OpenClassRooms

Data Scientist

P8 Deploy a model in the cloud

Fruits!

Pictures used for educational purpose only

Benoît DELORME
13 janvier 2022

Sommaire

I. Introduction

II. Big Data environment

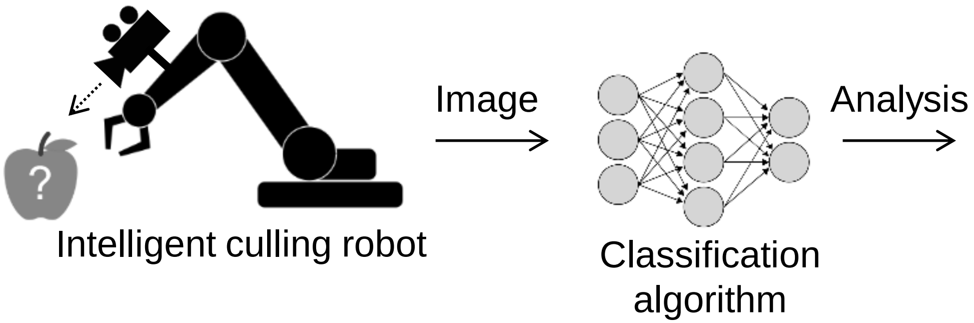
III. Image processing with Spark

IV. Conclusion

I. Introduction

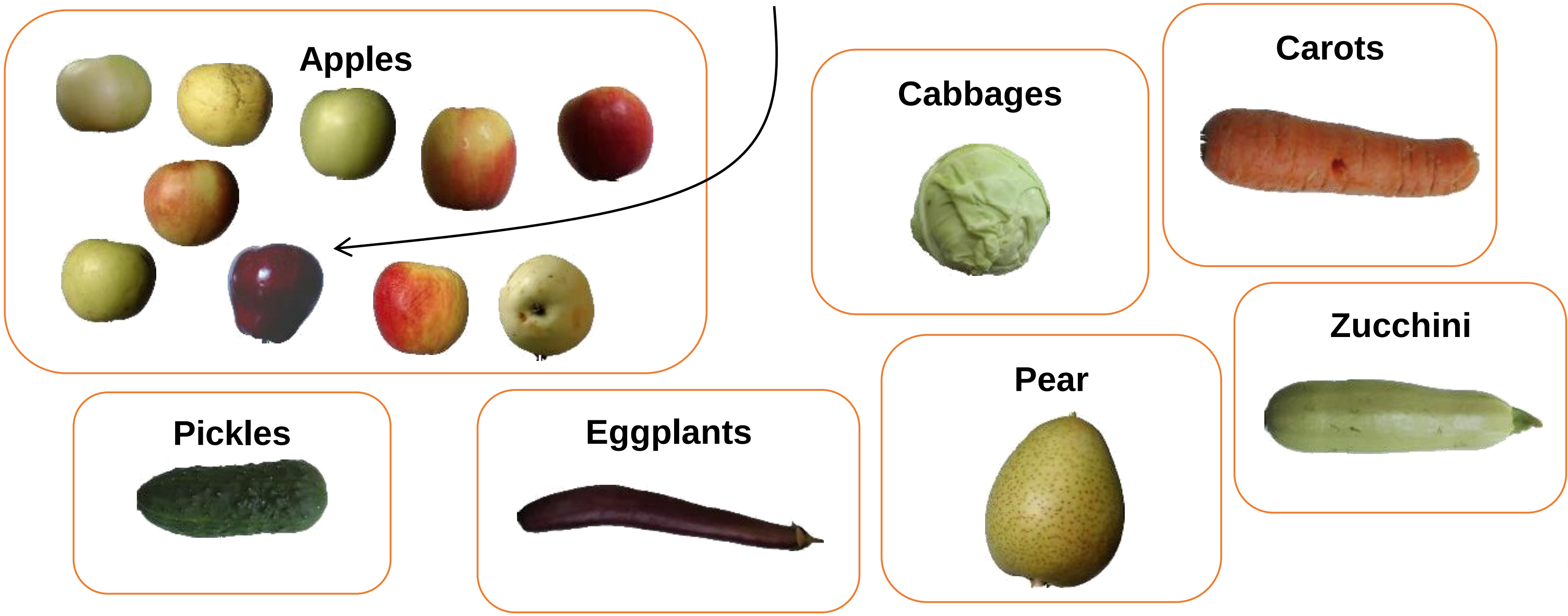
1) The project

Prototype of **mobile application** that enables to take a picture of a fruit and get information on it.
→ Final goal: provide with a specific treatment for each fruit species through intelligent culling robots.



Pomme Delicioso

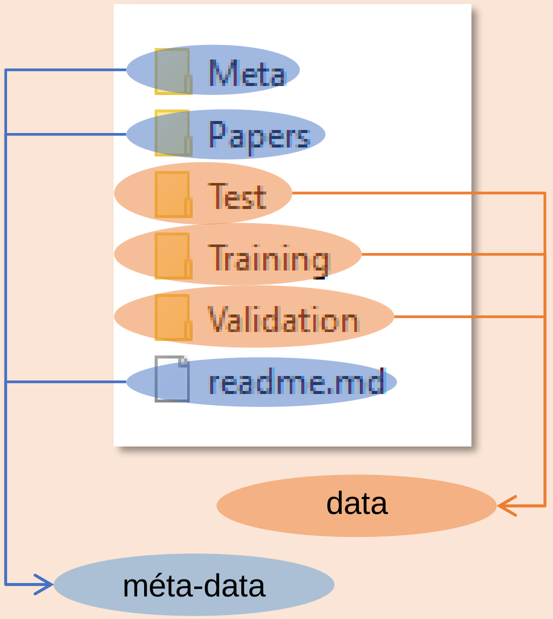
State	Good
Ripeness	120 days
Other	shocks
…	



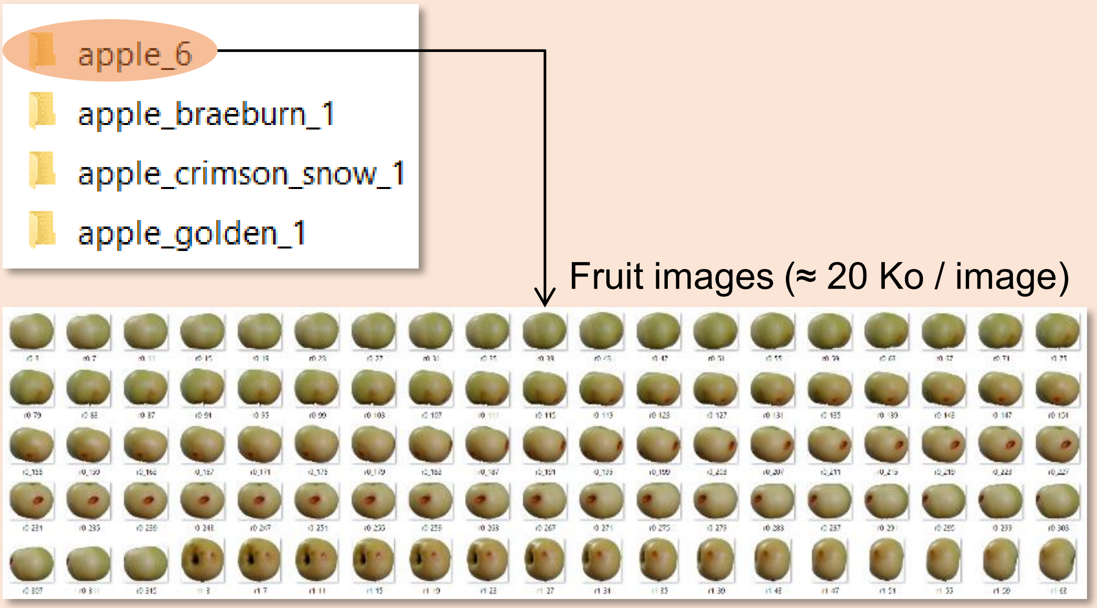
I. Introduction

2) The dataset

Root folder

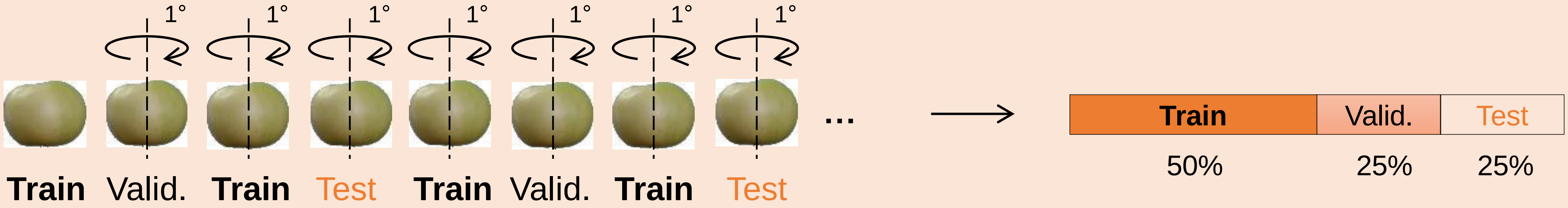


Folders *Test*, *Training* and *Validation*


apple_6
apple_braeburn_1
apple_crimson_snow_1
apple_golden_1

Fruit images (≈ 20 Ko / image)

Image repartition among sets

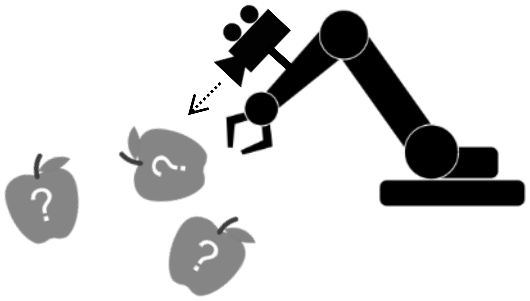


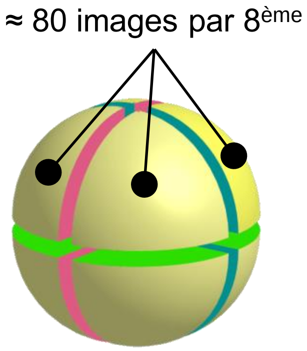
I. Introduction

3) Problematic

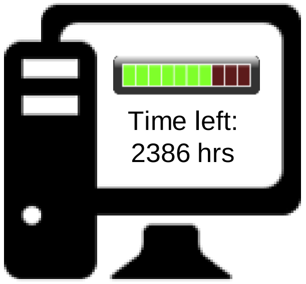
→ Currently 24 fruits in the database

→ Up to **650 photos per fruit**, in order to anticipate the different points of view of future images.

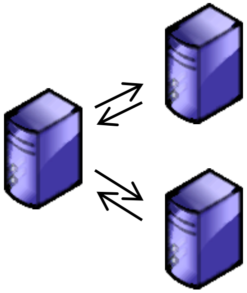




Problem: the database will grow with new fruits
→ risk of outgrowing regarding the **computing power** of a **single** computer.

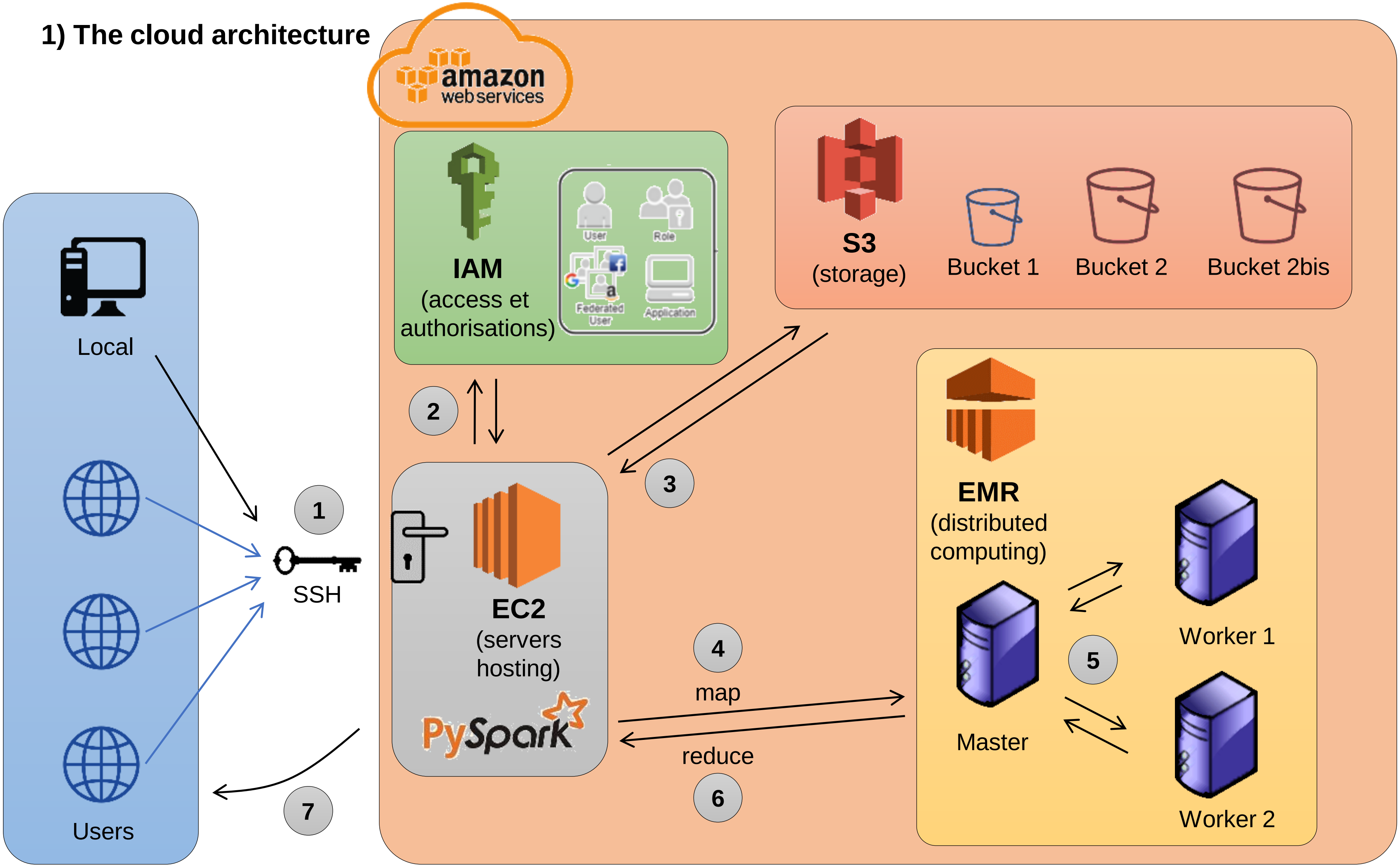


→ The future important data volume implies to scale up and use the **cloud** as well as **distributed computing**.



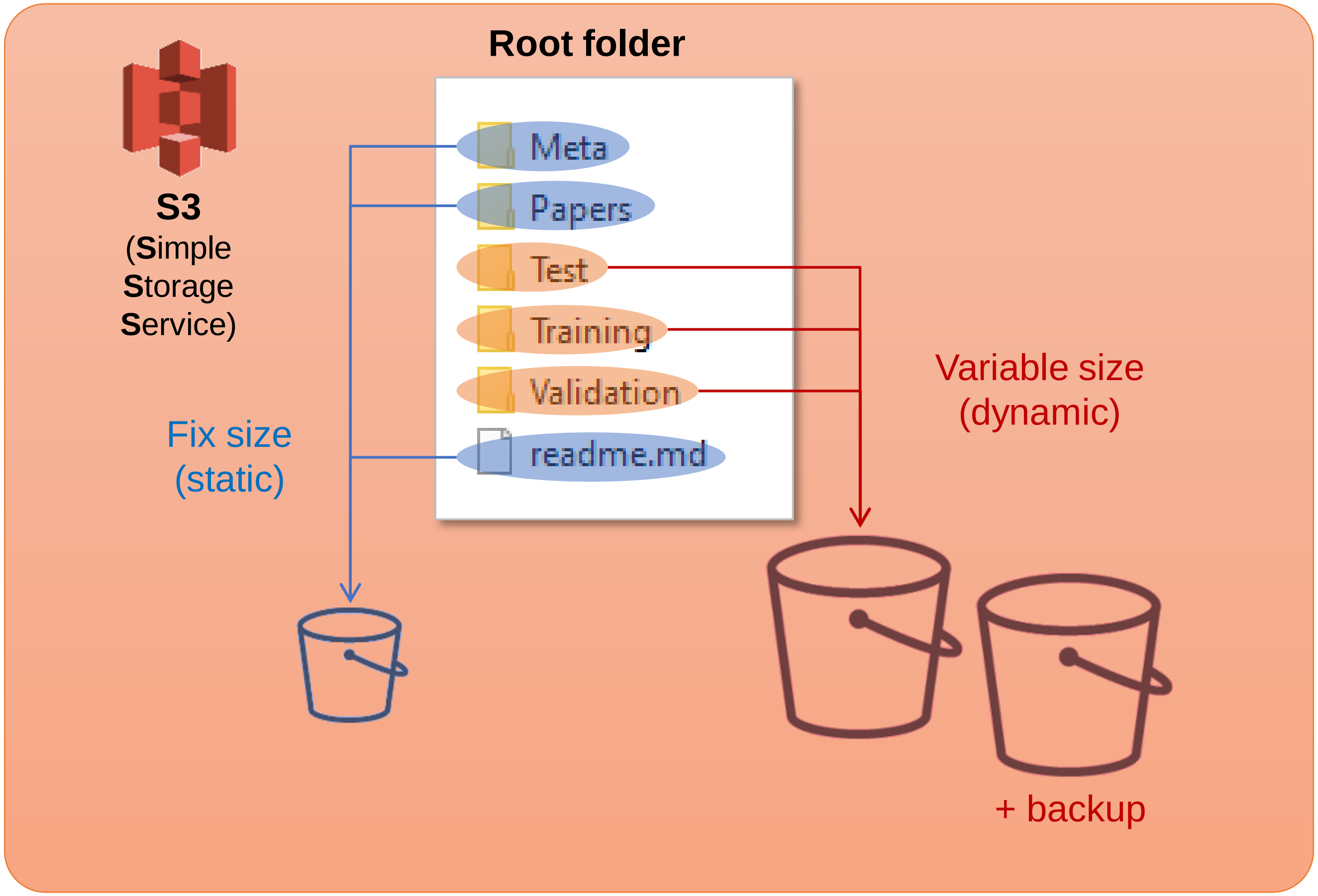
II. BigData environment

1) The cloud architecture



II. BigData environment

2) Data storage



Root folder

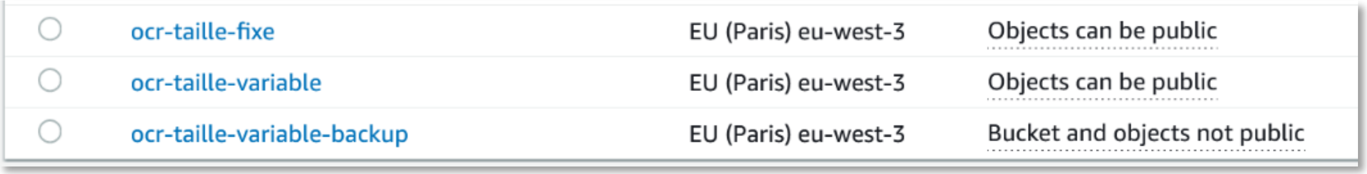
S3
(**S**imple
Storage
Service)

- Meta
- Papers
- Test
- Training
- Validation
- readme.md

Fix size
(static)

Variable size
(dynamic)

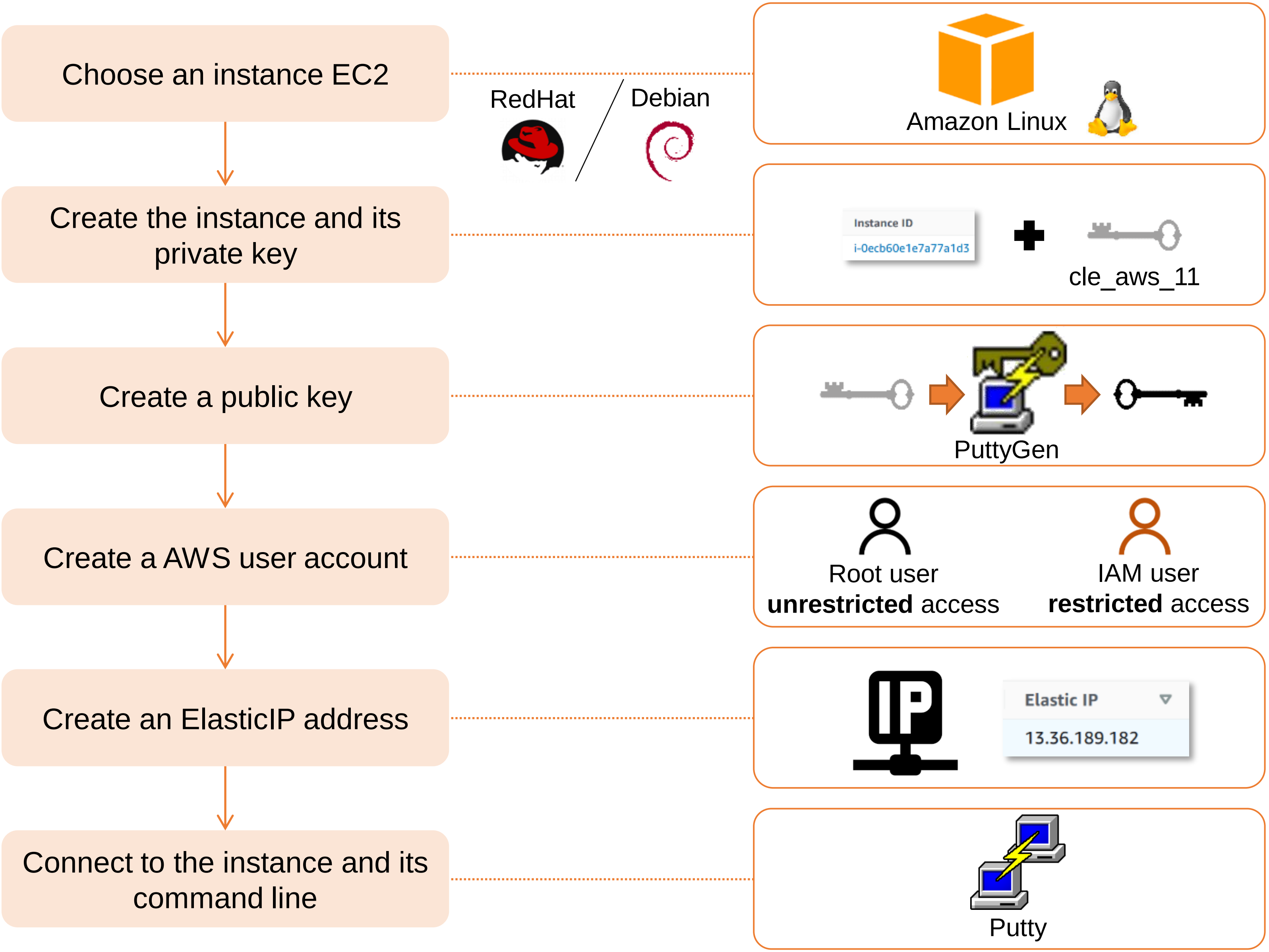
+ backup



○	ocr-taille-fixe	EU (Paris) eu-west-3	Objects can be public
○	ocr-taille-variable	EU (Paris) eu-west-3	Objects can be public
○	ocr-taille-variable-backup	EU (Paris) eu-west-3	Bucket and objects not public

II. BigData environment

3) Implementation of instance EC2



Choose an instance EC2

RedHat / Debian

Amazon Linux

Create the instance and its private key

Instance ID

i-0ecb60e1e7a77a1d3

cle_aws_11

Create a public key

PuttyGen

Create a AWS user account

Root user **unrestricted** access

IAM user **restricted** access

Create an ElasticIP address

Elastic IP

13.36.189.182

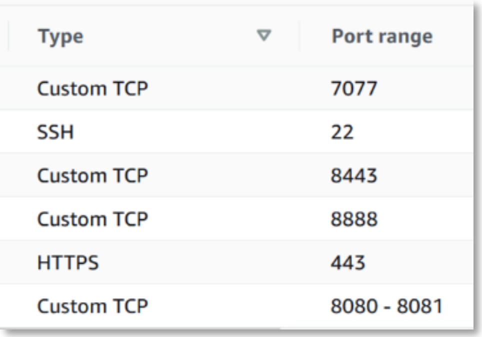
Connect to the instance and its command line

Putty

II. BigData environment

4) Preparation of Spark script

Implementation of security rules



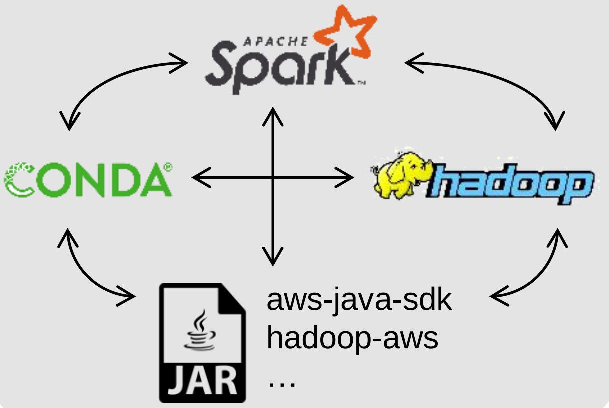
Type	Port range
Custom TCP	7077
SSH	22
Custom TCP	8443
Custom TCP	8888
HTTPS	443
Custom TCP	8080 - 8081

1ère try

Install Conda, Spark, .jar files

Install Tensorflow / Keras

Multiples errors of **memory** and **version** incompatibility



2ème try

Choose an instance :
→ **t3.large** size
→ prepared for **deep learning**
→ with tensorflow et conda **pre-installed**

Activate Tensorflow & Conda

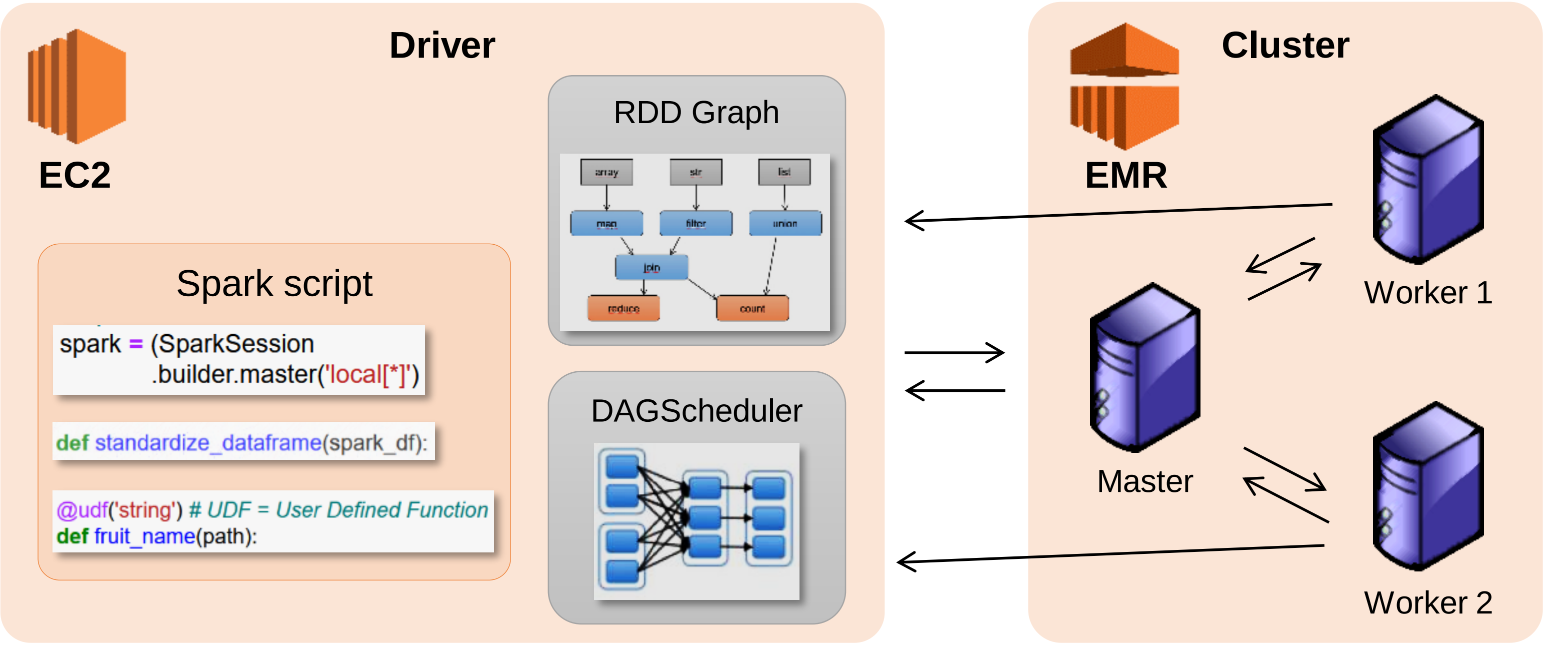
```
source activate tensorflow_p37
```

Open a Jupyter notebook

Write and execute the Spark script

II. BigData environment

5) Spark



```
spark = (SparkSession
         .builder.master('local[*]')
```

```
def standardize_dataframe(spark_df):
```

```
@udf('string') # UDF = User Defined Function
def fruit_name(path):
```

Programme Spark	Execute Spark instructions
RDD Graph	Spark Transformations and actions
DAGScheduler	Distribute computations according Hadoop MapReduce scheme
Master	Shares computations between workers
Workers	Execute computations

} Operations done by Spark

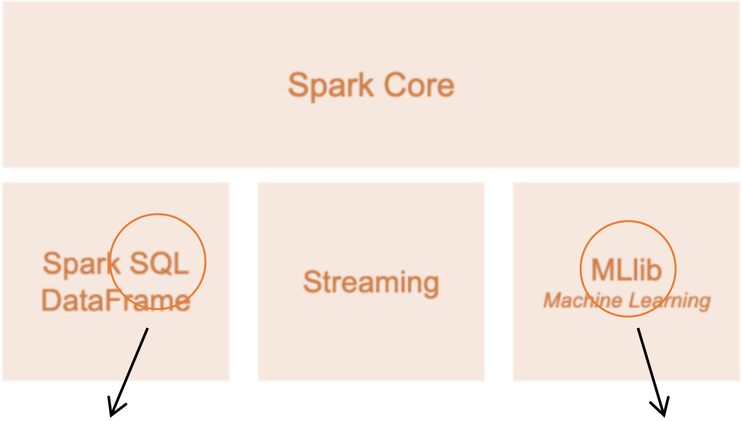
→ In the frame of this project, master and workers are hosted on the same server, i.e. the EC2 instance.

II. BigData environment

6) Libraries PySpark used

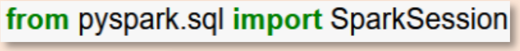
PySpark

(Python API for Spark)

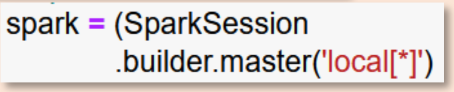


← PySpark libraries

1) Methods of the library

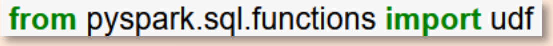


```
from pyspark.sql import SparkSession
```

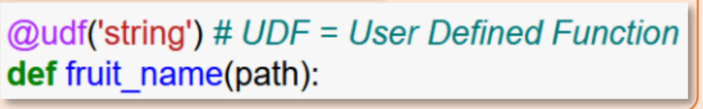


```
spark = (SparkSession
         .builder.master('local[*]')
```

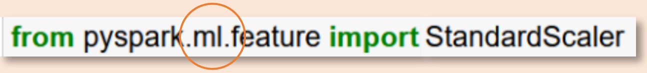
2) Decorators



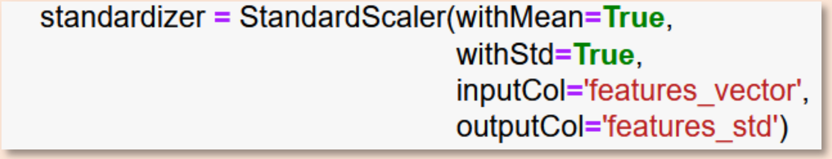
```
from pyspark.sql.functions import udf
```



```
@udf('string') # UDF = User Defined Function
def fruit_name(path):
```



```
from pyspark.ml.feature import StandardScaler
```



```
standardizer = StandardScaler(withMean=True,
                              withStd=True,
                              inputCol='features_vector',
                              outputCol='features_std')
```

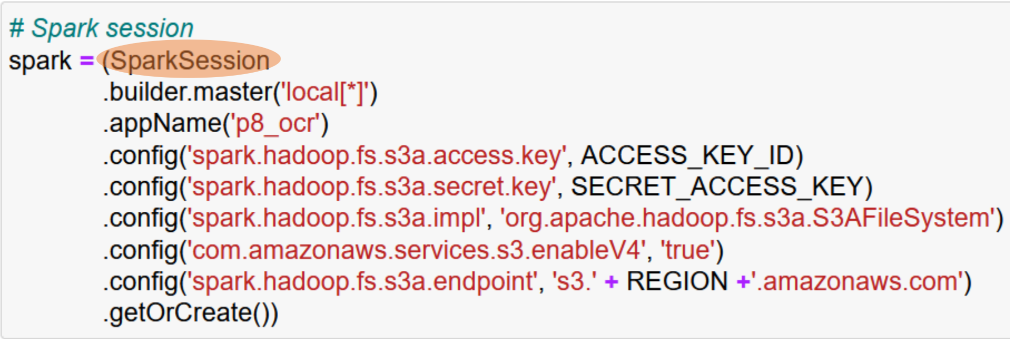
→ Computations can be distributed with **methods of the PySpark library**, or **decorated** functions.

III. Image processing with Spark

2) Spark script configuration

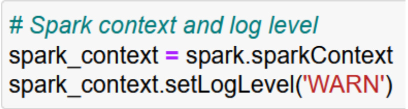
→ A Spark script needs additional settings in comparison to an usual python script.

SparkSession
→ entry point of all functionnality of Spark
→ encompasses all types of *contexts* :
Spark, Hive, SQL, …

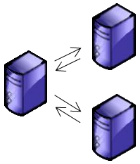


```
# Spark session
spark = (SparkSession
         .builder.master('local[*]')
         .appName('p8_ocr')
         .config('spark.hadoop.fs.s3a.access.key', ACCESS_KEY_ID)
         .config('spark.hadoop.fs.s3a.secret.key', SECRET_ACCESS_KEY)
         .config('spark.hadoop.fs.s3a.impl', 'org.apache.hadoop.fs.s3a.S3AFileSystem')
         .config('com.amazonaws.services.s3.enableV4', 'true')
         .config('spark.hadoop.fs.s3a.endpoint', 's3.' + REGION +'.amazonaws.com')
         .getOrCreate())
```

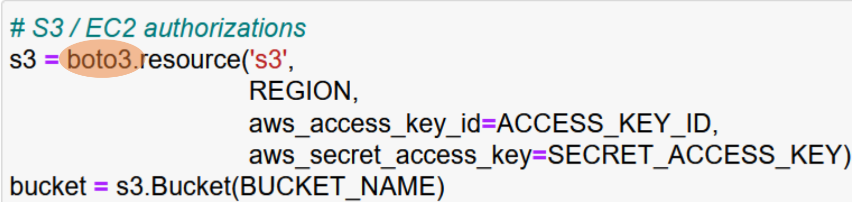
SparkContext
→ connexion to a master-workers **cluster**



```
# Spark context and log level
spark_context = spark.sparkContext
spark_context.setLogLevel('WARN')
```



boto3
→ “*Python Software Development Kit*” designed to configure and manage AWS services.
→ helps to configure the connexion to **S3**

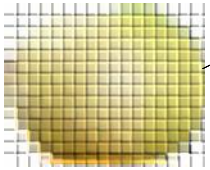
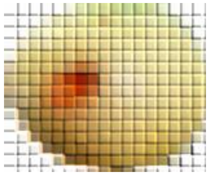
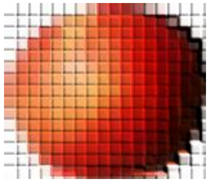
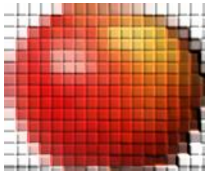


```
# S3 / EC2 authorizations
s3 = boto3.resource('s3',
                    REGION,
                    aws_access_key_id=ACCESS_KEY_ID,
                    aws_secret_access_key=SECRET_ACCESS_KEY)
bucket = s3.Bucket(BUCKET_NAME)
```

→ Now we can load images from S3 and use PySpark methods

III. Image processing with Spark

3) Raw table

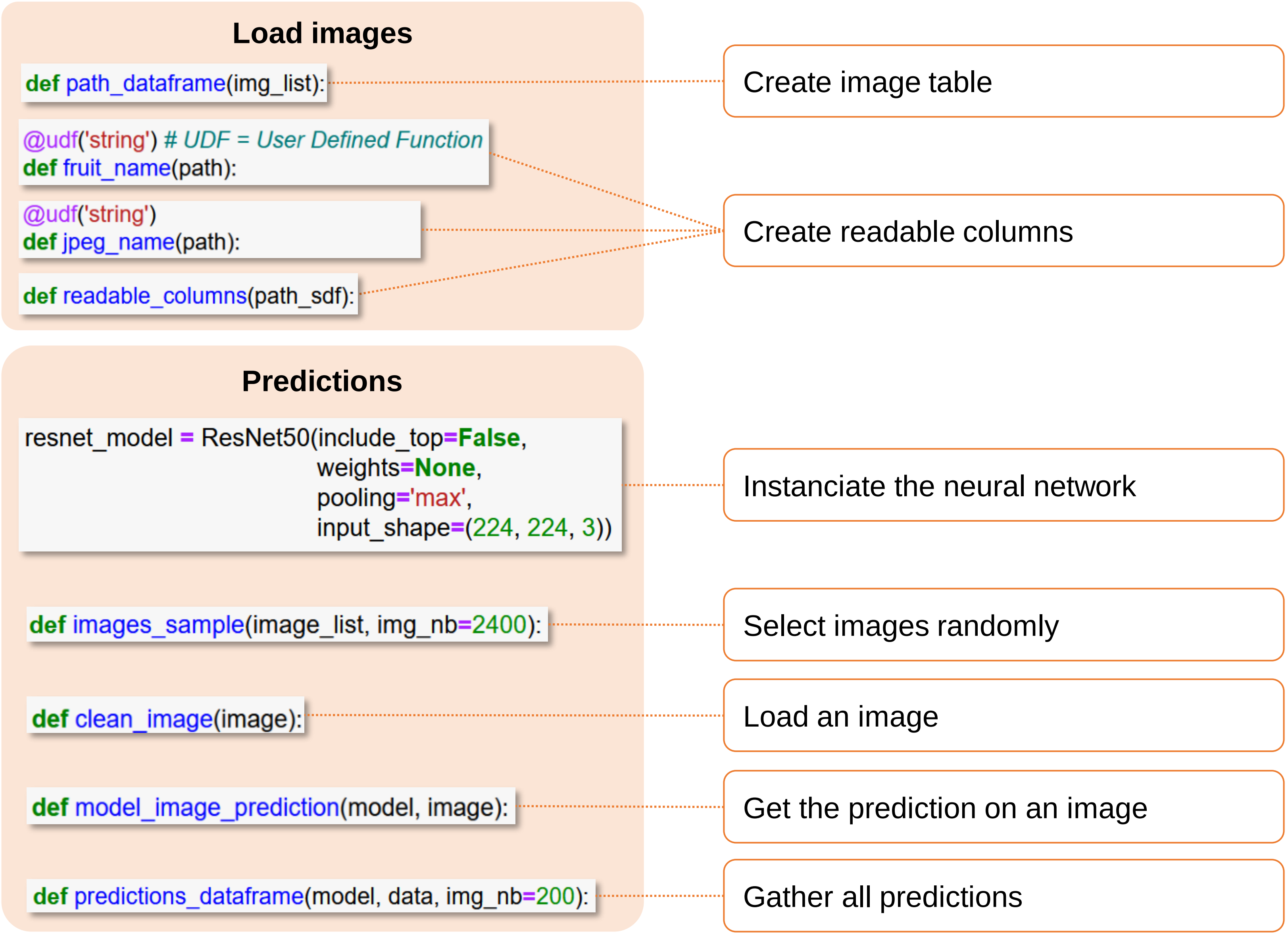
fruit	jpeg	image
apple_6	r0_0.jpeg	
apple_6	r0_2.jpeg	
…	…	…
apple_braeburn_1	r0_0.jpeg	
apple_braeburn_1	r0_2.jpeg	
…	…	…

≈ 400 x 400 pixels RGB

→ **Goal**: extract a table that can be used by an algorithm

III. Image processing with Spark

4.1) Image processing steps



Load images

```python
def path_dataframe(img_list):
```

Create image table

```python
@udf('string') # UDF = User Defined Function
def fruit_name(path):
```

```python
@udf('string')
def jpeg_name(path):
```

```python
def readable_columns(path_sdf):
```

Create readable columns

Predictions

```python
resnet_model = ResNet50(include_top=False,
                        weights=None,
                        pooling='max',
                        input_shape=(224, 224, 3))
```

Instanciate the neural network

```python
def images_sample(image_list, img_nb=2400):
```

Select images randomly

```python
def clean_image(image):
```

Load an image

```python
def model_image_prediction(model, image):
```

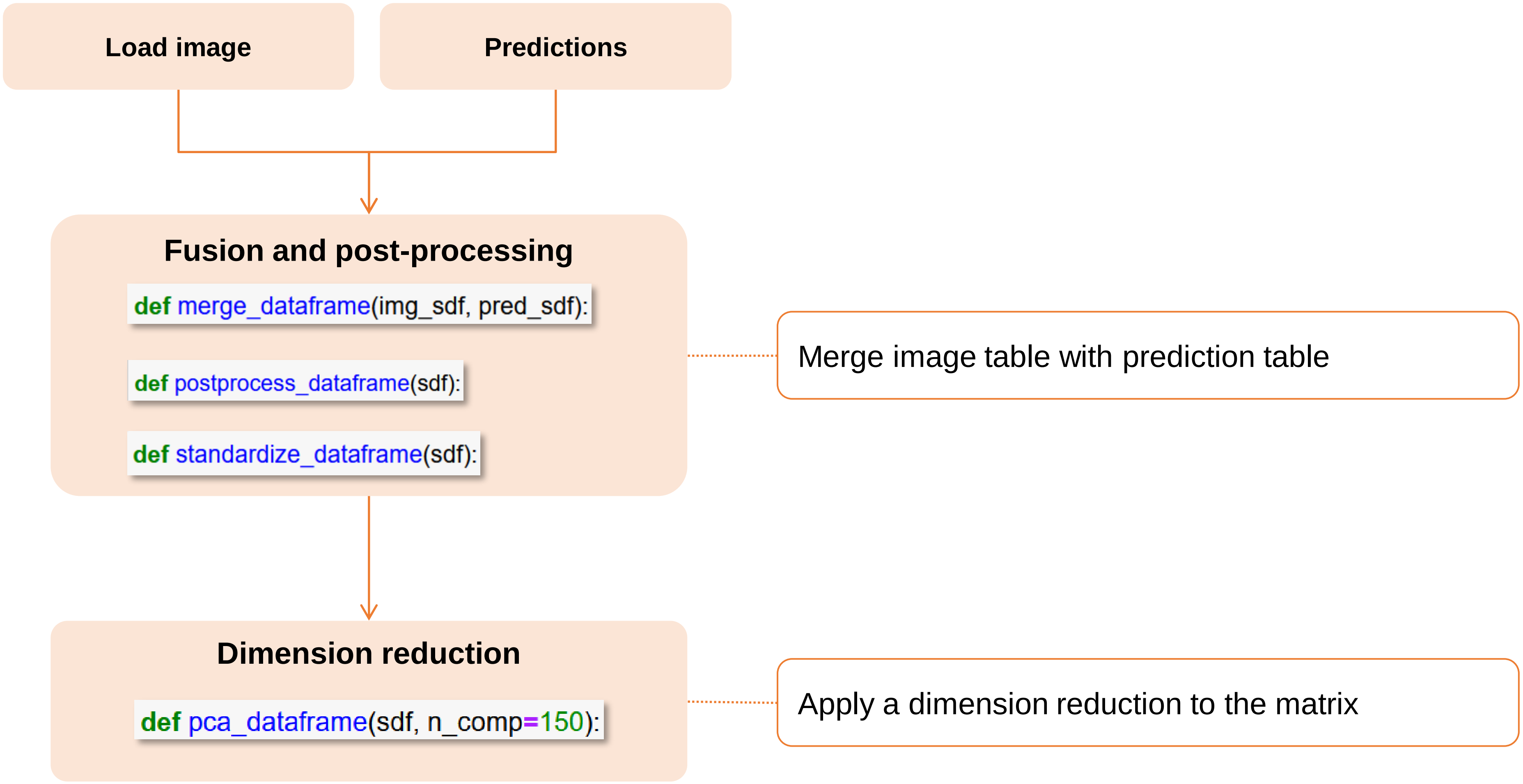
Get the prediction on an image

```python
def predictions_dataframe(model, data, img_nb=200):
```

Gather all predictions

III. Image processing with Spark

4.2) Image processing steps



Load image

Predictions

Fusion and post-processing

```python
def merge_dataframe(img_sdf, pred_sdf):
```

```python
def postprocess_dataframe(sdf):
```

```python
def standardize_dataframe(sdf):
```

Merge image table with prediction table

Dimension reduction

```python
def pca_dataframe(sdf, n_comp=150):
```

Apply a dimension reduction to the matrix

→ How many components for the reduced matrix ?

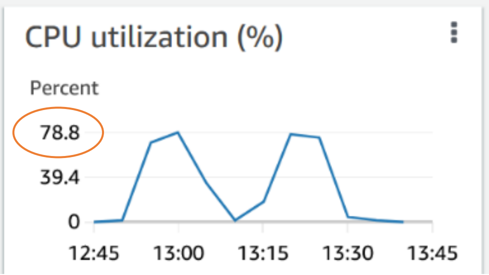
III. Image processing with Spark

5) PCA dimension reduction

→ Weights of components vary with the number of images:

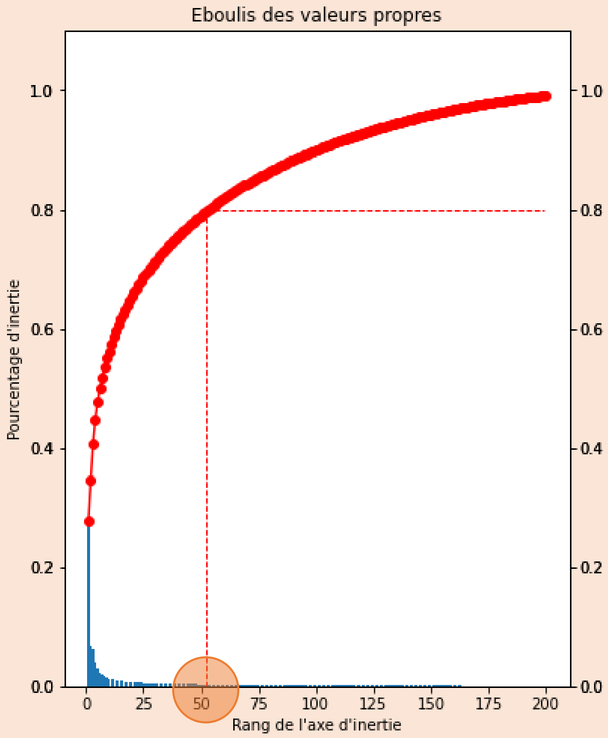
→ These graphs vary with the randomness of the `random.choices` method

→ Server utilization for 2400 images:



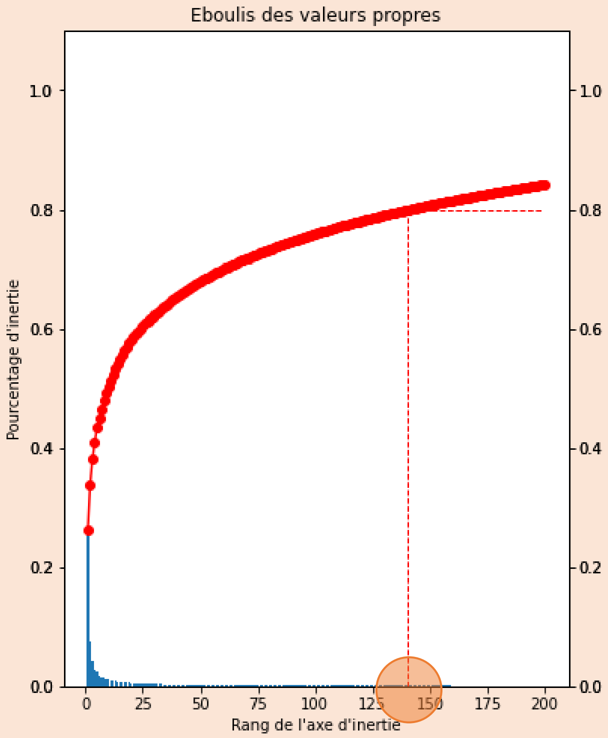
240 images (10 / fruit)

→ 80% in ≈ 50 components



2400 images (100 / fruit)

→ 80% in ≈ 140 components



→ As we increase the number of images processed, the 80% seem to converge to **150** principal components.

IV. Conclusion

1) Summary and outlooks

Dataset

- Ready for analysis in its current form.
- Growth is expected, so there is a need for **cloud** support (distributed computing)
- Majority of apple species (15 on 24 fruits): risk of bias.

BigData environnement

- Information exchanges take place between users, EC2 instance and S3 buckets.
- Instance **RAM** is important to be able to use Spark without trouble.
- **Version compatibility** is determining

Image processing with Spark

- Few images: need for a transfer learning with ResNet50 of Keras
- Distributed computations are done through **PySpark own methods** or **decorated functions**.
- PCA dimension reduction seems to be optimal with **150** components.

Outlooks

- Do parallelization of computations on a real EMR cluster.
- Diversify the fruits
- Consider other cloud solutions: Azure, Google Cloud, OVHCloud, …
- Capture the images in equal proportions according to the rotation axes.
- Feature augmentation

IV. Conclusion

2) Recommandations for scaling

Quality criteria of a cloud tool

1. Simple scaling
2. Simple maintenance
3. Simple data exploitation

AWS 5 pillars

1. Operational excellence
2. Security
3. Reliability
4. Performance efficiency
5. Cost optimization

End of the presentation

Thank you for your attention