



OPENCLASSROOMS

Data Scientist

Projet N°8:

Déployer un modèle dans le cloud



Fruits!

Elaborée par: Mariem Kchaou

Elaboré par: M. Abdou Karim Kandji

Février 2023



**Problématique et
présentation du jeu
de données**



**Rappels sur la
notion de Big Data**



**Architecture retenue et
chaîne de traitement**



Conclusion





1



**Problématique et
présentation du jeu
de données:**

Problématique



Fruits !

Souhaite proposer des solutions innovantes pour la récolte des fruits.
Développer des robots cueilleurs intelligent à l'aide d'une application qui permettra aux utilisateurs de prendre en photo un fruit et d'obtenir des information sur ce fruit.



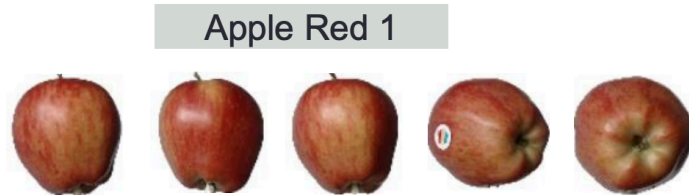
Mission

Développer un environnement Big Data.
Réaliser une première chaîne de traitement des données avec le préprocessing et une étape de réduction de dimension

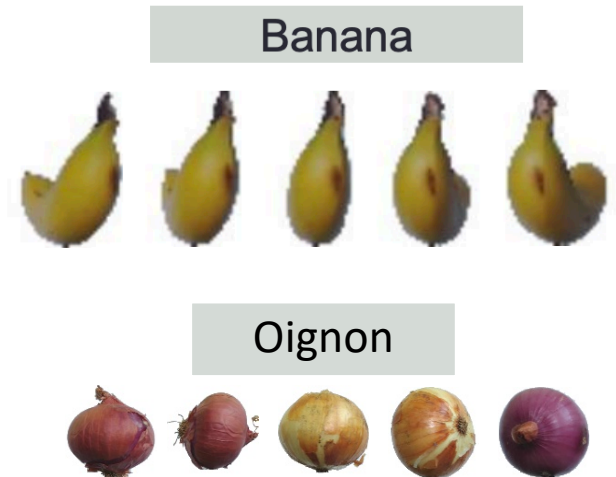
Problématique et présentation du jeu de données:

Jeux de données

- Origine:
 - Images de fruits et légumes et des labels associés ([Fruits 360](#), Mihai Oltean)
 - 131 variétés de fruits et légumes différents (un dossier par variété)
 - Plusieurs variétés du même fruit (exemple : pomme « red » et « golden »)
 - Plusieurs variétés du même légume (exemple : oignon « red » et « white »)
- Caractéristiques :
 - Images 100x100 JPEG RGB
 - Photos studio sur fond blanc de fruits centrée sur le fruit
 - Photos sous tous les angles (timelapse + rotation 3 axes)



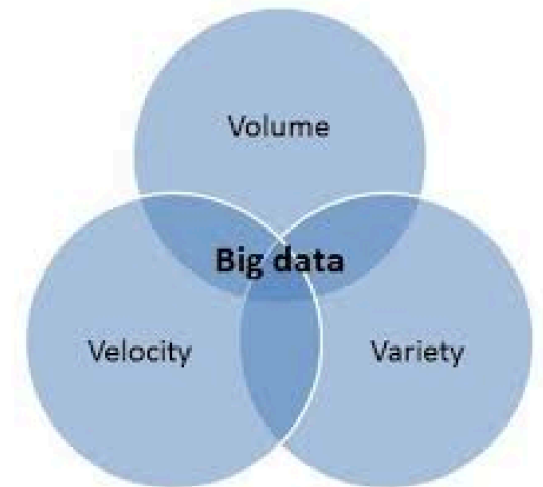
kaggle





Qu'est-ce que le Big Data?

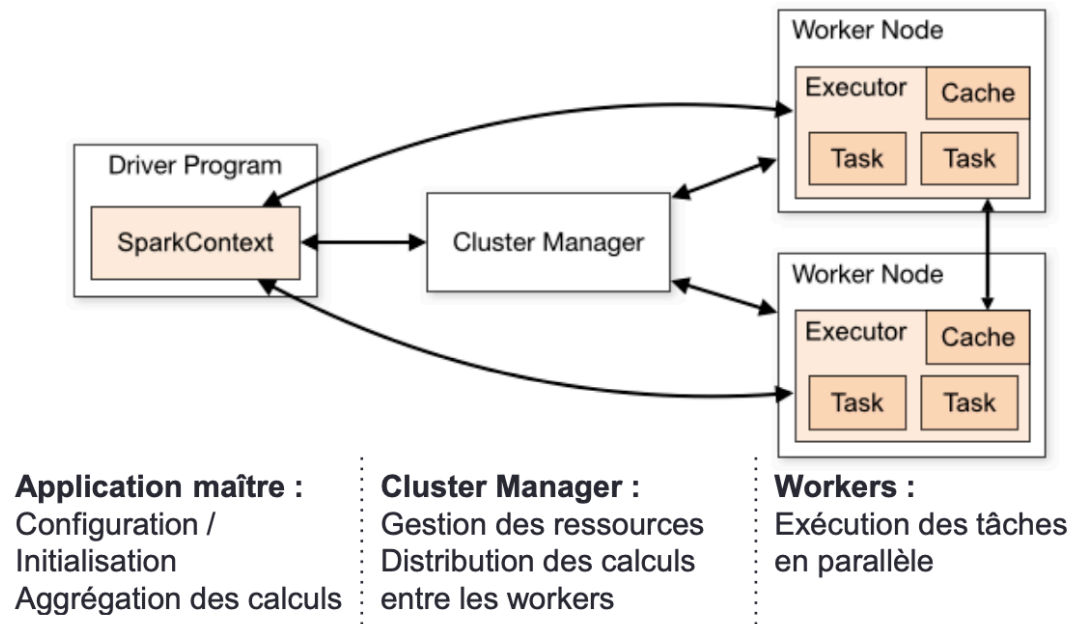
- En Français : les données massives
- Les enjeux en « V » :
 - **Volume** : trop important pour être stocké et/ou traité sur une seule machine avec des performances acceptables.
 - Dépassement de la capacité de RAM
 - Dépassement des capacités de stockage
 - Etc.
 - **Vitesse** à laquelle les données sont produites
 - Large **Variété** de types de données
 - Etc.



Comment répondre à ces enjeux?

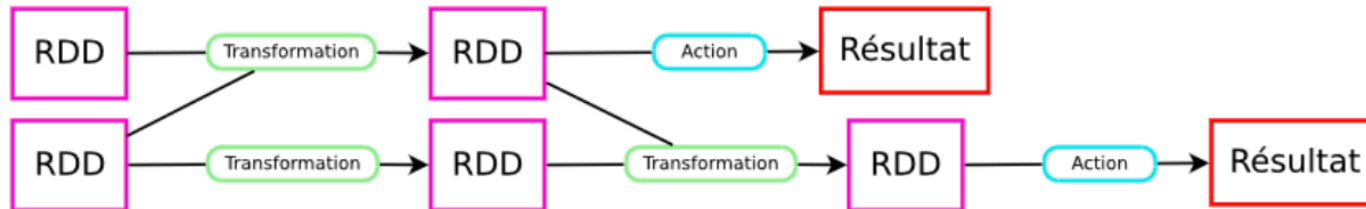
Capacités de calcul : Traitement par calculs distribués (MapReduce)

- Diviser les opérations en micro opérations distribuables entre différentes machines, réalisables en parallèle
- Agréger les résultats sur une même machine

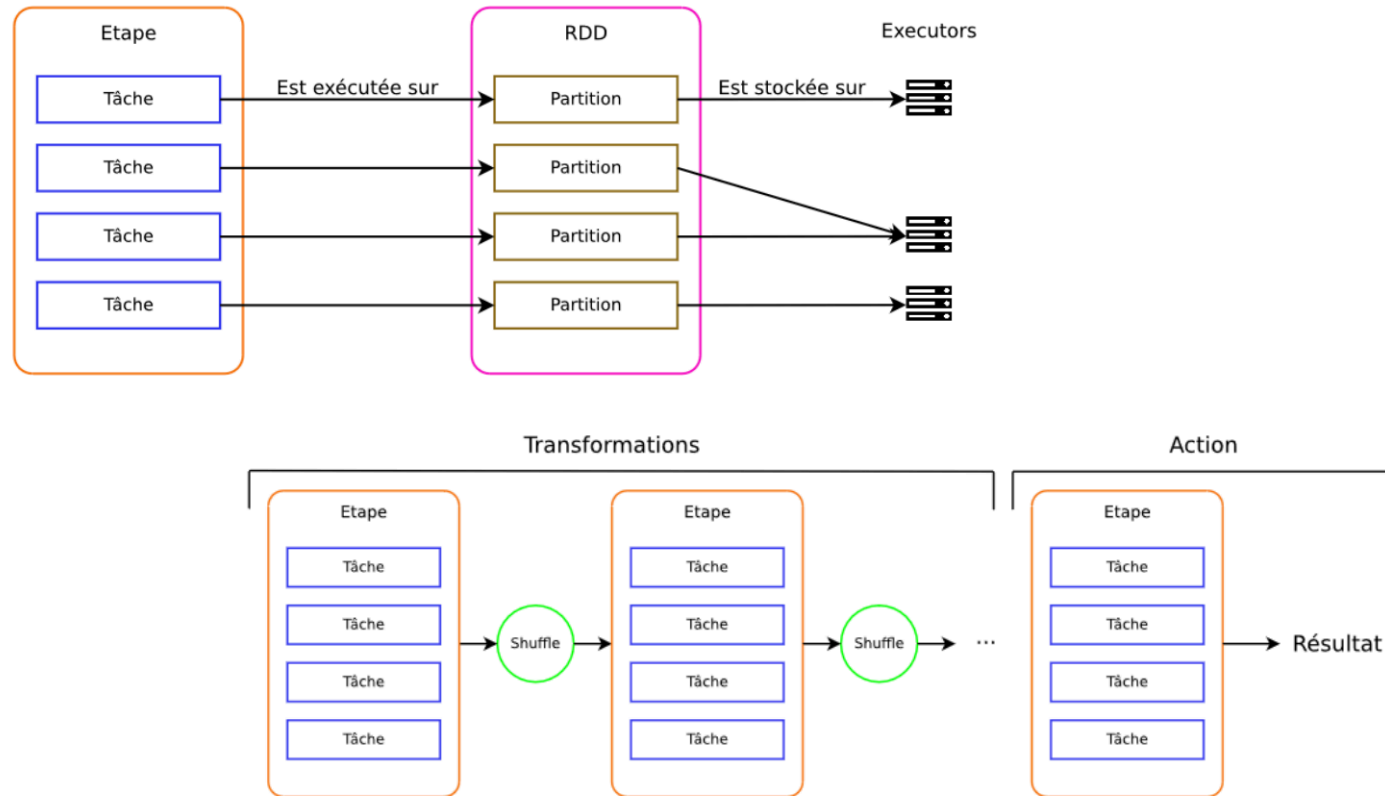


Comment répondre à ces enjeux?

- Stockage : système de fichier distribué (ex : HDFS)
- Tolérance aux pannes
 - Utilisation de Resilient Distributed Datasets (RDD)
 - Division des données en partitions
 - Duplication des données (3 machines par défaut)
 - Graphe Acyclique Orienté (DAG) :
 - Panne : Régénération à partir des noeuds parents
 - Noeuds (RDD ou Résultats) : liés par des actions et transformations



Comment répondre à ces enjeux?



Shuffle = redistribution des données entre les noeuds

**Architecture
retenue et chaîne
de traitement**



Architecture retenue et chaîne de traitement

Quel prétraitement?

Objectif: préparer les images pour le Learning

Réduction de dimensions

Extraction d'information des images

Solutions envisageables

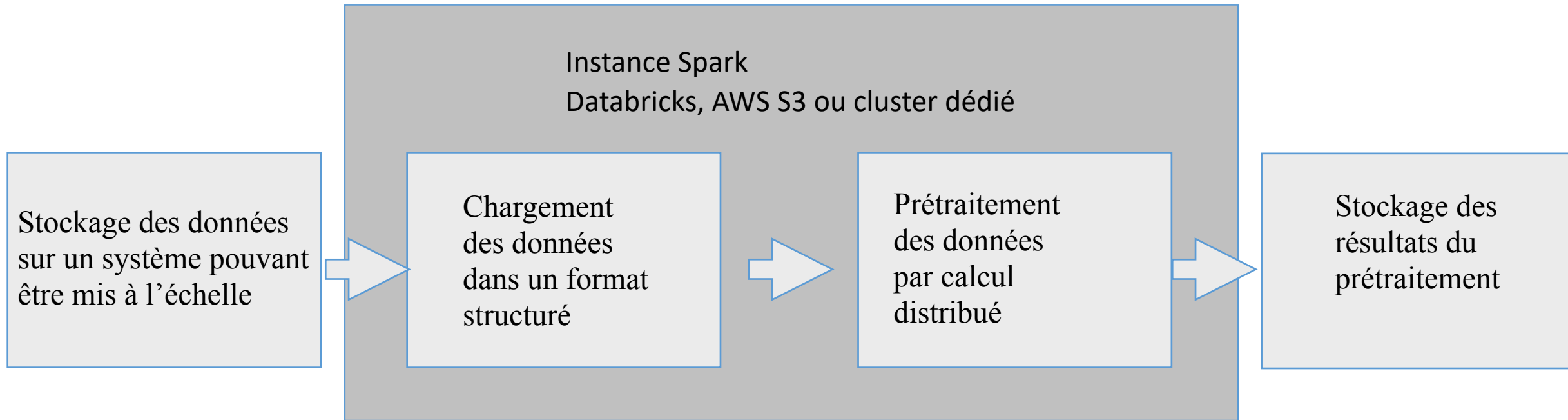
Redimensionnement

Traitement d'image +
Extraction de features

Algorithme
préentraînés

Architecture retenue et chaîne de traitement

Décomposition de la problématique



Architecture retenue et chaîne de traitement

Code avec Pyspark et Databricks

Liste des images dans l'arborescence

Création d'un Spark Dataframe (RDD) avec les chemins de tous les fichiers images à traiter

Extraction de la catégorie

Création d'une nouvelle colonne contenant la catégorie de chaque image

Chargement des images

Création d'une nouvelle colonne contenant le vecteur initial de chaque image

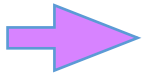
Réduction dimensionnelle

Création d'une nouvelle colonne contenant l'extraction des descripteurs de chaque image

Création du moteur Spark

```
spark = (SparkSession
    .builder
    .appName('P8')
    .master('local')
    .config("spark.sql.parquet.writeLegacyFormat", 'true')
    .getOrCreate())
```

```
sc = spark.sparkContext
```



	path	label	features
	dbfs:/FileStore/t...	Apple Golden 3	[0.017500168, 0.1...
	dbfs:/FileStore/t...	Apple Golden 3	[0.03510401, 0.03...
	dbfs:/FileStore/t...	Apple Golden 3	[0.046940308, 8.5...
	dbfs:/FileStore/t...	Apple Golden 3	[0.7509915, 0.0, ...]
	dbfs:/FileStore/t...	Apple Golden 3	[0.7926864, 0.0, ...]
	dbfs:/FileStore/t...	Apple Golden 3	[0.68344444, 0.0, ...]
	dbfs:/FileStore/t...	Apple Golden 3	[1.2806036, 0.073...
	dbfs:/FileStore/t...	Apple Golden 3	[1.3994256, 0.046...
	dbfs:/FileStore/t...	Banana	[1.4588063, 0.217...
	dbfs:/FileStore/t...	Banana	[0.031361267, 0.0...
	dbfs:/FileStore/t...	Banana	[1.4642526, 0.0, ...]
	dbfs:/FileStore/t...	Banana	[1.3081231, 0.064...
	dbfs:/FileStore/t...	Banana	[1.4883254, 0.054...
	dbfs:/FileStore/t...	Banana	[0.042054642, 0.1...
	dbfs:/FileStore/t...	Banana	[0.08807566, 0.25...
	dbfs:/FileStore/t...	Banana	[0.55912465, 0.04...
	dbfs:/FileStore/t...	Banana	[0.065307796, 0.4...
	dbfs:/FileStore/t...	Banana	[0.10381209, 0.45...

Chargement des données
Charger les images en format
« jpg » et le lire en format binaire

```
PATH = '/FileStore/tables/projet_8/'
```

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

Préparation du modèle
Charger le modèle MobileNetV2
avec les poids précalculés issus
d'imagenet

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

Extraction des features
Exécuter la featurisation sur
l'ensemble de notre DataFrame
Spark

```
features_df = images.repartition(20).select(col("path"),
    col("label"),
    featurize_udf("content").alias("features")
)
```

Affiche dataframe
C'est l'action .show() ici qui permet
de lancer toute la procédure ci-
dessus

```
df.show()
```

Architecture retenue et chaîne de traitement

Réduction de dimension

Réducteur

Transformer des tableaux en vecteurs pour effectuer une réduction

```
from pyspark.ml.linalg import Vectors, VectorUDT
array_to_vector_udf = udf(lambda l: Vectors.dense(l), VectorUDT())
```

Initialiser et appliquer PCA

Les résultats peuvent varier avec un ensemble de données plus grand, crée une tâche lourde Action qui affecterait les performances globales.

```
import time
from pyspark.ml.feature import PCA
start = time.perf_counter()
pca = PCA(k=3, inputCol='cnn_vectors', outputCol='pca_vectors')
model = pca.fit(vectorized_df)
stop = time.perf_counter()
print(f'pca - fit best k nb, elapsed time: {stop - start:0.2f}s')
```

Transformation inverse : de vecteurs à tableau

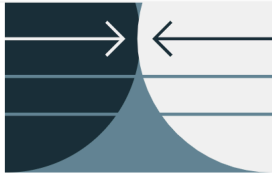
```
vector_to_array_udf = udf(lambda v: v.toArray().tolist())
```

Architecture retenue et chaîne de traitement

Réduction de dimension

```
+-----+-----+-----+-----+-----+
-+
|           path|       label|       features|       cnn_vectors|       pca_vector
s|
+-----+-----+-----+-----+-----+
-+
|dbfs:/FileStore/t...|Apple Golden 3|[-5.7529388963716...|[0.01750016771256...|[-5.752938896371
6...|
|dbfs:/FileStore/t...|Apple Golden 3|[-7.0583027208667...|[0.03510401025414...|[-7.058302720866
7...|
|dbfs:/FileStore/t...|Apple Golden 3|[-6.7665607771005...|[0.04694030806422...|[-6.766560777100
5...|
|dbfs:/FileStore/t...|Apple Golden 3|[-6.1485216111750...|[0.75099152326583...|[-6.148521611175
0...|
|dbfs:/FileStore/t...|Apple Golden 3|[-6.1959023618131...|[0.79268640279769...|[-6.195902361813
1...|
+-----+-----+-----+-----+-----+
-+
```


Architecture retenue et chaîne de traitement

Name	Status	Pricing tier	Region	Bucket name	Credentials name	Created	Metadata
							
<h3>Workspaces</h3> <p>Your workspace is the environment for doing work in Databricks. Create one to get started.</p> <p>Create workspace</p> <p>Quickstart (recommended)</p> <p>Custom AWS configuration</p>							

Let's set up your workspace

We're going to send you to your AWS Console to configure your account.

Once you sign in, we'll pre-populate a CloudFormation template that creates an IAM role and S3 bucket for you, then deploys your workspace.

If you encounter any errors during the process, visit the [Databricks Community](#) for troubleshooting guidance.

Workspace Name Nom du projet

Human readable name for your workspace

AWS Region of the Databricks workspace Choisir la région

E2, CMK, PL

AWS Region where the workspace will be created

☐ I have data in S3 that I want to query with Databricks

[Cancel](#) [Start Quickstart](#)

Créer une pile rapidement

Modèle

URL modèle
<https://databricks-prod-public-cfts.s3.us-west-2.amazonaws.com/templates/default-cluster-grow-553/databricks-trial.template.yaml>

Description de la pile
Set up resources and deploy a Databricks workspace in your AWS account. If you encounter any errors during the process, visit this Databricks Community post for troubleshooting guidance: <https://databricks.co/AWSQuickStartHelp>

Nom de la pile Nom du pile

Nom de la pile

Le nom de pile peut inclure des lettres (A-Z et a-z), des chiffres (0-9) et des tirets (-).

Architecture retenue et chaîne de traitement

Création d'un cluster

[default]basic-starter-cluster

More

Terminate

Edit

Configuration

Notebooks (1)

Libraries

Event log

Spark UI

Driver logs

Metrics

Apps

Spark cluster UI - Master

Policy

Unrestricted

Multi node

Single node

Access mode

Custom

Performance

11.0 (includes Apache Spark 3.3.0, Scala 2.12)

Summary

2-8 Workers

61-244 GB Memory

8-32 Cores

1 Driver

30.5 GB Memory, 4 Cores

Runtime

11.0.x-scala2.12

i3.xlarge

3-9 DBU/h

Use Photon Acceleration

Workers type

Min workers

Max workers

Current

2

8

2

Driver type

1

30.5 GB Memory, 4 Cores

Enable autoscaling

Enable autotuning local storage

Advanced options

On-demand/spot composition

2-8 Workers: 61-244 GB Memory, 8-32 Cores

0 All Spot

1 Driver

2-8 Workers

Spot fall back to On-demand

IAM role passthrough

Enable credential passthrough for user-level data access

Instances

Spark

Logging

Init Scripts

JDBC/ODBC

Permissions

SSH

Availability zone

eu-central-1b

Max spot price

100% of on-demand instance price

Version du cluster

Type de worker

Type de driver

La région du cluster

Téléchargement des fichiers

DBFS

Upload File

DBFS

Select a file from DBFS

FileStore

tables

featurized_sample

test

Apple Braeburn

Apple Crimson Snow

Apple Pink Lady

Apple Red 1

Apple Red 3

Architecture retenue et chaîne de traitement

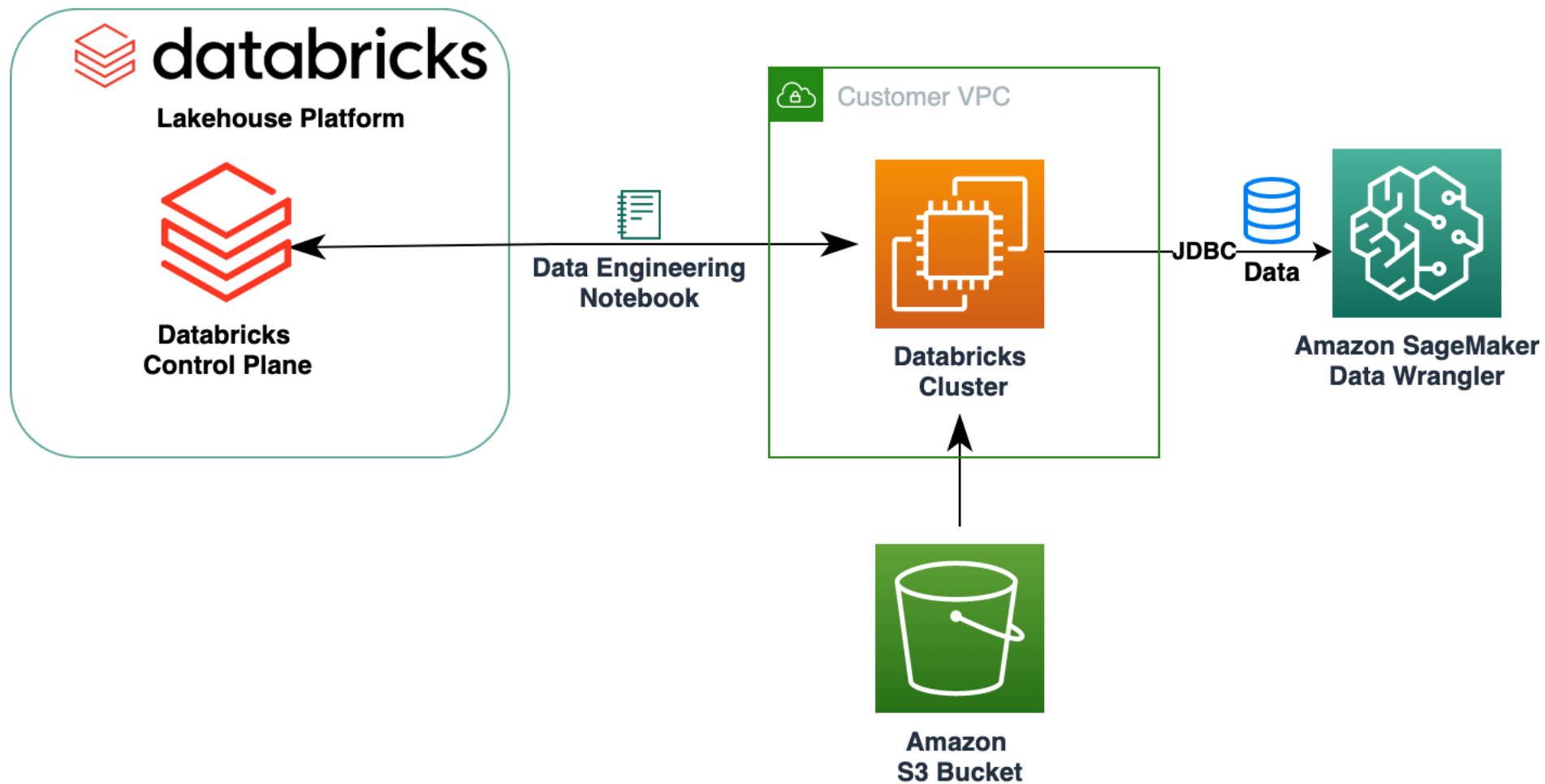
La Bucket trouvé avec S3 Amazon

	Nom	Région AWS	Accéder	Date de création
<input type="radio"/>	databricks-startupfruit-stack-lambdazipsbucket-1gawfps5mktci	UE (Francfort) eu-central-1	Les objets peuvent être publics	15 Feb 2023 01:24:22 PM C
<input type="radio"/>	db-d514e3a73fe3bef6fbe66b39fab22dc0-s3-root-bucket	UE (Francfort) eu-central-1	Compartiment et objets non publics	15 Feb 2023 01:24:23 PM C



<input type="checkbox"/>	Nom	Type	Dernière modification	Taille	Classe de stockage
<input type="checkbox"/>	part-00010-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-90-1-c000.snappy.parquet	parquet	15 Feb 2023 03:10:53 PM CET	52.7 Ko	Standard
<input type="checkbox"/>	part-00011-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-91-1-c000.snappy.parquet	parquet	15 Feb 2023 03:10:53 PM CET	51.7 Ko	Standard
<input type="checkbox"/>	part-00012-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-92-1-c000.snappy.parquet	parquet	15 Feb 2023 03:10:57 PM CET	51.8 Ko	Standard
<input type="checkbox"/>	part-00013-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-93-1-c000.snappy.parquet	parquet	15 Feb 2023 03:10:57 PM CET	47.5 Ko	Standard
<input type="checkbox"/>	part-00014-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-94-1-c000.snappy.parquet	parquet	15 Feb 2023 03:10:58 PM CET	48.9 Ko	Standard
<input type="checkbox"/>	part-00015-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-95-1-c000.snappy.parquet	parquet	15 Feb 2023 03:10:58 PM CET	48.4 Ko	Standard
<input type="checkbox"/>	part-00016-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-96-1-c000.snappy.parquet	parquet	15 Feb 2023 03:11:02 PM CET	48.2 Ko	Standard
<input type="checkbox"/>	part-00017-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-97-1-c000.snappy.parquet	parquet	15 Feb 2023 03:11:02 PM CET	47.0 Ko	Standard
<input type="checkbox"/>	part-00018-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-98-1-c000.snappy.parquet	parquet	15 Feb 2023 03:11:02 PM CET	42.6 Ko	Standard
<input type="checkbox"/>	part-00019-tid-8743318253540963959-86cac222-79f4-491f-ae63-5598f1277b56-99-1-c000.snappy.parquet	parquet	15 Feb 2023 03:11:03 PM CET	39.8 Ko	Standard

Architecture retenue et chaîne de traitement



4 → **Conclusion**

Conclusion

- Enseignements
 - Prise en main Pyspark
 - Découverte du format distribué parquet
 - Découverte de l'écosystème AWS
 - Découverte de Datbricks

le déploiement de l'application PySpark sur un cluster AWS, beaucoup plus économique et qui permet un passage à l'échelle très simple.

Choix d'un modèle adapté avec le client.
Avec le déploiement, bien penser l'évolution de l'infrastructure en fonction des besoins et du budget alloué.



***Merci pour votre
attention!***



Questions

