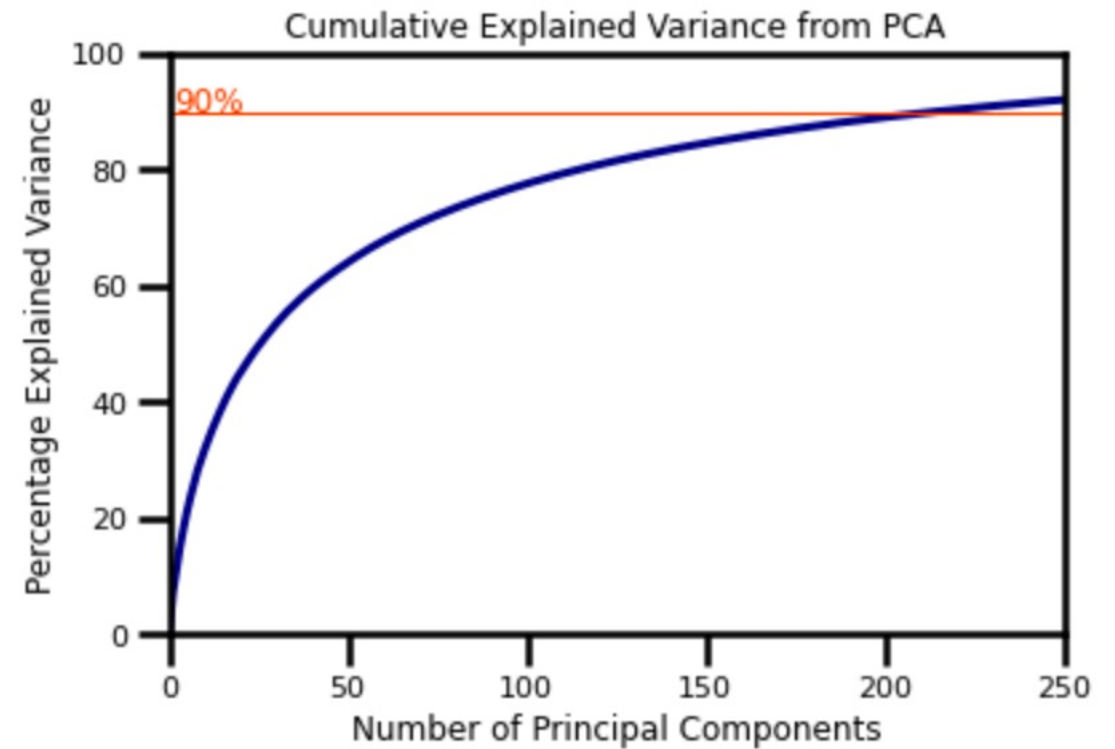
- Resize des images 100 x 100 px à 224 x 224 px et application de la fonction `preprocessing_input` importé de `tensorflow.keras.applications.resnet50`
- Transfer Learning avec ResNet50 pour extraction de features, utilisation des poids « imagenet » et suppression de la dernière couche, pour obtenir 2048 features pour chaque image

```
avg_pool (GlobalAveragePooling2 (None, 2048))      0      conv5_block3_out[0][0]
=====
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
```

- Transformation de la liste des features extraites du ResNet50 en vecteur dense

RÉDUCTION DE DIMENSIONS

- Standard Scaler() sur le vecteur dense des features obtenues du ResNet50
- Réduction de dimensions avec ACP (200 composantes gardées pour le output final)
- Sauvegarde de l'output de l'ACP sous forme d'un fichier csv sur S3



CONCLUSION ET RECOMMANDATIONS

CONCLUSION ET RECOMMANDATIONS

- Pour aller plus loin :
 - Amazon EMR (Elastic Map Reduce), pour créer un cluster EMR qui permettra la distribution des calculs sur différentes instances EC2 au lieu d'une seule instance EC2
 - Parquet au lieu de csv pour le output envoyé au bucket S3
- Pour la classification :
 - Entraînement du modèle ResNet50 ou autre CNN sur nos images
 - Choix de différents modèles de classification et du nombre de composantes ACP

QUESTIONS - RÉPONSES

