



DÉPLOYEZ UN MODÈLE DANS LE CLOUD

FORMATION DATA SCIENTIST – PROJET 8

OCTAVE POUILLOT

OCTOBRE 2023



Fruits!

SOMMAIRE

- Mission & Dataset
- Environnement Big Data
- Preprocessing
- Démonstration
- Conclusion



MISSION

- Data Scientist dans la start-up de l'AgriTech "Fruits!", qui cherche à proposer des solutions innovantes pour la récolte des fruits.
- Se faire connaître par une application mobile permettant de prendre en photo un fruit et d'obtenir des informations sur ce fruit.

Objectifs :

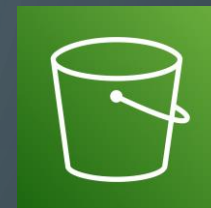
- S'appropriier les travaux réalisés par l'alternant
- Compléter la chaîne de traitement
- Mettre en œuvre une architecture Big Data



DATASET

Images de fruits :

- 90483 images
- 67692 images train / 22688 images test
- 131 classes (variétés de fruits)
- Dossier "Test1" de 30 images, 3 fruits (banane, citron, pomme)



ENVIRONNEMENT BIG DATA

MACHINE VIRTUEL

ARCHITECTURE GLOBALE

IAM / S3 / EMR

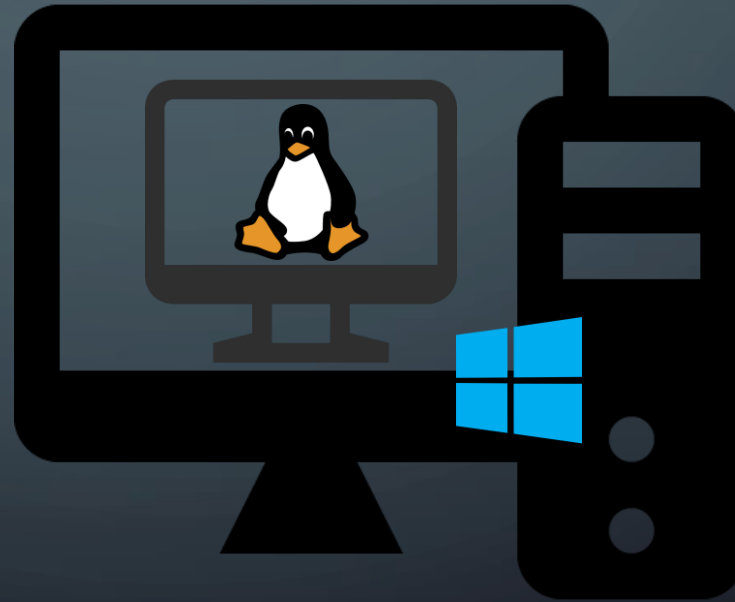
TUNNEL SSH



ENVIRONNEMENT BIG DATA

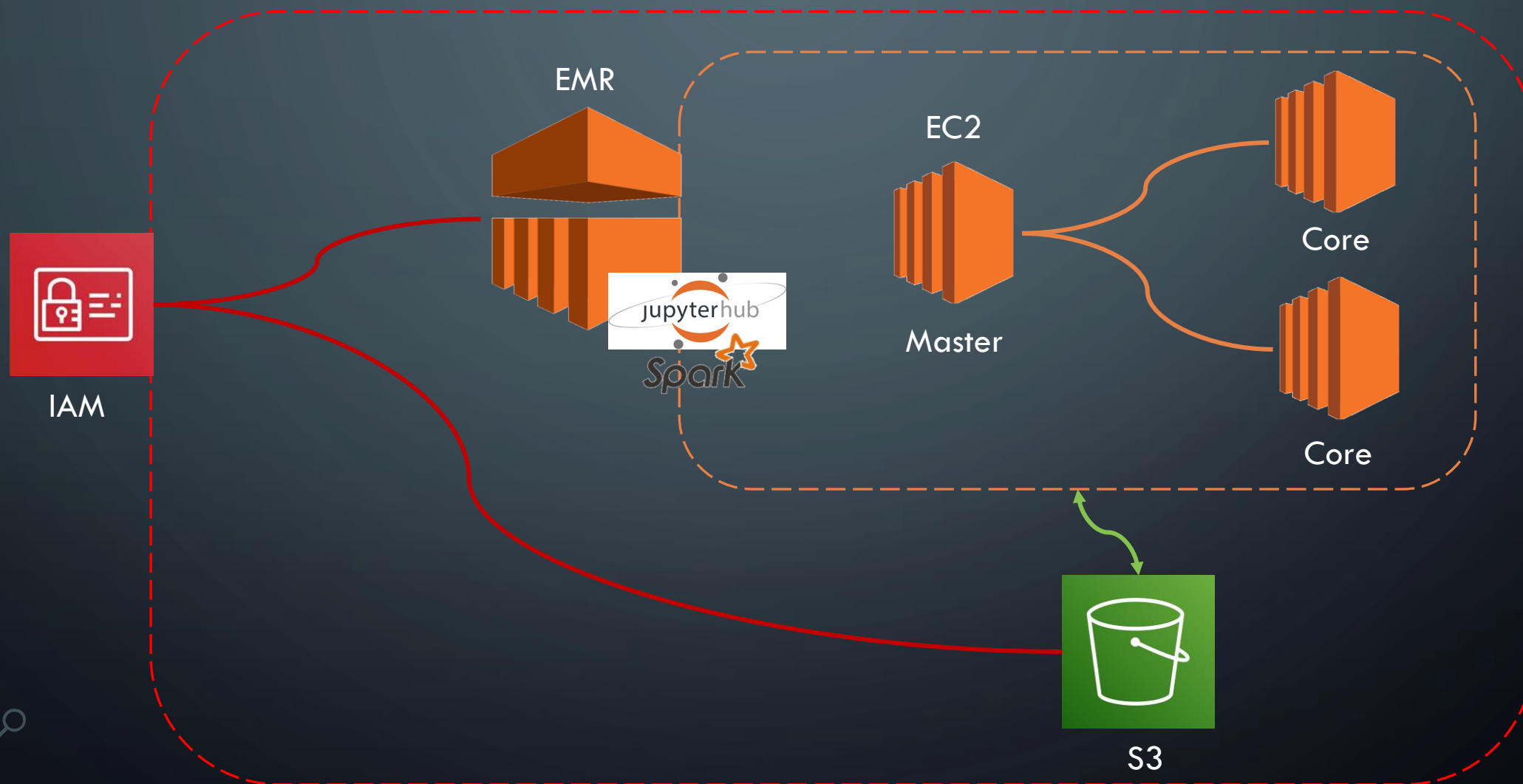


- Virtual Box
- Linux Ubuntu
- Installation :
 - Pip
 - Spark
 - Jupyter
 - Aws cli





ENVIRONNEMENT BIG DATA



ENVIRONNEMENT BIG DATA



- Root user
 - All access
 - Gestion EMR
- UserP8
 - S3 Full Access
 - AWS CLI
 - Génération de clés SSH
- OPouillotP8Evaluator
 - S3 Read Access

ENVIRONNEMENT BIG DATA



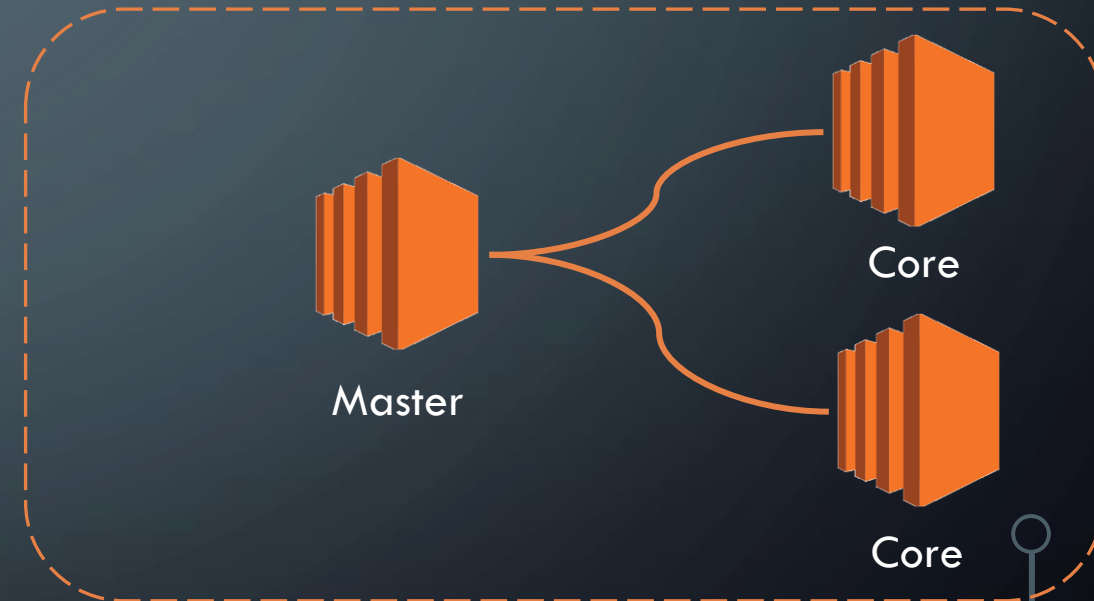
- Définition de la région sur Paris (eu-west-3)
- Création d'un bucket « p8octave-dataset »
- Copie des données « Test1 »

```
aws s3 mb s3://p8octave-dataset
```

```
aws sync . s3://p8octave-dataset /Test
```

ENVIRONNEMENT BIG DATA

- Région : Paris (eu-west-3)
- Applications
 - Spark
 - JupyterHub
 - Tensorflow
- Instances c6g.xlarge (\$)
- Résiliation auto : 1h



ENVIRONNEMENT BIG DATA



- Création d'un fichier Bootstrap :
 - Upgrade setuptools & pip
 - Installation des librairies nécessaires
+ Tensorflow
- Clés SSH EC2

ENVIRONNEMENT BIG DATA



aws Services [Search] [Alt+S] Paris

✓ Your cluster "P8_fruits" has been successfully created.

Amazon EMR > EMR on EC2: Clusters > P8_fruits

P8_fruits

Updated less than a minute ago [Refresh] [Terminate] [Clone in AWS CLI] [Clone]

▼ Summary

Cluster info	Applications	Cluster management	Status and time
<p>Cluster ID</p> <p>J-1ES3C595L0IY</p> <p>Cluster configuration</p> <p>Instance groups</p> <p>Capacity</p> <p>1 Primary 2 Core 1 Task</p>	<p>Amazon EMR version</p> <p>emr-6.9.0</p> <p>Installed applications</p> <p>JupyterHub 1.4.1, Spark 3.3.0</p>	<p>Log destination in Amazon S3</p> <p>aws-logs-752957977056-eu-west-3/elasticmapreduce</p> <p>Primary node public DNS</p> <p>ec2-15-237-41-27.eu-west-3.compute.amazonaws.com</p> <p>Connect to the Primary Node using SSH</p>	<p>Status</p> <p>⌚ Bootstrapping</p> <p>Creation time</p> <p>October 09, 2023, 08:42 (UTC+02:00)</p> <p>Elapsed time</p> <p>3 minutes, 2 seconds</p>

< Properties Bootstrap actions Instances (Hardware) Steps Applications Configurations Monitoring **Events** >

Events (1) [Info](#)

< 1 >

Status and time

Status

✓ **Waiting**

Creation time

October 09, 2023, 08:42
(UTC+02:00)

Elapsed time

6 minutes, 59 seconds

ENVIRONNEMENT BIG DATA



- Connexion SSH

- -D 5555

Connect to the primary node using SSH

You can connect to the Amazon EMR primary node using SSH to perform actions like running interactive queries, examining log files, submit Linux commands, and view web interfaces hosted on Amazon EMR clusters. [Learn more](#)

Windows

Mac/Linux

1. Open a terminal window. On Mac OS X, choose Applications > Utilities > Terminal. On other Linux distributions, terminal is typically found at Applications > Accessories > Terminal.

2. To establish a connection to the primary node, enter the following command. Replace `~/octave-ec2-rsa.pem` with the location and filename of the private key file (.pem) that you used to launch the cluster.

```
ssh -i ~/octave-ec2-rsa.pem hadoop@ec2-15-237-41-27.eu-west-3.compute.amazonaws.com
```

3. Enter yes to dismiss the security warning.

[View web interfaces hosted on Amazon EMR clusters](#)

Close

ENVIRONNEMENT BIG DATA



- Paramétrage de FoxyProxy

Extension (Foxy Standard) moz-extension://267eeb71-5c2d-473b-93e2-6f057de888cf/proxy.html

Edit Proxy EMR

Title or Description (optional)
EMR

Color
#66cc66


Send DNS through SOCKS5 proxy ☒ On

Proxy Type
SOCKS5

Proxy IP address or DNS name ★
localhost

Port ★
5555

Username (optional)
username

Password (optional) 

Cancel Save & Add Another Save & Edit Patterns Save

ENVIRONNEMENT BIG DATA



- Connexion à JupyterHub
- Upload du notebook
- Kernel PySpark

A screenshot of the AWS Management Console, specifically the EMR console, showing the details of an EMR cluster. The browser address bar shows the URL: https://eu-west-3.console.aws.amazon.com/emr/home?region=eu-west-3#/clusterDetails/j-1ES3... A "FoxyProxy" extension notification is visible in the top right corner. The main content area is titled "Application UIs" and contains two tables. The first table, "Application UIs on the primary node", lists the URLs for various services, with the "JupyterHub" entry highlighted by a red rectangle. The second table, "Application UIs on the core and task nodes", lists the URLs for the HDFS Data Node and Node Manager. At the bottom, the "Installed Applications (2)" section shows "JupyterHub 1.4.1" and "Spark 3.3.0".

Application	UI URL
HDFS Name Node	http://ec2-15-237-41-27.eu-west-3.compute.amazonaws.com:9870/
JupyterHub	https://ec2-15-237-41-27.eu-west-3.compute.amazonaws.com:9443/
Resource Manager	http://ec2-15-237-41-27.eu-west-3.compute.amazonaws.com:8088/
Spark History Server	http://ec2-15-237-41-27.eu-west-3.compute.amazonaws.com:18080/

Application	UI URL
HDFS Data Node	http://ec2-000-000-000-000.compute-1.amazonaws.com:9864/
Node Manager	http://ec2-000-000-000-000.compute-1.amazonaws.com:8042/

Installed Applications (2)	
JupyterHub 1.4.1	Spark 3.3.0



Fruits!

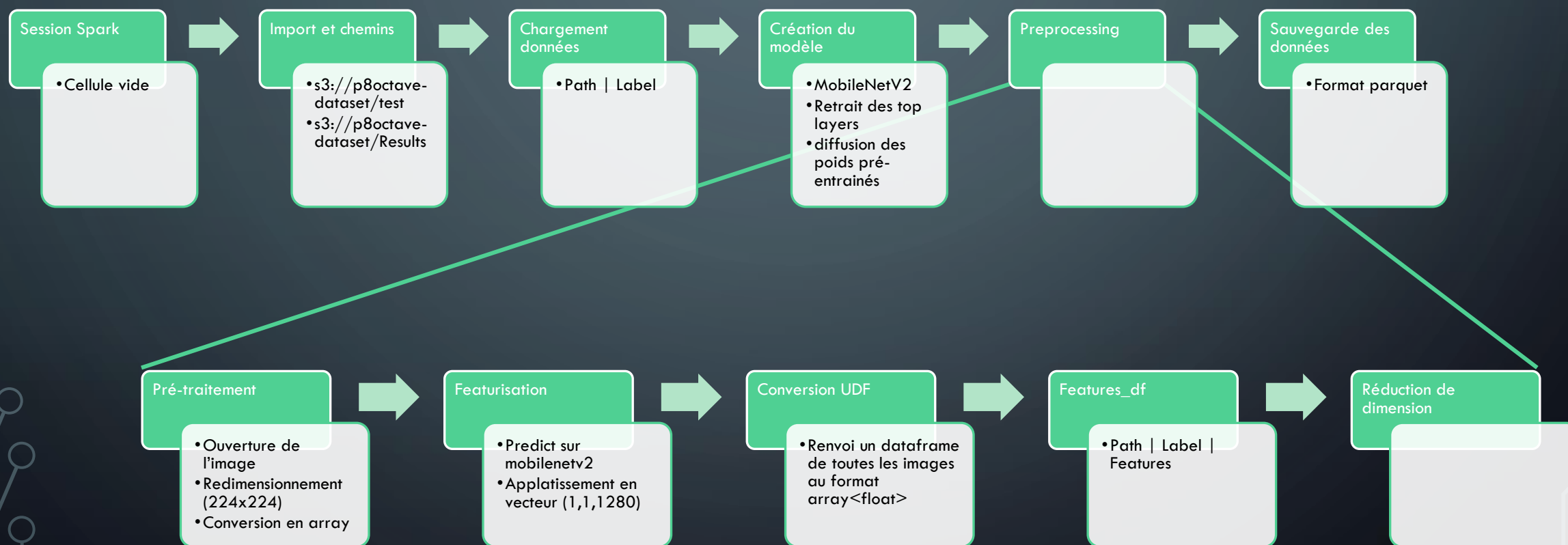
PREPROCESSING

ETAPES GÉNÉRALE

PREPROCESSING



PREPROCESSING



PREPROCESSING



```
In [*]: # Appliquer la conversion à la colonne 'features'
features_df = features_df.withColumn('features', array_to_vector(features_df['features']))

# Standardisation
scaler = StandardScaler(inputCol="features", outputCol="scaled_features")

# PCA
pca = PCA(k=2, inputCol="scaled_features", outputCol="pca_features")

# Création du pipeline
pipeline = Pipeline(stages=[scaler, pca])

# Application du pipeline sur le DataFrame
model = pipeline.fit(features_df)
result = model.transform(features_df)

# Sélection des colonnes
result = result.select("path", "label", "pca_features")

# Conversion des vecteurs en listes Python
features_df = result.withColumn("pca_features", vector_to_list("pca_features"))
```

Progress:

Conversion « features » array
→ vector

Définition StandardScaler

Définition PCA

Définition pipeline

Application du pipeline sur
« features »

Création d'un dataframe
« path | label | pca_features »

Conversion « pca_features »
vector → list



Fruits!

DÉMONSTRATION !



Fruits!

CONCLUSION



CONCLUSION

- Mise en place d'un environnement Big Data
- Lancement d'un notebook via JupyterHub
- PCA via PySpark

➔ PoC valide

- Prochaines étapes :
 - Entraînement et évaluation du modèle
 - Déploiement du modèle dans le cloud
 - Scaling automatique via EMR

