05/02/2021
P8 Datascientist – OpenClassrooms
Etienne Lardeur
Mentor : **Xavier Tizon**
Evaluateur : **Julien Heiduck**

Solutions innovantes pour la récolte des fruits
Le robot cueilleur intelligent

*Développer dans un environnement Big Data, une chaîne de traitement d'images incluant **preprocessing** et **réduction de dimension***

Sommaire

- Introduction **5'**
  - Contexte, jeu de données, use case **-> slides 3, 4 et 5**
- Dispositif proposé et rôle de chaque brique **8'**
  - Architecture: base, dispositifs en Local ou Cloud **-> slides 6 et 7**
  - Illustration Spark UI **-> slides 8**
- Chaînes de traitement **7'**
  - Prototype, transposition Cloud, complémentarité **-> slides 9 et 10**
- Conclusion, Recommandations **5'** **-> slides 11 et 12**
- Question-réponses

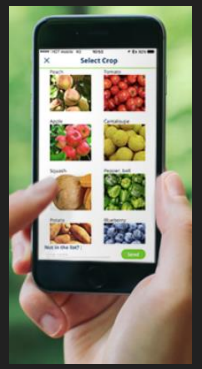
https://github.com/EtienneLardeur/P8_FruitsRecognition

Contexte

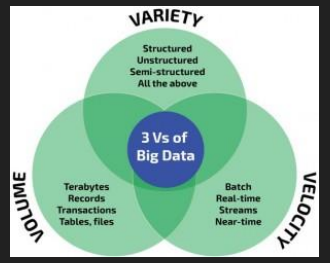
- **Finalité :** robots cueilleurs intelligents

  - **Antériorité** (projets Magali (1985) et Citrus (1990) :
    - Enjeux : remédier à la pénurie de main d'œuvre (**Saisonnalité & Savoir-faire** ⚔ **Productivité**) *[1]*
    - Principaux verrous progressivement levés : performance caméras, ***puissance de calcul*** et avènement du GPS.
  - **Actualité** : investissements R&D, enjeux de **P**ropriété **I**ntellectuelle
    - Tevel aerobotics (Isrl), Robocrops (UK), Octinion (Blg), Airsprid (Fr), …

- **Première étape** : populariser Fruits! via une App mobile de reconnaissance de fruits



  - **Justifie** une architecture Big Data (modèle des 3 « V » *[2]*)
    - ***Volume*** : f(données labellisées, variétés, stades de développement, nouvelles données)
      - *initial* : **1Mo/i**, *pre-process* : **10ko/i**, soit proto en **Go** et usage en **PétaOctet**
    - ***Vitesse*** : collecte et partage de données, puissance de calcul, latence à minimiser.
    - ***Variété*** (sources et structures de données) :
      - environnement : imagerie, géolocalisation, capteurs…
      - bases de données tierces : ex, bio-agresseurs et auxiliaires *[3]*



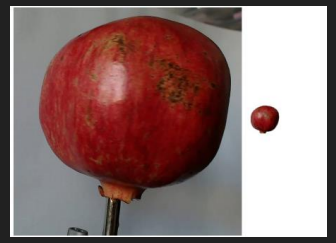
Les 3 "Vs"

- *Et voici comment le **Big Data** investit le « champ » de l'**Arboriculture***

[1] itw P. Grenier, Technique de l'Ingénieur, 09/2020
[2] Big Data – University of Granada (research group)
[3] Bases de données ecophytopic.fr

Jeu de données

- **Dataset Kaggle** *[1]* **riche de 131 variétés de fruit et legumes -** ***labellisées***



  - Photos "360°" extraites d'une captation spécifique :
    - rotation tri-axiale
    - post-traitée (fond blanc reconstruit + resizing 100x100 pixel)
  - Train : 67 692 Fichiers / Test : 22 688 Fichiers

- **Intérêt : focus sur feature extraction et stratégies de classification**
- **Limites : procédé initial lourd et non représentatif**
  - Diversités d'aspects, formes et couleurs f(croissance et maturité)
  - Implique l'ajout d'un preprocessing 'conditions réelles' : cropping & background-removal, resizing

- **Opportunité :** enrichissement des données en conditions réelles, partiellement labellisées

[1] https://www.kaggle.com/moltean/fruits

Use case

- Application

Feature : chaîne de traitement

Collect
- image capture
- *Details collection*
- *Right labels*

Store Data
- image upload
- *Upstream preprocessing*
- *Details storage (enrich schema)*

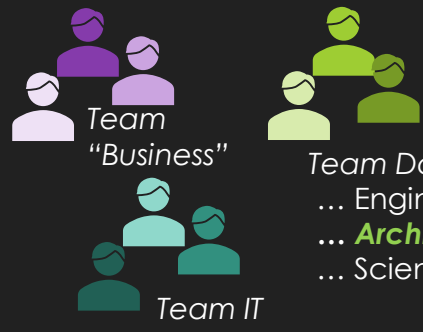
preprocess & reduce

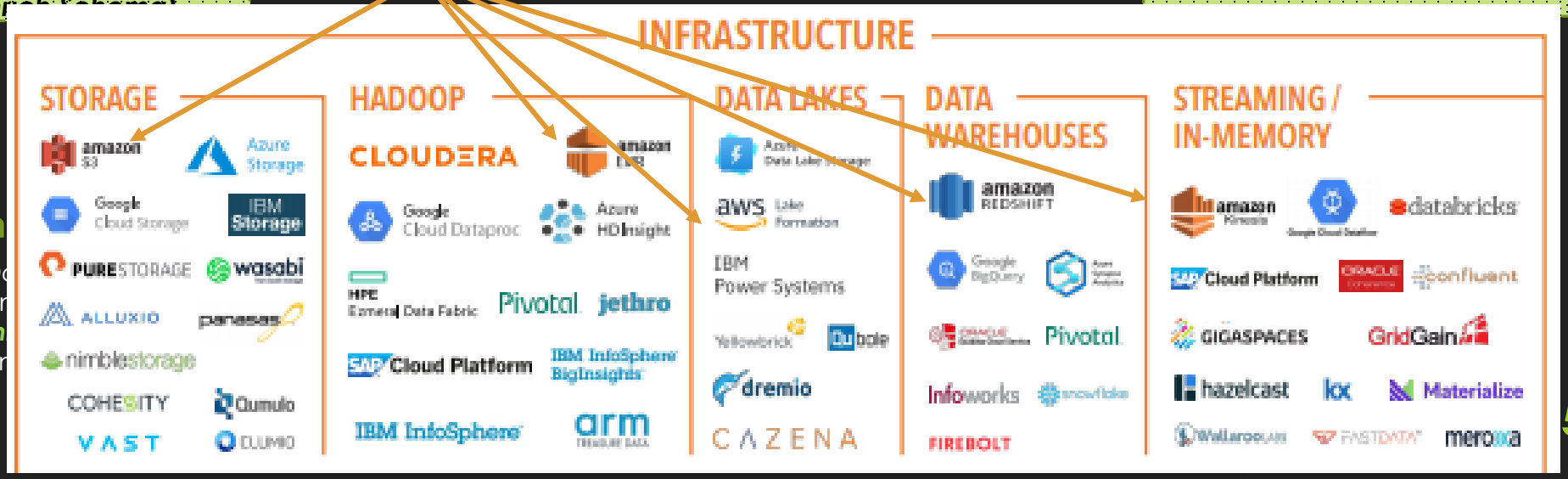
aws *everywhere*

Store results

Classify
- Return labels
- *Attach details*
- *Catch new labels or details*

- Rôles / compétences

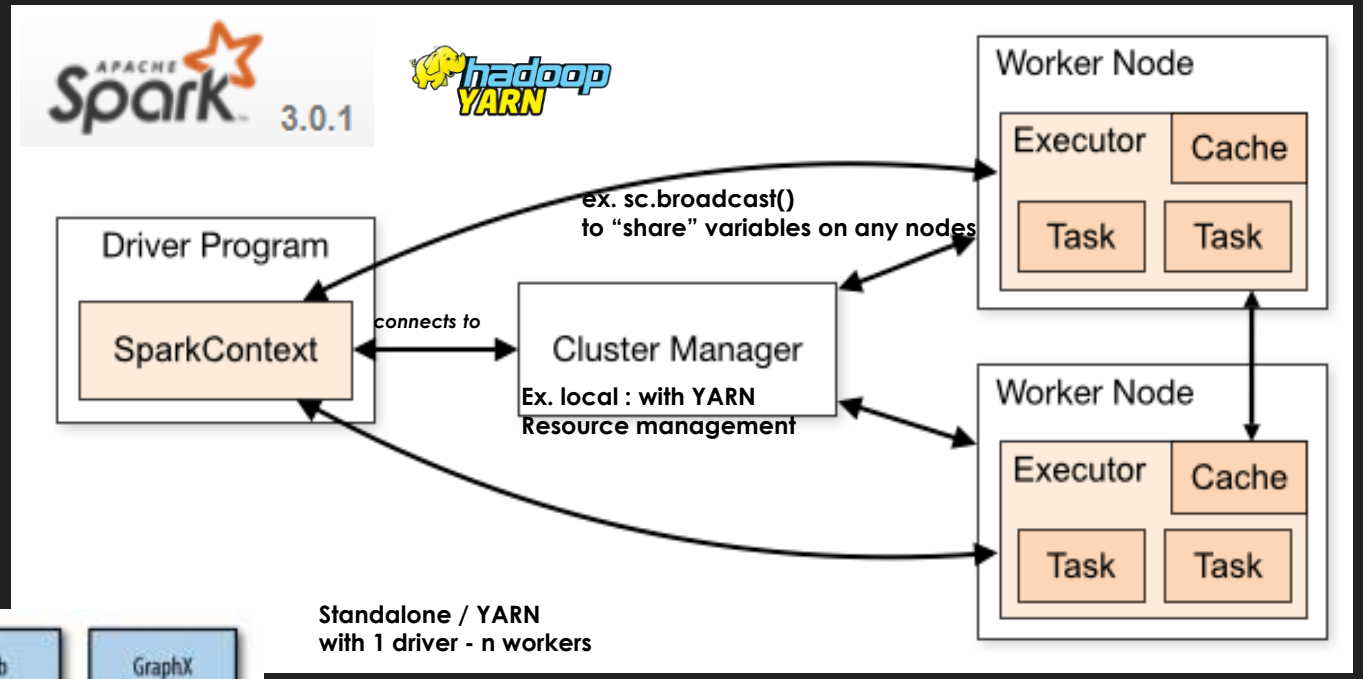




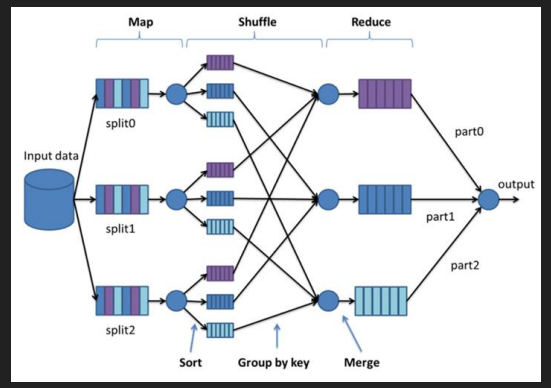
[1] FirstMark : 2020/09/2020-Data-and-AI-Landscape
[2] exemple positionnement de la société Saagie

Architecture, base

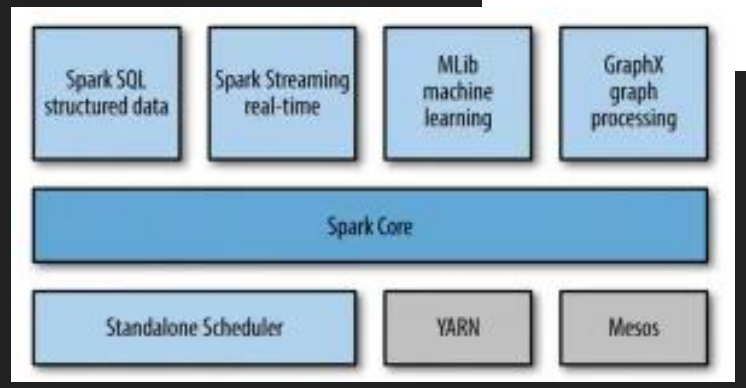
- **Fondements : ecosystem Hadoop + framework Spark**
  - Base Hadoop map/reduce + traitement "in memory"
  - Base Resilient Distributed Datasets + Spark DataFrame
    - Assurant la **Parallélisation** des operations
    - **Résilient** au moyen de graphes acycliques orientés (d'où la tolérance au pannes)
  - "Lazy evaluation" (Transformation vs Action)
  - Performance conditionnée par la stratégie de partitioning *[1]*



Cluter : worker-nodes



Map/Reduce



Features

[1] Medium: On Spark Performance and partitioning strategies

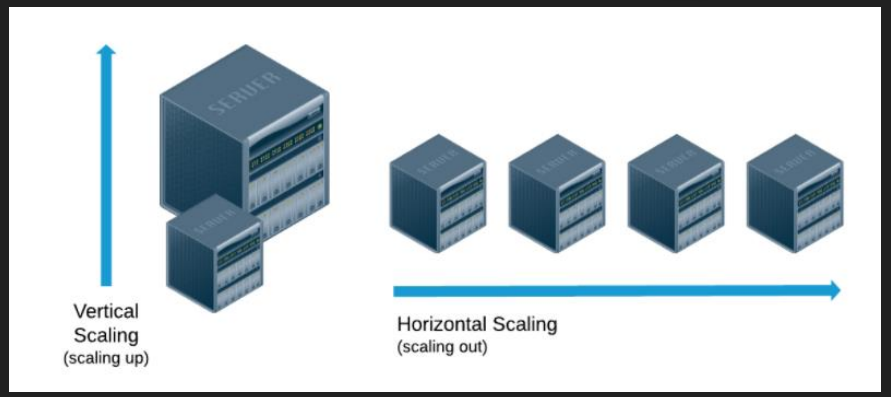
Architecture, dispositifs

- **Dispositif Local (prototype) :** ***montée en compétence ok***
  - script pySpark sur linux : ubuntu WSL (yc *"tunnelisation" ssh*)

! Java 1.8

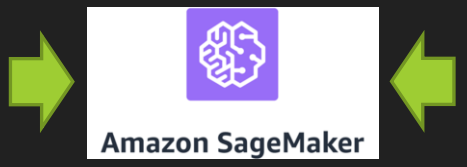
- **Dispositifs Cloud possible pour assurer la scalabilité :** ***montée en compétence en cours***

  - Scaling Vertical versus Horizontal *[1]*



+ L'écosystème AWS:
  - "Servitudes" d'architecture :
    - Solutions de stockage (S3) + Gestion de permissions (IAM)
  - Alternatives calcul :
    - Puissance de calcul (dimension "fixe") (EC2)
    - Cluster management – logique Spark workers-nodes (EMR- n EC2s)
    - Environnement de développement (ex. Jupyter Notebooks)
  - Et au delà…

+ autres solutions **PAAS**…

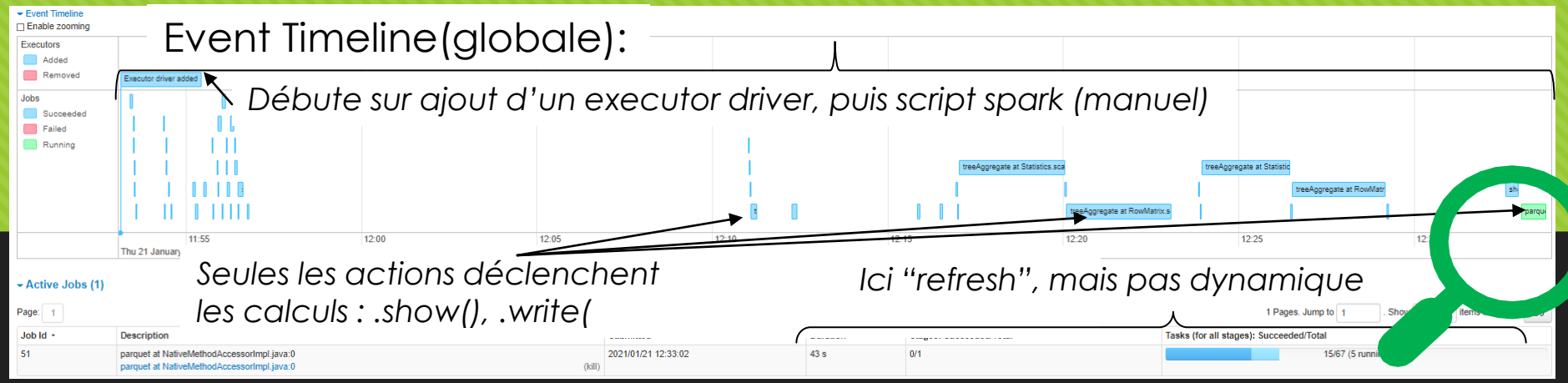


[1] Scaling Horizontally vs. Scaling Vertically

Spark UI

- **Outil de monitoring** disponible sur un port en complement
- **Enregistre** le log (py4j)
- Instructif mais empilage des technologies mises en oeuvre **déroutant**

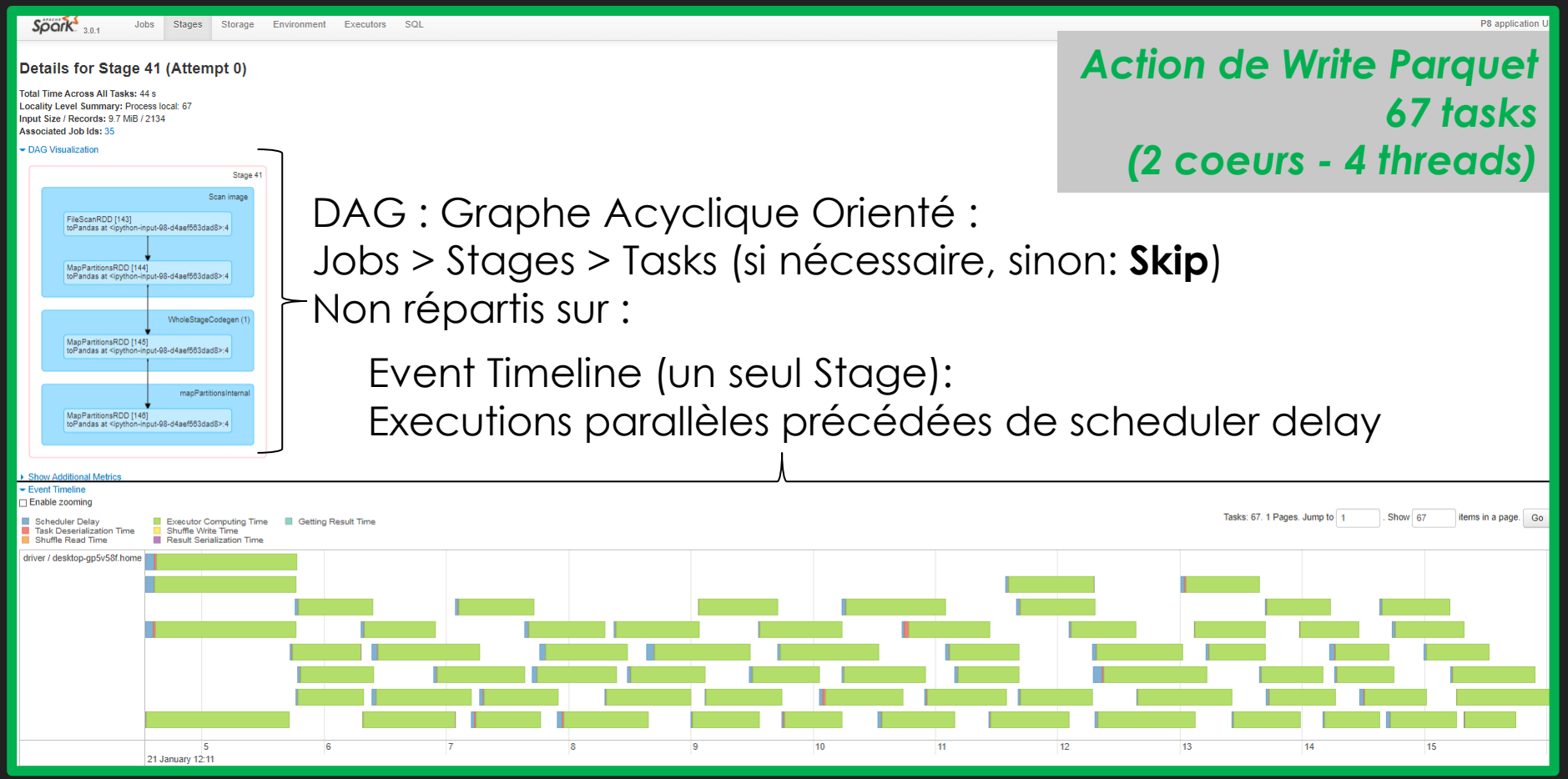
Event Timeline(globale):



Débute sur ajout d'un executor driver, puis script spark (manuel)

Seules les actions déclenchent les calculs : .show(), .write(

Ici "refresh", mais pas dynamique



Action de Write Parquet
67 tasks
(2 coeurs - 4 threads)

DAG : Graphe Acyclique Orienté :
Jobs > Stages > Tasks (si nécessaire, sinon: **Skip**)
Non répartis sur :

Event Timeline (un seul Stage):
Executions parallèles précédées de scheduler delay

Chaine de traitement, pySpark local

- Transformations : non évaluées
- Actions : conserver/introduire les actions necessaires

persistence
.cache()

une fois :
Nb optimal de PCs

spark.SparkSession → *spark.stop()*

Collect → **Store data** → **Load data** → ***Featurize*** → ***Reduce*** → **Store results** (ARROW, Parquet) → **Classify**

Spark DataFrame :

spark.read.format('image')
.load(work_path, inferschema=True)

Extract Label :

.withColumn('label', element_at(
split(df_img['image.origin'], "/"), -2))

```
root
 |-- image: struct (nullable = true)
 |    |-- origin: string (nullable = true)
 |    |-- height: integer (nullable = true)
 |    |-- width: integer (nullable = true)
 |    |-- nChannels: integer (nullable = true)
 |    |-- mode: integer (nullable = true)
 |    |-- data: binary (nullable = true)
 |-- label: string (nullable = true)
```

"Capture" des métadonnées (flexible)

Load image
(pandas_udf scalar iterator)

CNN predict
(vgg16)

PCA
spark.ml

array_to_vector *(udf)*

vector_to_array *(udf)*

Iteration d'une function "get tensor from path"
Attention **imbrication via tensorflow .predict**

image	label	cnn_features
[file:///mnt/c/us...	Apple_Golden_1	[0.19929896295070...
[file:///mnt/c/us...	Apple_Golden_1	[0.16625839471817...
[file:///mnt/c/us...	Apple_Golden_1	[0.17010329663753...

Alternative : *ImageSchema.toNDArray à partir de la structure de données 'image'*
Nb: DeepImageFeaturizer deprecated.

Ecriture puis lecture vers pd.DataFrame au fromat parquet, engine pyarrow

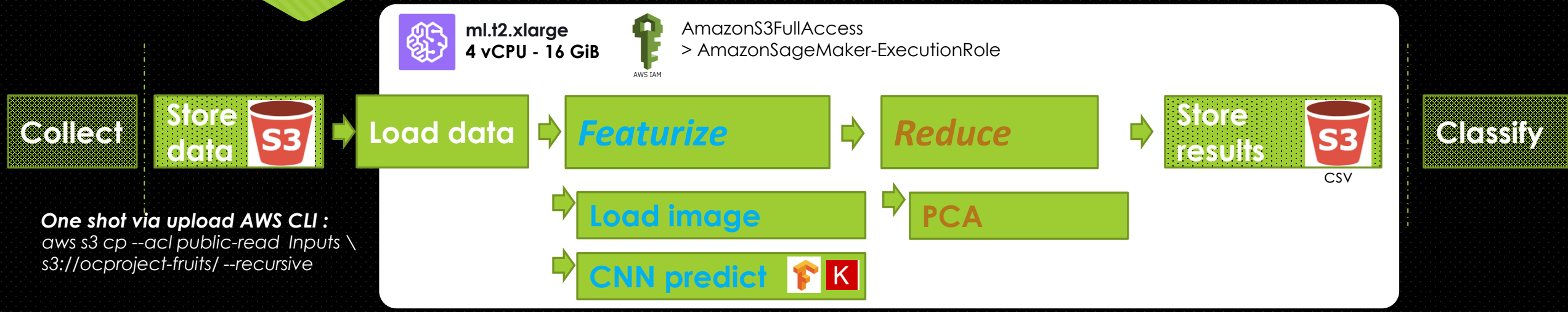
	image	label	features
0	{'origin': 'file:///mnt/c/users/etien/datascie...	Apple_Golden_1	[-6.463369, 1.1861173, -0.948696, -0.30090705,...
1	{'origin': 'file:///mnt/c/users/etien/datascie...	Apple_Golden_1	[-6.493184, 1.1764057, -0.9451786, -0.35384464...
2	{'origin': 'file:///mnt/c/users/etien/datascie...	Apple_Golden_1	[-6.4693136, 1.1791035, -0.9266322, -0.3449764...

Encapsulation array-vector
Attention évaluation du nb de PCs optimal et réalisation du PCA.Fit dans la séquence

res	cnn_vectors	pca_vectors	features
...	[0.19929896295070...	[-6.4633689011531...	[-6.463369, 1.186...
...	[0.16625839471817...	[-6.4931841884080...	[-6.493184, 1.176...
...	[0.17010329663753...	[-6.4693134061408...	[-6.4693136, 1.17...

- **Nombreuses variantes** de séquences possibles pour un même but
  - **Types** et **Schemas** des données, **packages** exploités *(! versions, maintenance)*, codage optimal, …

Chaîne de traitement Cloud



One shot via upload AWS CLI :
*aws s3 cp --acl public-read Inputs *
s3://ocproject-fruits/ --recursive

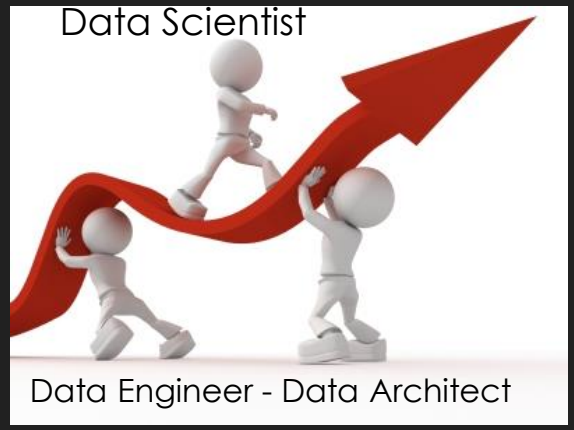
- Gain de performance initial **Spark Local** : **Load** (image) + **Store** (parquet) **impressionnant**
- Fonctionnalité "Spark Cloud" **valide** (volume, variété, vitesse):
  - Fonctionnalité et performance **Load** et **Store** **dégradée** (*disponibilité packages - versions et faisabilité*)
  - Meilleure performance ***Featurize*** (*tensorflow exploite la puissance de calcul, optimisable à état de l'art [1] + "pruning" cnn*)
  - Gains substantiels ***Reduce***

- **Poursuivre** l'instanciation du meilleur assemblage
- Explorer le **Streaming** et recherche de réduction « incrémentale » *[2]*

[1] *Paper with code*
[2] Incremental Sparse-PCA Feature Extraction For Data Streams

Conclusion

- Un feature au sein du projet, au sein de l'approche business de l'entreprise, au sein d'une mutation digitale
- Sensibilisation aux compétences expertes requises:
  - Projets **collaboratifs** aux mains **d'équipes** pluridisciplinaires !



Recommendations

Perspectives techniques (industrialisation)

- Etat de l'art choix technologiques:
  - Transfert learning avec fine tuning pour meilleur accuracy
  - Feature map pruning pour simplification et rapidité
- Code refactoring (selon la technologie : langage **Scala**)
- Scaling vertical vs horizontal: **analyse technico-économique**
- Extension cas réels logique utilisateur (preprocessing)
- Exploitation pour le développement du Robot Cueilleur…

Perspectives orientées business :

- Contractuel & économie de la solution choisie
- Exploiter l'application pour enrichir en dynamique :
  - Labellisation, informations additionnels
  - Utilisateurs ou professionnels
- Augmenter le cycle de vie
  - Cueillette et l'entretien : maturité, pathologies, conseils de taille

Partenariats – collaboration - compétences

- Accélérer la recherche (ex. partenaire universitaire)
- Optimiser la mise en oeuvre (ex. collab. acteur majeur)
- Recours à l'expertise & montée en competence
  - Ex. containerization, ML Ops, …

Questions / réponses

Merci de votre attention !