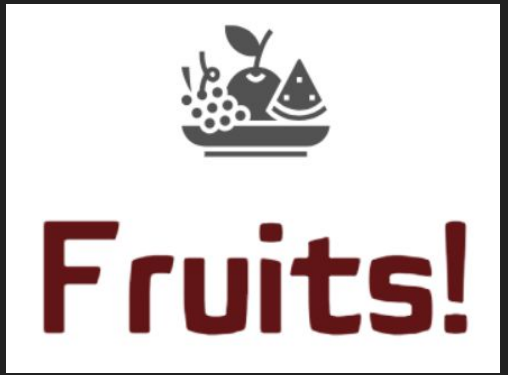
Cloud computing

Julien Le Boucher

07-28-2023

Context



- We "Fruits!" want to develop fruit harvesting robots.
- Beforehand, an intermediate goal is set :
  - build an application to raise public awareness about fruits diversity and use it to improve the vision tool which will be used to pilote robots.

Missions

- Prepare an environment (AWS EMR / IAM / S3 & Spark) to ensure massive storage and data processing in the cloud.
- Design a first draft pipeline to process images (feature extraction + dimensionality reduction)

The dataset : "Fruits 360" `Version: 2020.05.18.0`

- ~90 000 images from 131 distinct classes of fruits (100 x 100 pixels).
- reduce to a subset of 320 images (for time and costs motivation)

A kumquat example

The subset organization :

- A main folder.
- 32 subfolders – for each selected class of fruit.
- 10 images randomly selected within the class.

```
images_subset
├── Apple Golden 2
│   ├── 20_100.jpg
│   ├── 284_100.jpg
│   ├── 313_100.jpg
│   ├── 320_100.jpg
│   ├── r_170_100.jpg
│   ├── r_187_100.jpg
│   ├── r_238_100.jpg
│   ├── r_289_100.jpg
│   ├── r_305_100.jpg
│   └── r_318_100.jpg
├── Apple Pink Lady
│   ├── 121_100.jpg
│   ├── 58_100.jpg
│   ├── 9_100.jpg
│   ├── r_162_100.jpg
│   ├── r_167_100.jpg
│   ├── r_188_100.jpg
│   ├── r_195_100.jpg
│   ├── r_298_100.jpg
│   ├── r_301_100.jpg
│   └── r_43_100.jpg
├── Apple Red 1
│   ├── 156_100.jpg
│   ├── 205_100.jpg
│   ├── 254_100.jpg
```

Plan

1. Big Data Challenges and solutions
2. Setting up AWS
3. Image processing flow on the cluster
4. Conclusions

1 - Big Data challenges and solutions

Big data challenges

Data Storage

- scalability (virtually illimited)
- fast-access for processing
- manage access policy (security for private information)
- back-up

Computation Power to process the data.

2 strategies :

- increase the power of CPU/GPU/TPU (vertical scalability)
- increase the number of CPU/GPU/TPU and parallelize computations (horizontal scalability).

Big data challenges

SOLUTIONS

Data Storage

- scalability (virtually illimited)
- fast-access for processing
- manage access policy (security for private information)
- back-up

→ AWS S3 (Amazon Simple Storage Service) + IAM (Identity and Access Management)

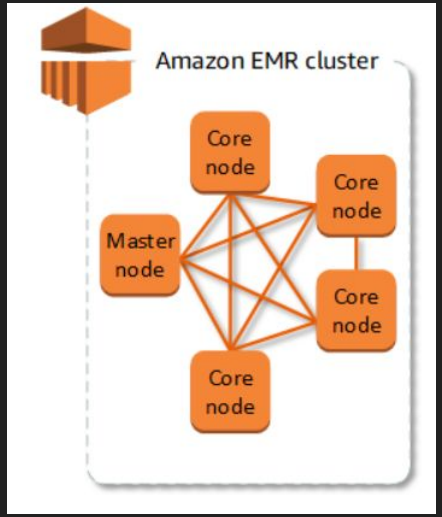
Computation Power to process the data.

2 strategies :

- increase the power of CPU/GPU/TPU (vertical scalability)
- increase the number of CPU/GPU/TPU and parallelize computations (horizontal scalability).

→ Apache Spark is an open-framework that programmatically enables horizontal scalability. It is based on the "Map Reduce principle" discovered @ google which ensures that algorithms can be parallelized on a cluster of many machines thanks to elementary operations such as map and reduce.

Chosen strategy



Horizontal scalability :

Rent a cluster of machines in the cloud and manage it with AWS EMR (Elastic MapReduce)

EMR helps managing a cluster of machines by :

- assisting the nodes configuration
  - OS + software installations + compatibility between versions
- allowing scalability and flexibility
  - add or remove nodes (EC2 instances) in the cluster to scale accordingly to the needs
  - mix with cheaper instances (Spot) to compute certain type of tasks
- ensuring efficient data flows (I/O)
  - compatibility with S3.

Note : the EMR service add an extra cost to the EC2 instances and to the S3 (13% during the tests). Nevertheless, EMR has an option which could help a better management of the dynamic behavior of the cluster which could compensate its cost when the application becomes popular.

2 - Setting up AWS

1. Users accounts (IAM)
2. S3
3. EMR

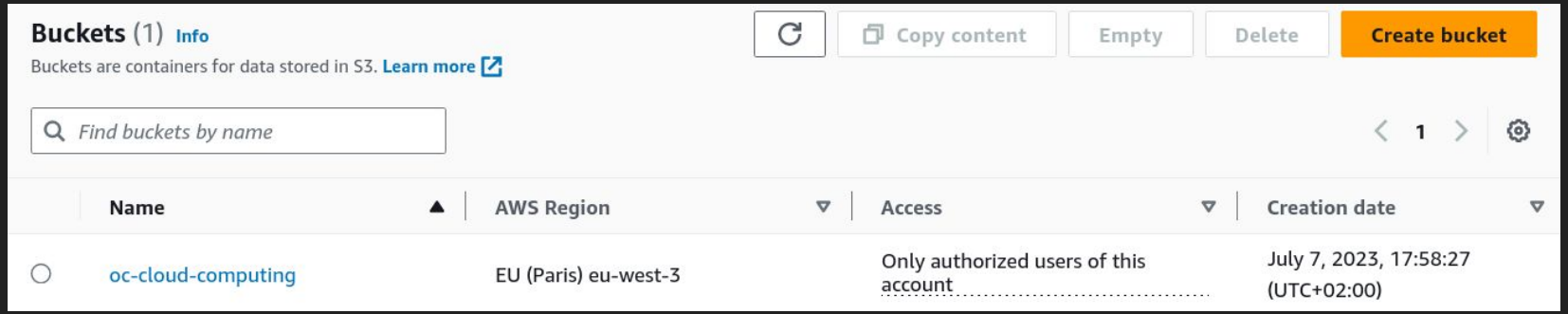
AWS accounts (IAM)

1. Root account creation and connection
   - All accesses
2. Administrative user creation
   - Activation of an AWS organization
   - Activation of the IAM Identity Center in which :
     - creation of an user called 'admin1'
     - attachment of the **'AdministratorAccess'** permission set to 'admin1'
3. Connection to AWS with 'admin1'

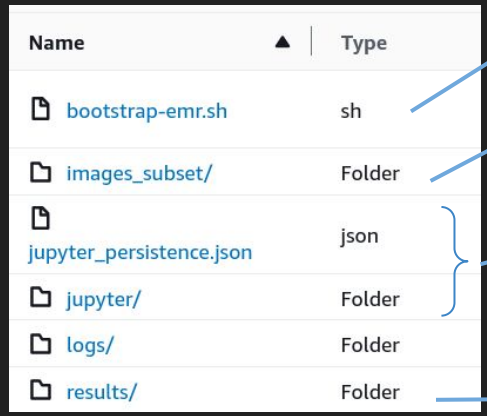
Note : The root account should not be used on daily basis.
All the following screenshots come from the administrative user account.

S3

Creation of a bucket in the parisian servers (for GDPR reasons)



Content



A script to pip install packages on the master and the executors nodes during the EC2 instanciations.

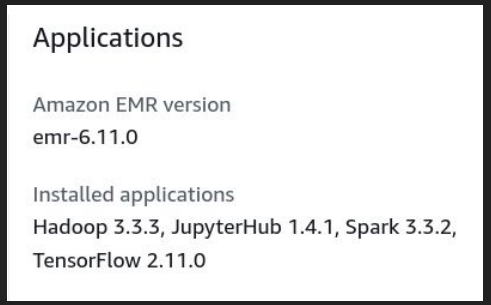
Data Folder with 320 images.

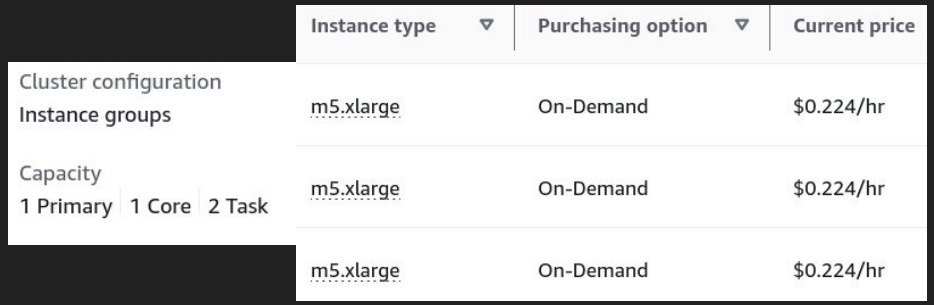
Configuration file and notebook zone to ensure persistence when closing EC2 instances.

Folder to write the results of the image processing to.

EMR configuration

1. SSH key pair creation to allow access to the master node (and the Jupyter and spark applications).
2. Add an inbound rule for SSH access via my computer IP address to the default ElasticMapReduce-master security group.
3. Configuration of the EC2 instances through the EMR console :





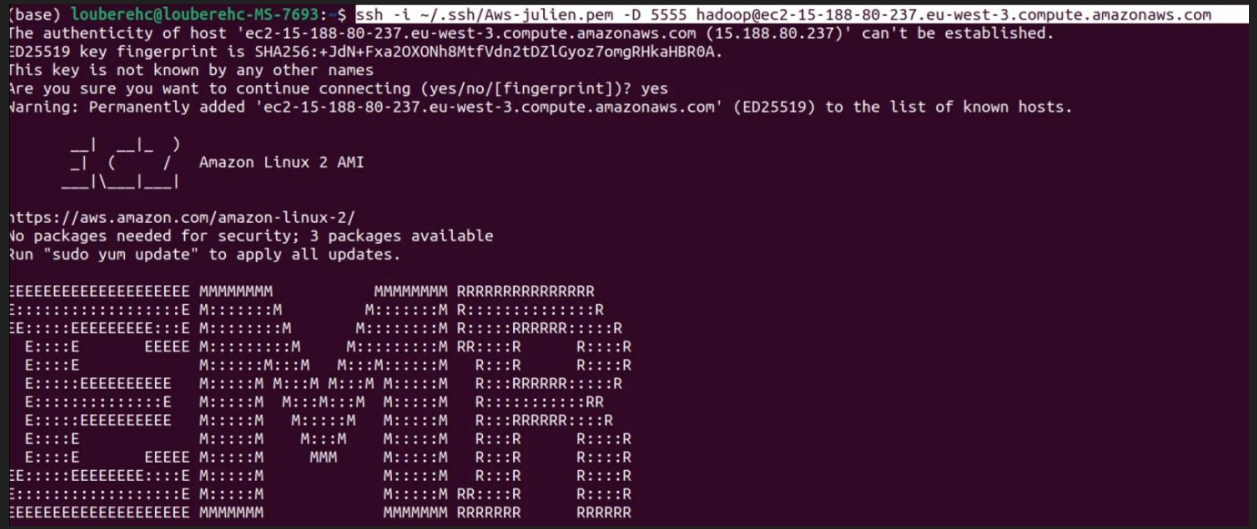
4. Extra configuration with a script and a configuration file

```
#!/bin/bash
sudo python3 -m pip install -U setuptools
sudo python3 -m pip install -U pip
sudo python3 -m pip install keras==2.11.0
sudo python3 -m pip install wheel
sudo python3 -m pip install pillow
sudo python3 -m pip install pandas
sudo python3 -m pip install pyarrow
sudo python3 -m pip install boto3
sudo python3 -m pip install s3fs
sudo python3 -m pip install fsspec
```

```
[
    {
        "classification":"jupyter-s3-conf",
        "properties":
            {
                "s3.persistence.bucket":"oc-cloud-computing",
                "s3.persistence.enabled":"true"
            }
    }
]
```

Run the spark application inside a notebook on the master node

1. Connect to the master node with ssh on the 5555 port :



2. Use a proxy to access the jupyter application and upload the notebook.

3 - Image processing with Spark and MLlib on a cluster

What is MLlib ?

"MLlib is a scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction…" from *Databricks.com*

For example:

- **PCA** and **standard scaler** are straightforward available and efficient tools in MLlib.
- Feature extraction with MobiletNetV2 is a specific task which is not available.

How to leverage Spark power for the feature extraction ?

"you are not limited to just using MLlib for making batch or streaming predictions with Spark—you can create custom functions to apply your pretrained models at scale, known as user-defined functions (UDFs) [...]
If you define your own UDF to apply a model to each record of your DataFrame in Python, opt for pandas UDFs for optimized serialization and deserialization" from *learning Spark (2nd edition)*

The Spark image processing chain.

1. Load images in a spark dataframe from S3:
   - path + binary content + label (extracted from the path).

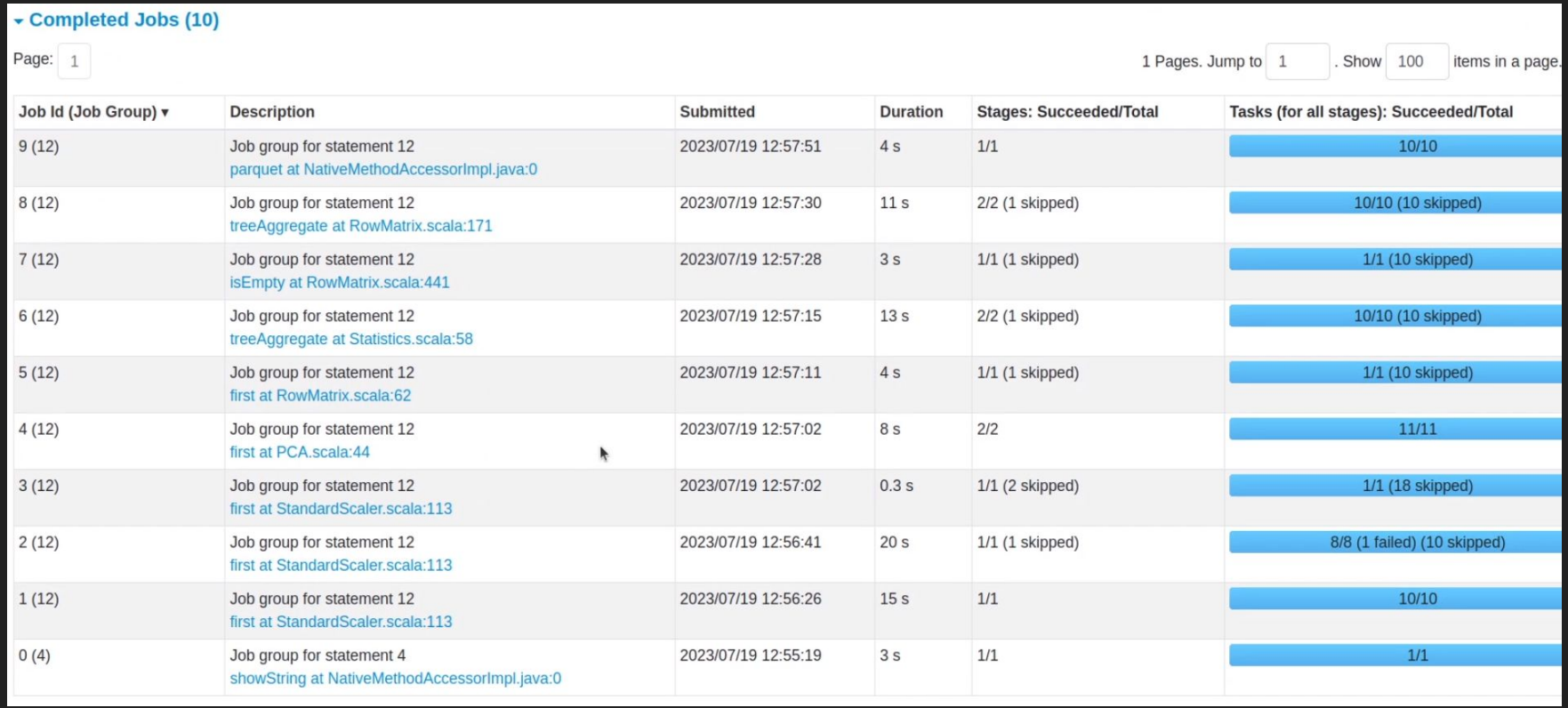
2. Extract features via a pandas_udf (user defined function) which processes the spark dataframe binary 'content' column in a distributed manner.

   Steps of the function:
   - load a MobilenetV2 model without its last layer on each executor node with broadcasted weights.
     Note : the model is loaded once and for all the batches to come thanks to the pandas iterator.
   - iterate over data batches (as if they were pd.Series : API), and, per batch:
     i. preprocess:
        - read the binary content as a Pillow Image and resize it : (100 x 100) → (224 x 224)
        - cast to a numpy array and map the pixels' values between -1 and 1.
     ii. extract the features thanks to the model. (1280,)

3. Reduce the dimensionality with a MLlib pipeline
   - transform arrays to a vector format.
   - apply a standard scaler followed by a PCA with 300 components.

4. Write results into parquet files on s3

Spark UI (1)



▾ Completed Jobs (10)

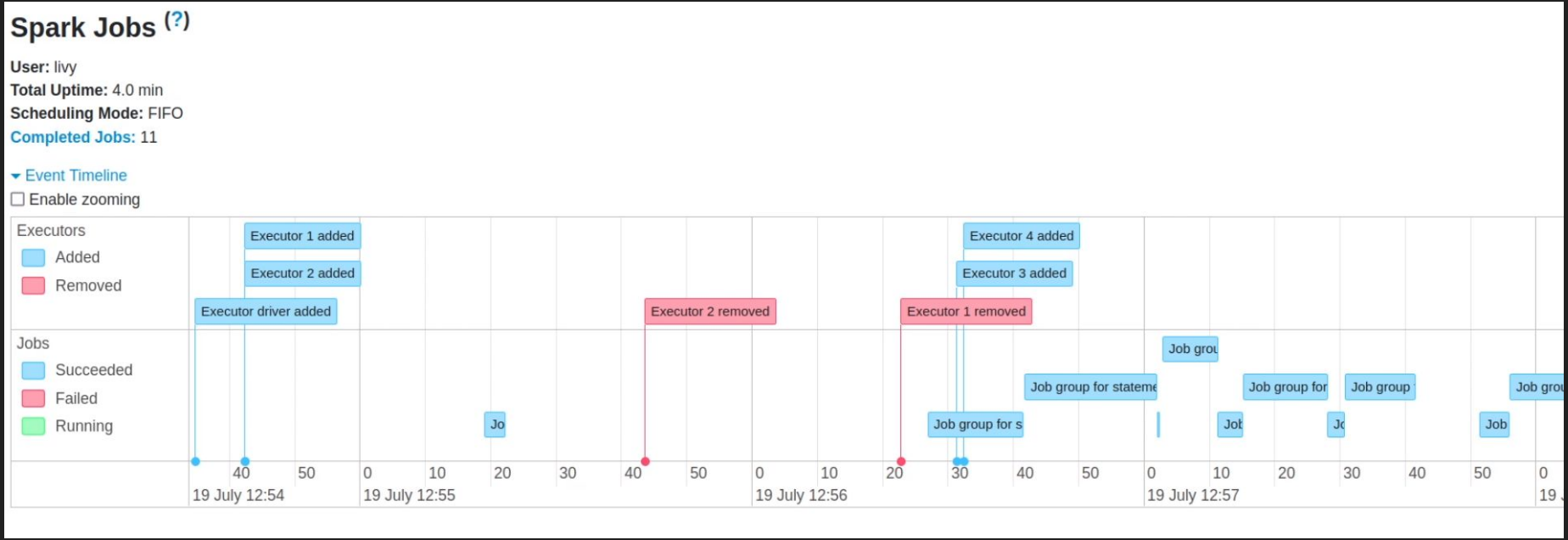
Page: 1

1 Pages. Jump to 1 . Show 100 items in a page.

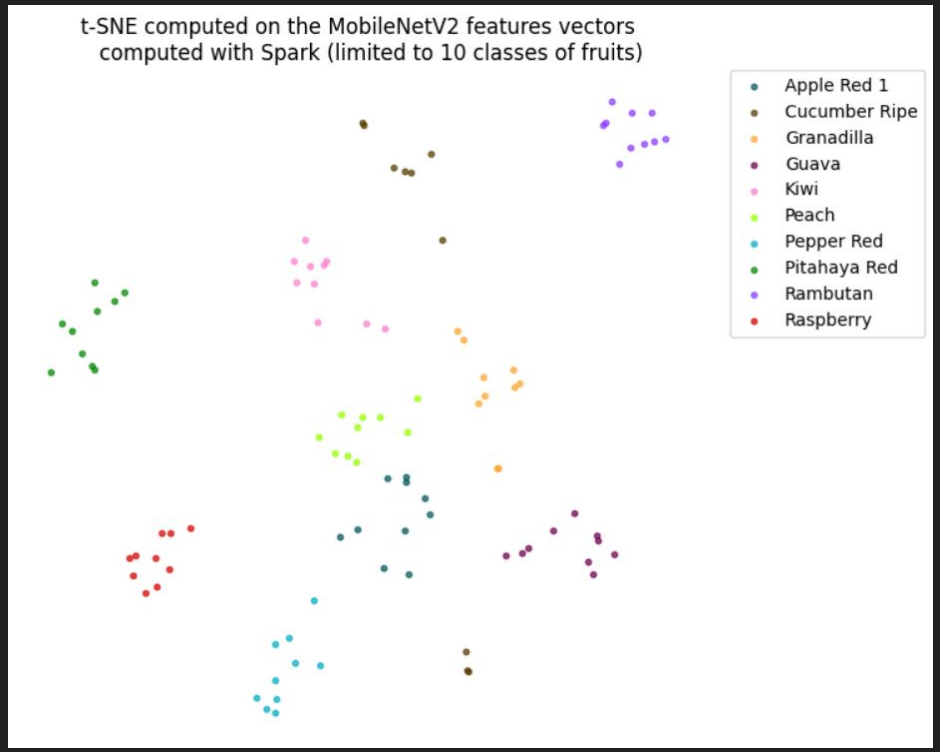
Job Id (Job Group) ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
9 (12)	Job group for statement 12 parquet at NativeMethodAccessorImpl.java:0	2023/07/19 12:57:51	4 s	1/1	10/10
8 (12)	Job group for statement 12 treeAggregate at RowMatrix.scala:171	2023/07/19 12:57:30	11 s	2/2 (1 skipped)	10/10 (10 skipped)
7 (12)	Job group for statement 12 isEmpty at RowMatrix.scala:441	2023/07/19 12:57:28	3 s	1/1 (1 skipped)	1/1 (10 skipped)
6 (12)	Job group for statement 12 treeAggregate at Statistics.scala:58	2023/07/19 12:57:15	13 s	2/2 (1 skipped)	10/10 (10 skipped)
5 (12)	Job group for statement 12 first at RowMatrix.scala:62	2023/07/19 12:57:11	4 s	1/1 (1 skipped)	1/1 (10 skipped)
4 (12)	Job group for statement 12 first at PCA.scala:44	2023/07/19 12:57:02	8 s	2/2	11/11
3 (12)	Job group for statement 12 first at StandardScaler.scala:113	2023/07/19 12:57:02	0.3 s	1/1 (2 skipped)	1/1 (18 skipped)
2 (12)	Job group for statement 12 first at StandardScaler.scala:113	2023/07/19 12:56:41	20 s	1/1 (1 skipped)	8/8 (1 failed) (10 skipped)
1 (12)	Job group for statement 12 first at StandardScaler.scala:113	2023/07/19 12:56:26	15 s	1/1	10/10
0 (4)	Job group for statement 4 showString at NativeMethodAccessorImpl.java:0	2023/07/19 12:55:19	3 s	1/1	1/1

Total duration to process the 320 images and write results to s3 : 95s

Spark UI (2)


Spark Jobs (?)
User: livy
Total Uptime: 4.0 min
Scheduling Mode: FIFO
Completed Jobs: 11
Event Timeline
Enable zooming
Executors
Added
Removed
Executor 1 added
Executor 2 added
Executor driver added
Executor 2 removed
Executor 4 added
Executor 3 added
Executor 1 removed
Jobs
Succeeded
Failed
Running
Job group for stateme
Job group for
Job group
Job group for s
19 July 12:54
19 July 12:55
19 July 12:56
19 July 12:57

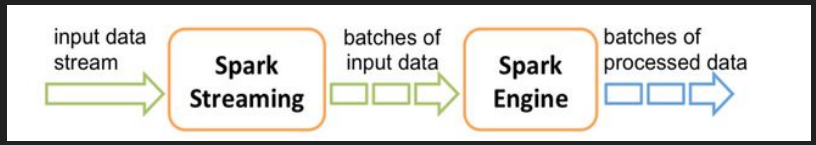
Partial visualizations



The image processing discriminates classes : good.

4 - Conclusions

- The image processing flow is operational in the cloud.
- EMR + Spark + S3 advantages :
  - Scalability (adding/removing EC2 instances.)
    → investigate mixing the use of spot/on-demand instances (cost optimization)
  - Fault-tolerant
  - Streaming option with Spark
    (adapted to app users requests)



- Cost of the simple cluster deployed : ~0.8$ / hour → ~600$ per month if permanent.
- Spark processing duration : 95s / python 9s (not efficient for small dataset : 320 images)
  → Clusters are not efficient until the traffic is not saturating a PC.
- Need specific skills, trial and errors to optimize the EMR configuration and minimize costs
  → investigate EMR serverless which would manage the flexibility automatically.

Thank you for listening