Réaliser un traitement dans un environnement Big Data sur le Cloud

Lien vers le Github : https://github.com/Bastien441237/P8_OpenClassroomsProject.git

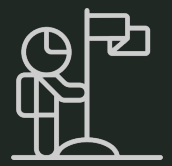
Sommaire

Contexte du projet

Fruits!

Problématique : La start-up Fruits souhaite se faire connaître en mettant à disposition du grand public une application mobile qui permettrait aux utilisateurs de prendre en photo un fruit et d'obtenir des informations sur ce fruit

Objectif : Mettre en place une première version de l'architecture Big Data pour l'application qui servira ensuite le projet principal qui est de développer des robots cueilleurs intelligents



Missions :

- S'approprier les travaux réalisés par l'alternant
- Compléter la chaine de traitement en Pyspark pour effectuer du calcul distribué
- Mettre en place les premières briques de traitement
- Migrer la chaîne de traitement sur le Cloud (AWS)

Résultat attendu :

- Un notebook sur le Cloud contenant les scripts en Pyspark exécutables
- Les images du jeu de données initial
- Une sortie de la réduction de dimension (en CSV) disponible sur un espace de stockage Cloud

Présentation du jeu de données

Le jeu de données est le Fruits-360 dataset que l'on peut retrouver sur kaggle : https://www.kaggle.com/datasets/moltean/fruits

Nombre d'images : 90,380

Nombre de fruits différents : 131

Utilité : Ce dataset est une excellente base pour la création de l'architecture de notre base de données Big Data.

Chaîne de traitement des images .2/6

Stockage et chargement des images

STOCKAGE

- DATA : Dossier avec les images de fruits
- RESULTS : Dossier qui contiendra les caractéristiques des images
- EXPORT CSV : Dossier avec la sortie en CSV

Chargement des images :

```
+--------------------+-------------------+------+--------------------+
|                path|   modificationTime|length|             content|
+--------------------+-------------------+------+--------------------+
|s3://p8-data-bast...|2024-05-15 18:31:07|  7353|[FF D8 FF E0 00 1...|
|s3://p8-data-bast...|2024-05-15 18:31:07|  7350|[FF D8 FF E0 00 1...|
|s3://p8-data-bast...|2024-05-15 18:31:07|  7349|[FF D8 FF E0 00 1...|
|s3://p8-data-bast...|2024-05-15 18:31:07|  7348|[FF D8 FF E0 00 1...|
|s3://p8-data-bast...|2024-05-15 18:31:09|  7328|[FF D8 FF E0 00 1...|
+--------------------+-------------------+------+--------------------+
```

Chaîne de traitement des images .1/6

Quelques mots sur PySpark

PySpark est l'interface Python pour Apache Spark, un moteur de traitement de données massivement parallèle. Il permet d'effectuer des calculs distribués sur de grands ensembles de données de manière efficace.

Principales fonctionnalités :

- Traitement de données à grande échelle : capable de gérer des pétaoctets de données.
- Intégration avec Hadoop : fonctionne bien avec les systèmes de fichiers distribués comme HDFS.
- Machine Learning : inclut MLlib pour des algorithmes de machine learning distribués.
- SQL et DataFrames : supporte les requêtes SQL et les manipulations de données structurées.

Avantages :

- Vitesse : traite les données beaucoup plus rapidement que les systèmes traditionnels.
- Évolutivité : peut s'adapter à des clusters de milliers de nœuds.
- Simplicité : syntaxe Python simple et intuitive pour les opérations complexes.

Chaîne de traitement des images .3/6

Préparation du modèle

Le modèle MobileNetV2 a été choisi pour sa rapidité d'exécution, car il est adapté au traitement des gros volumes de données et pour la faible dimensionnalité du vecteur de caractéristiques en sortie.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

On va donc extraire les features des images grâce au transfer learning.

En enlevant la dernière couche de classification du modèle on obtient le vecteur de caractéristiques :

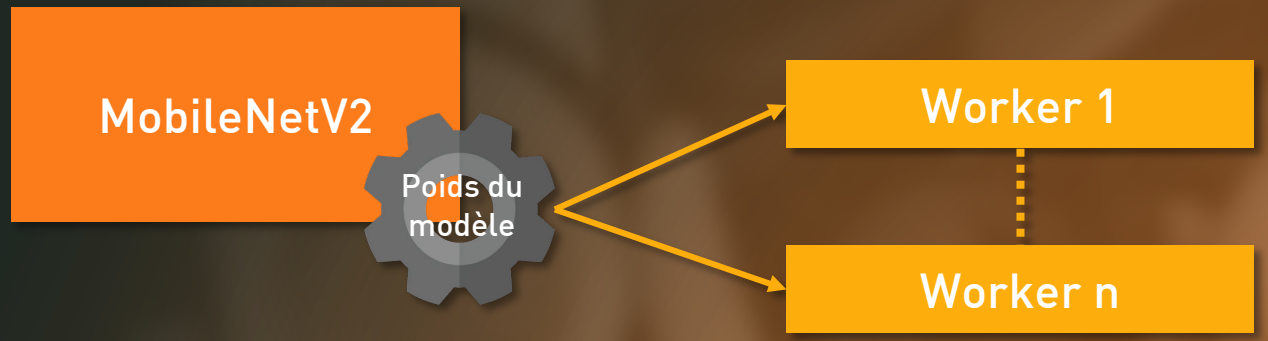
(1 x 1 x 1280)

On aura donc 1280 variables pour caractériser le images.

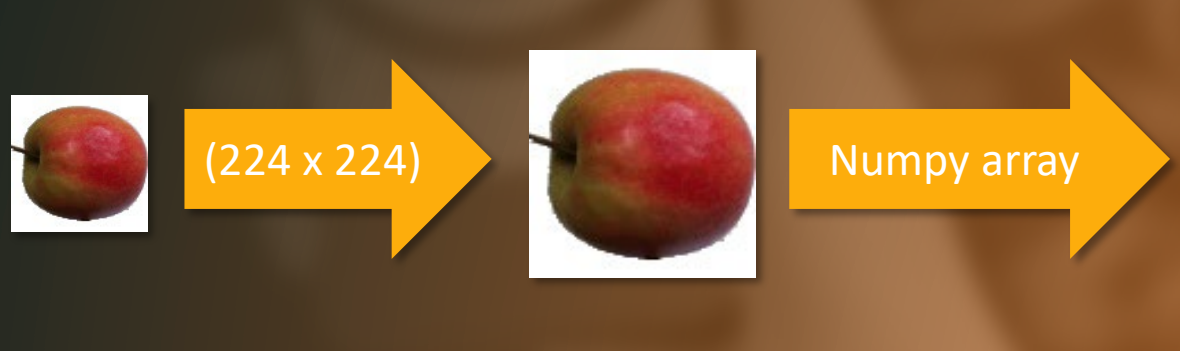
Chaîne de traitement des images .4/6

Définition des poids, préprocessing et featurisation

Il faut diffuser les poids du modèle aux différents nœuds du cluster Spark pour que chaque nœud dispose des mêmes poids lors des opérations distribuées.



Il faut ensuite pouvoir préprocesser les images brutes en les transformant en image utilisable par le modèle.



Puis on va stocker les caractéristiques de l'image dans un seul vecteur sous forme de série Pandas.

[caractéristique1, caractéristique2, … , caractéristique1280]

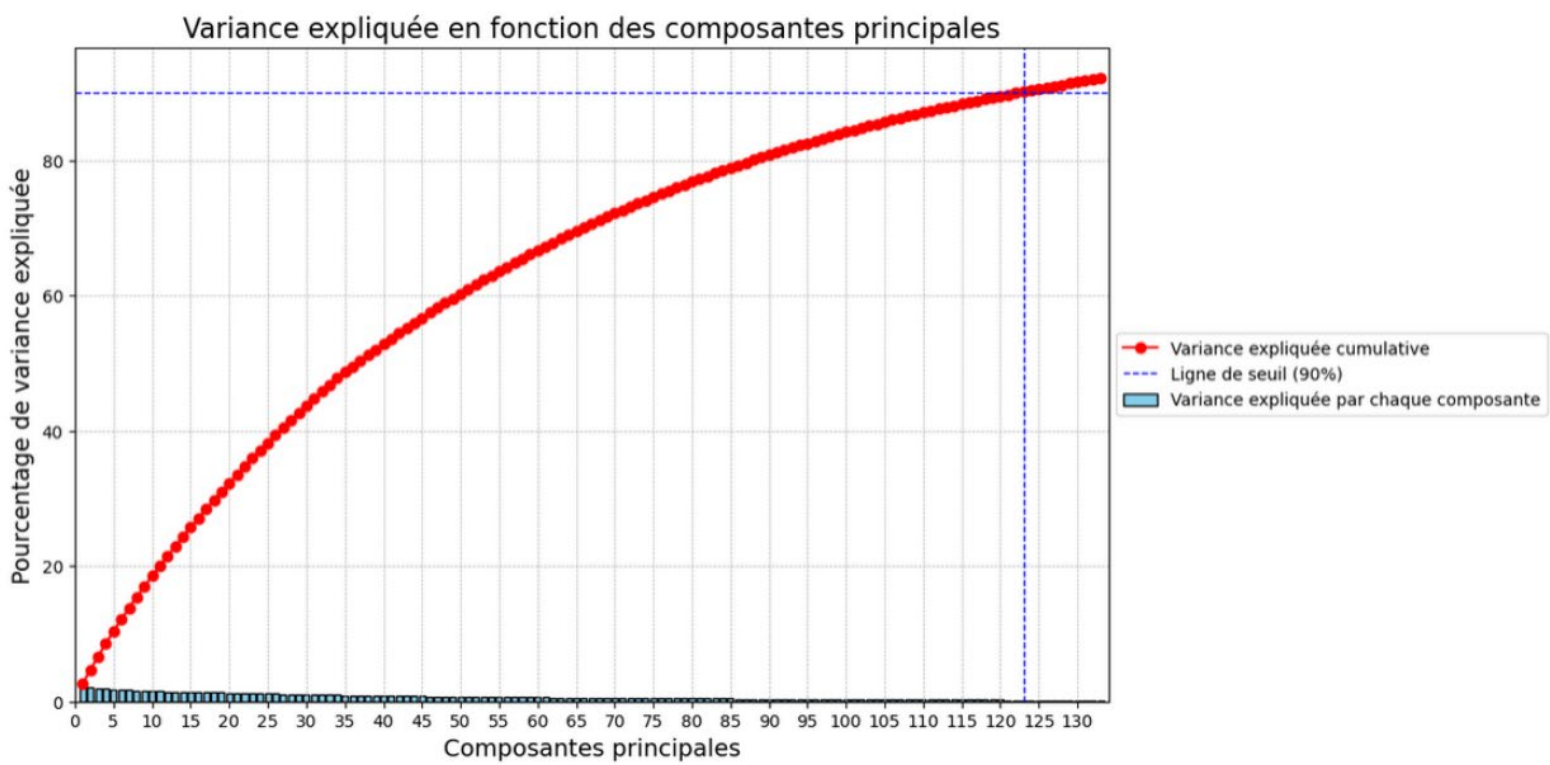
Chaîne de traitement des images .5/6

Analyse en Composantes Principales (ACP)

Pour terminer la chaîne de traitement, on doit réaliser une diminution de dimensionnalité afin de réduire la consommation des traitements futurs.

On doit donc centrer et réduire les caractéristiques des images dans un premier temps.

Puis, le but est de savoir combien de composantes expliquent un maximum de variance expliquée.



Ici, on voit que 123 composantes sur 1280 expliquent 90% de la variance.

On peut donc réduire les caractéristiques à un vecteur de [, 123].

Chaîne de traitement des images .6/6

Stockage des résultats

Il nous faut ensuite stocker une sortie à notre chaîne.

```
+--------------------+-----------------+--------------------+
|                path|            label|  features_pca_array|
+--------------------+-----------------+--------------------+
|file:/home/bastie...|            Cocos|[-5.6063524217208...|
|file:/home/bastie...|    Nectarine Flat|[10.3650136358250...|
|file:/home/bastie...|    Nectarine Flat|[9.58629377904878...|
|file:/home/bastie...|Tomato not Ripened|[-2.6434576540140...|
|file:/home/bastie...|Tomato not Ripened|[-4.8078410185243...|
+--------------------+-----------------+--------------------+
only showing top 5 rows
```

/Results

Parquet

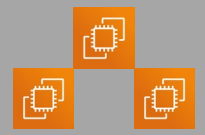
	path	label	features_pca_array
0	file:/home/bastien/Documents/P8_/data/Test1/Pe...	Peach Flat	[7.363243975530696, -7.059919770889569, -0.682...
1	file:/home/bastien/Documents/P8_/data/Test1/Ap...	Apple Crimson Snow	[11.482035949516904, -4.068673682681118, 0.657...
2	file:/home/bastien/Documents/P8_/data/Test1/Co...	Cocos	[-4.888620858267934, 19.243194999786798, 0.127...
3	file:/home/bastien/Documents/P8_/data/Test1/To...	Tomato not Ripened	[-2.68613878820059, 3.9367374402902087, -0.984...
4	file:/home/bastien/Documents/P8_/data/Test1/Me...	Melon Piel de Sapo	[-14.755015309750215, 23.353922085484328, 2.04...

/Export_csv

CSV

Création de l'environnement Big Data .1/2

Choix des briques d'architecture du cloud



Bucket S3

Pour stocker les images du DataSet et les futures images de l'application mobile

Amazon EMR

Permet l'exécution d'Apache Spark pour le calcul distribué et Jupyter pour le lancement du script PySpark

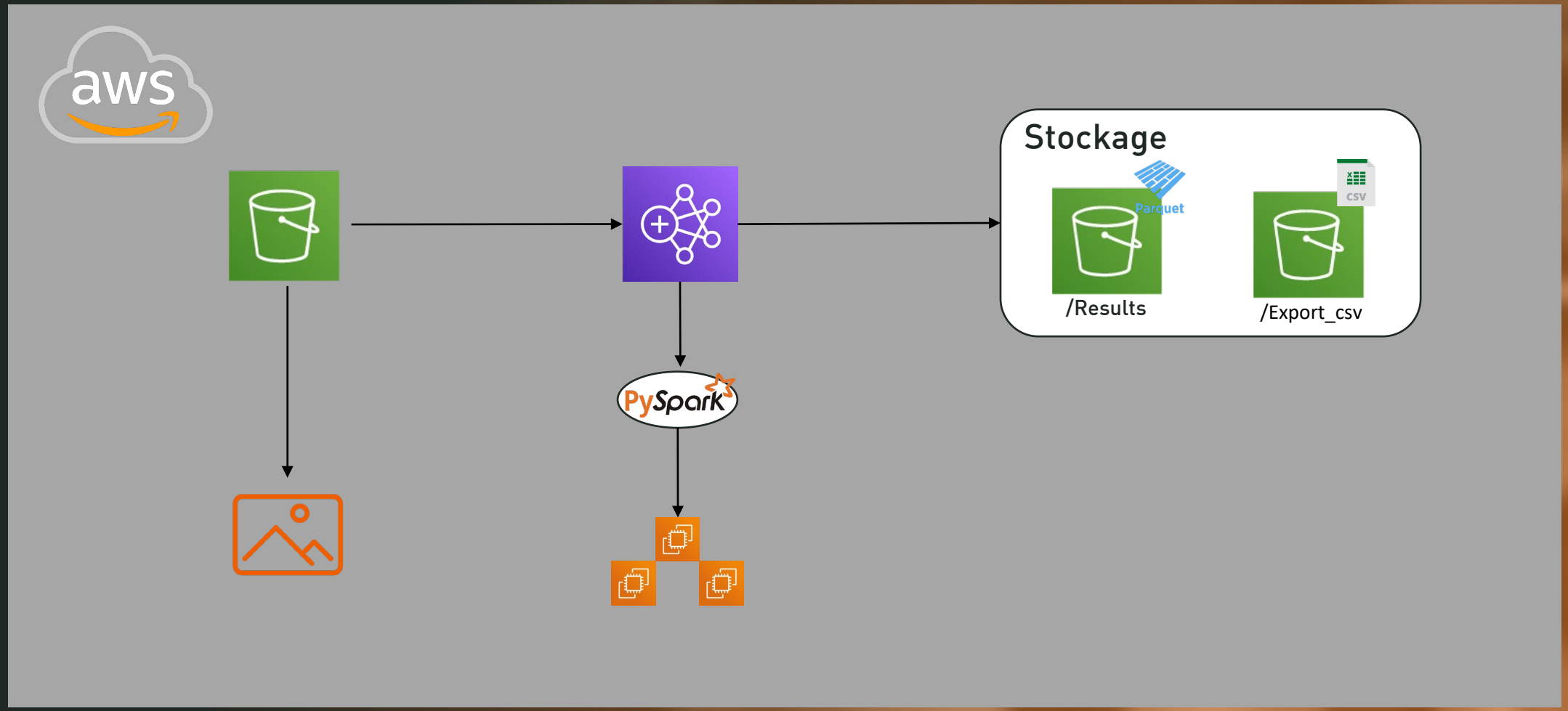
Instances EC2

Instances créées pour l'EMR afin de réaliser le calcul distribué

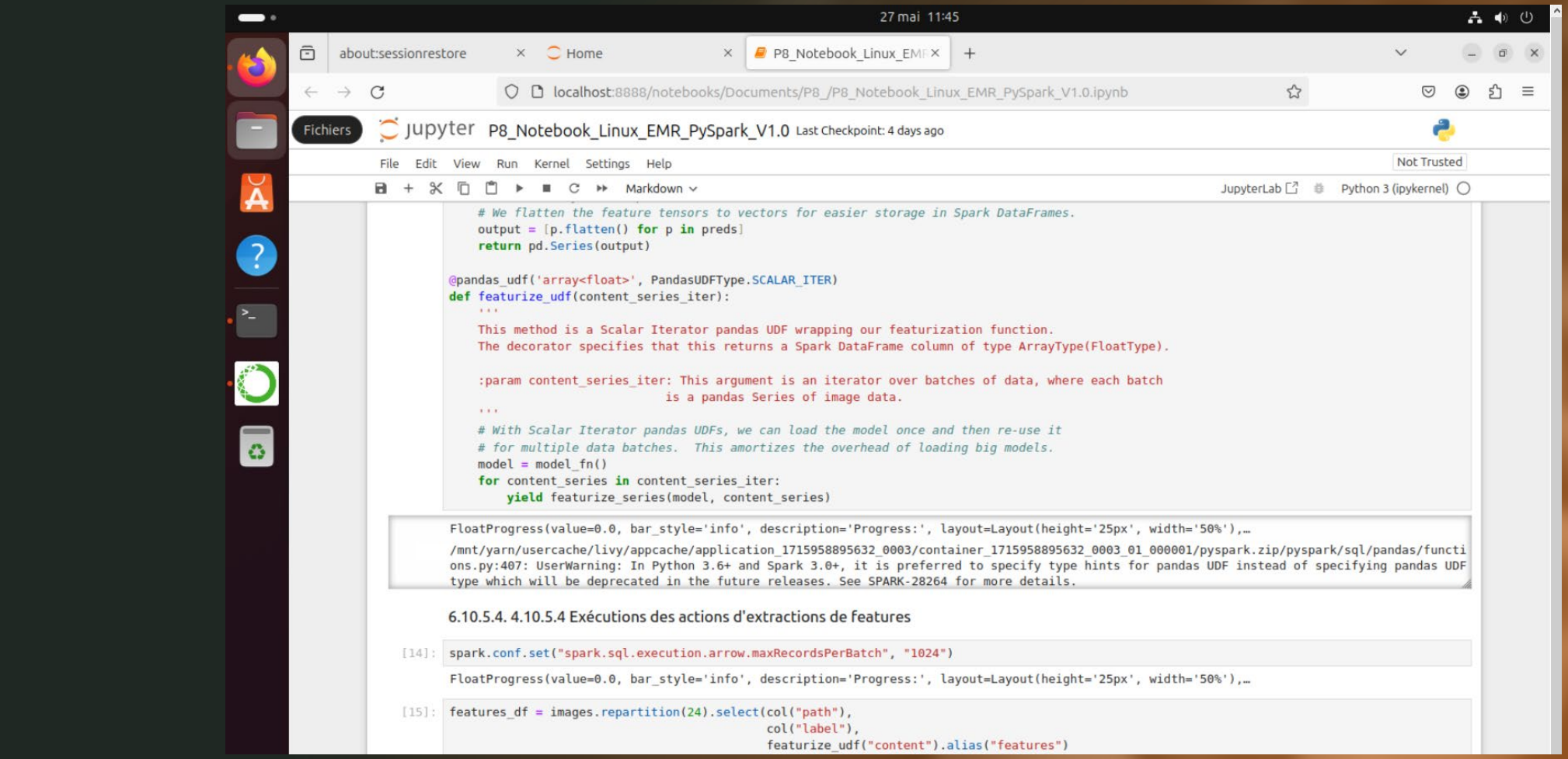
Actuellement au nombre de 3 (m5.xlarge)

Création de l'environnement Big Data .2/2

Architecture Cloud



Démonstration du script sur le Cloud



Conclusion

Nous pouvons maintenant utiliser AWS pour :

- Stocker via S3 les données qui vont croître au fur et à mesure de l'utilisation de l'application.
- Utiliser des instances EC2 via EMR pour réaliser des calculs distribués afin d'améliorer le temps de traitement de la pipeline de traitement des images présentes dans le bucket S3