



IMPLÉMENTATION D'UN
PIPELINE DE
TRAITEMENT DISTRIBUÉ
SUR AWS EMR AVEC
PYSPARK ET
TENSORFLOW

FRUITS!

Jade Suchaud

AGENDA

- | | |
|---|--------------------------------------|
| 1 | Rappel de la problématique |
| 2 | Architecture Cloud et Justifications |
| 3 | Infrastructure et Déploiement |
| 4 | Chaîne de Traitement PySpark |
| 5 | Démonstration et Validation |
| 6 | Conclusion et perspectives |



RAPPEL DE LA PROBLÉMATIQUE



Contexte



Contexte :

- Robots cueilleurs intelligents + App mobile de classification
- Croissance : 22k images → 50M prévues (×2000 en 18 mois)
- Besoin urgent : Valider architecture avant investissement 2M€

Défis techniques :

- Performance : SLA <30 secondes par traitement + scalabilité 0→1000 workers
- Coûts : ROI Cloud élastique vs infrastructure on-premise
- Disponibilité : 99.9% uptime pour application mobile critique

Objectifs projet :

- Valider pipeline Big Data production-ready
- Prouver scalabilité millions d'images
- Démontrer conformité RGPD et optimiser coûts avant scaling



Problématique

Comment traiter efficacement des millions d'images de fruits dans un environnement Big Data scalable et conforme RGPD ?



Jeu de données

Dataset : Kaggle Fruits-360-dataset

The screenshot shows the Kaggle interface for the Fruits-360 dataset. The left sidebar is visible with options like Home, Competitions, Datasets, Models, Benchmarks, Game Arena, Code, Discussions, Learn, and More. The main content area displays the dataset's title, "Fruits-360 dataset", which contains 158396 images of 226 classes. It includes sections for "About Dataset", "Content" (listing various fruit and vegetable names), and "Tags" (Arts and Entertainment, Health, Food, Image, Multiclass Classification, Plants). A preview image of several green fruits is shown.

The screenshot shows the Data Explorer for the dataset. It lists the following directory structure:

```
fruits-360_100x100
├── fruits-360
│   ├── Test
│   ├── Training
│   └── LICENSE
└── README.md

fruits-360_3-body-proble
├── fruits-360-3-body-prol
└── fruits-360_dataset_meta
    ├── fruits-360-meta
    └── fruits-360_multi
        ├── test-multiple_fruits
        │   └── LICENSE
        └── README.md

fruits-360_original-size
└── fruits-360-original-size
```

Volume et Format :

- 22,819 images JPG 100×100 pixels (2.8 GB total)
- Organisation S3 : Structure hiérarchique par dossiers
- Distribution équilibrée : ~174 images par classe

Diversité :

- 131 espèces de fruits et légumes différentes
- Variétés principales : Pommes, Bananes, Oranges, Tomates...
- Qualité : Labels validés, dataset production-ready

Défis Techniques :

- Scalabilité : Architecture doit supporter 50M+ images futures
- Variabilité : Éclairage, maturité, angles de prise de vue
- Performance : Chargement distribué S3 optimisé



ARCHITECTURE CLOUD ET JUSTIFICATIONS



Choix Architecture Cloud

IAAS vs PAAS :

- IAAS (EC2) : Gestion infrastructure manuelle
- PAAS (EMR) : Service managé, focus sur le développement

Choix : Amazon EMR (PAAS)

- Cluster Spark pré-configuré
- Intégration native S3, TensorFlow, JupyterHub
- Monitoring et scaling automatiques
- Arrêt automatique pour contrôle des coûts

Comparaison IAAS vs PAAS

IAAS (EC2)	PAAS (EMR)
Applications	Applications
Runtime	Runtime
Middleware	Middleware
OS	OS
Virtualisation	
Serveurs	Serveurs
Stockage	Stockage
Réseau	Réseau

Utilisateur

AWS



Configuration Technique

Hardware :

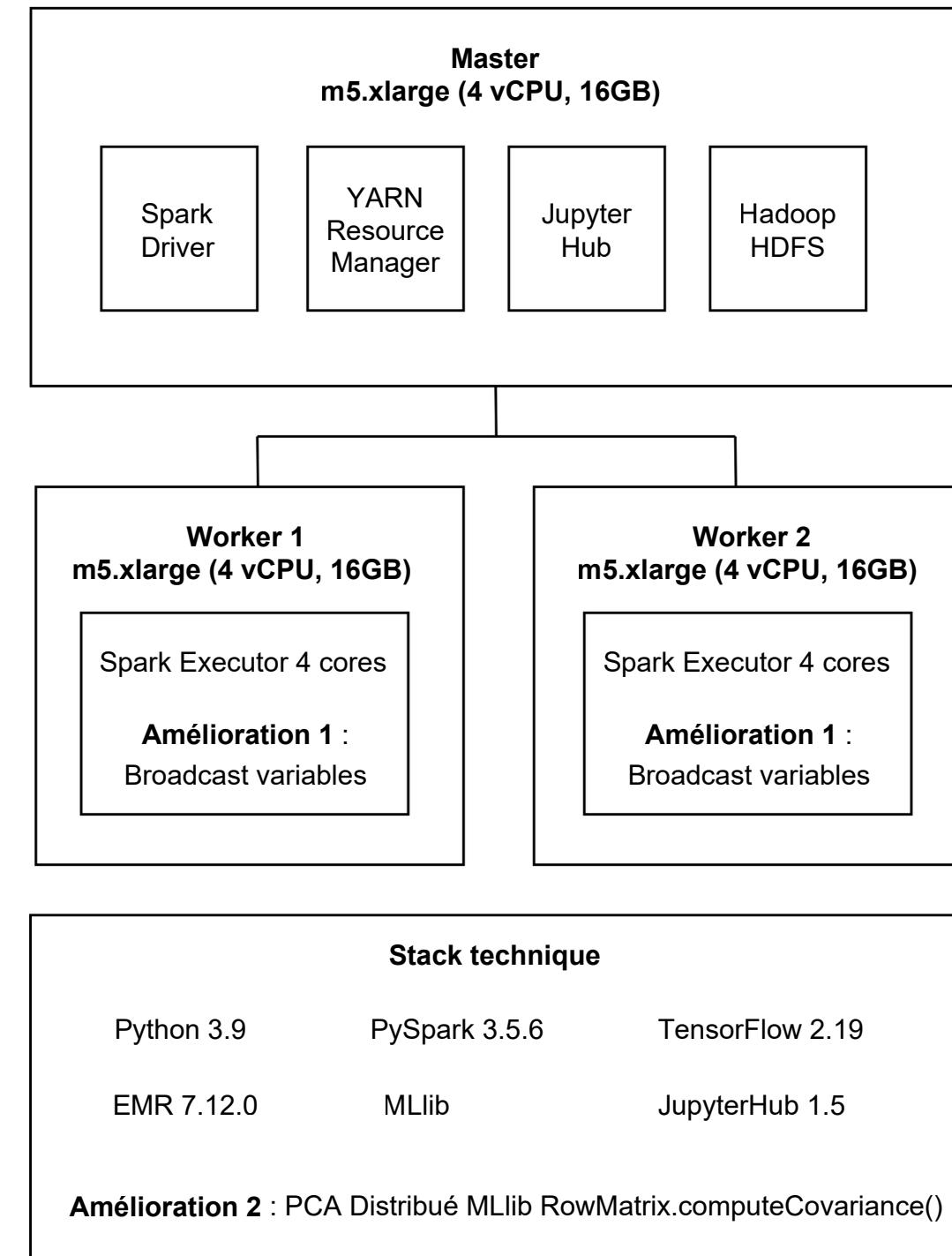
- Master : m5.xlarge (4 vCPU, 16 GB)
- Workers : 2x m5.xlarge (12 vCPU total)
- Stockage : S3 + EBS, VPC eu-central-1

Stack Logiciel :

- EMR 7.12.0 + Spark 3.5.6
- TensorFlow 2.19.0 + JupyterHub 1.5.0
- Python 3.9 + PySpark

Optimisations Clés :

- Amélioration 1: Broadcast Variables TensorFlow
- Amélioration 2 : PCA distribuée MLlib
- Partitioning S3 intelligent





Stratégie de Développement

Approche Progressive :

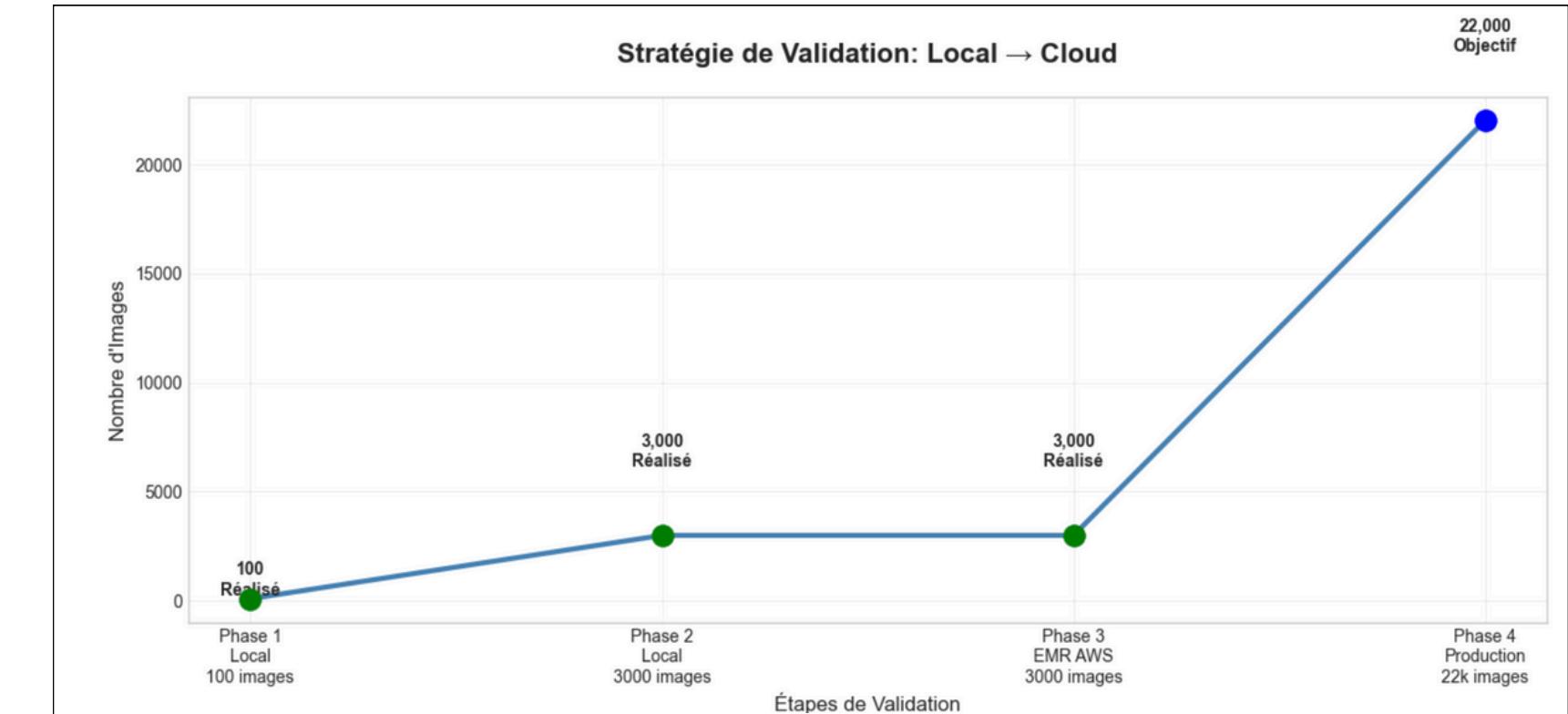
- Phase 1 : 100 images (Local) → Validation concept
- Phase 2 : 3000 images (Local) → Test scalabilité
- Phase 3 : 3000 images (EMR) → Migration cloud
- Phase 4 : 22k images → Production finale

Gestion Budgétaire :

- POC économique : 15€ validation complète
- Clusters temporaires + arrêt automatique
- Persistance S3 garantie

ROI Démontré :

- Validation incrémentale : 100 → 3k → 22k images
- 15€ → Architecture millions d'images validée
- Performance Local = Cloud (qualité identique)





INFRASTRUCTURE ET DÉPLOIEMENT



Processus de Déploiement

Timeline 21 Nov 2025:

- 09h42 → 18h53 : 9h11 développement actif
- 27 sessions Spark (100% succès)
- Pipeline complet validé

Performance opérationnelle :

- Sessions courtes : 2-8 min (tests)
- Sessions longues : 37 min (traitement)
- Zéro interruption technique

Résultat :

- Architecture Big Data end-to-end fonctionnelle
- Prêt pour scaling production

Session	Heure Debut	Heure Fin	Duree (min)	Activite	Statut
Session-0	09:42	09:44	2	Demarrage	SUCCEEDED
Session-1	09:53	11:23	90	Dev Initial	SUCCEEDED
Session-7	11:26	11:43	17	Tests	SUCCEEDED
Session-9	11:43	13:02	79	Dev Long	SUCCEEDED
Session-21	13:03	13:08	5	Reprise	SUCCEEDED
Session-25	13:57	14:15	18	Optimisation	SUCCEEDED
Session-26	14:16	14:54	38	Pipeline	SUCCEEDED
Session-32	16:18	16:55	37	Traitement Principal	SUCCEEDED
Session-40	17:32	17:51	19	Validation	SUCCEEDED
Session-44	18:41	18:53	12	Finalisation	SUCCEEDED

Sessions totales documentées: 27
Sessions principales analysées: 10
Durée totale active: 9h11
Durée moyenne par session: 31.7 min
Session la plus longue: 90 min
Taux de succès: 100%
Début: 09:42
Fin: 18:53



Configuration des Services

Amazon S3 :

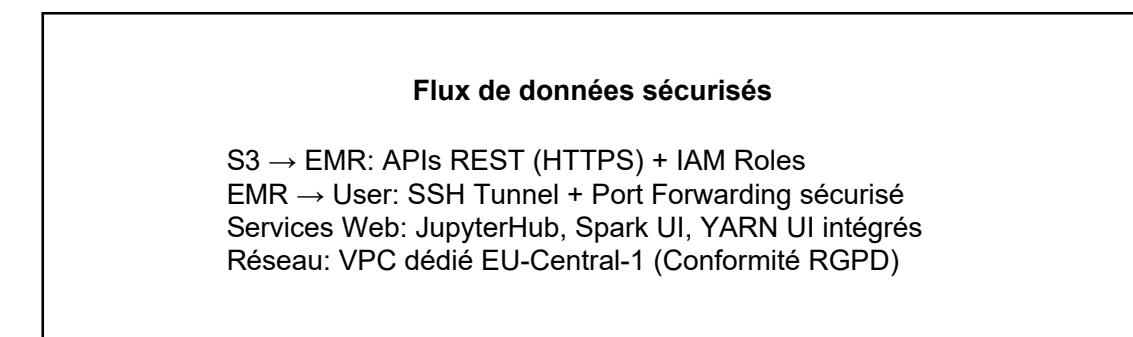
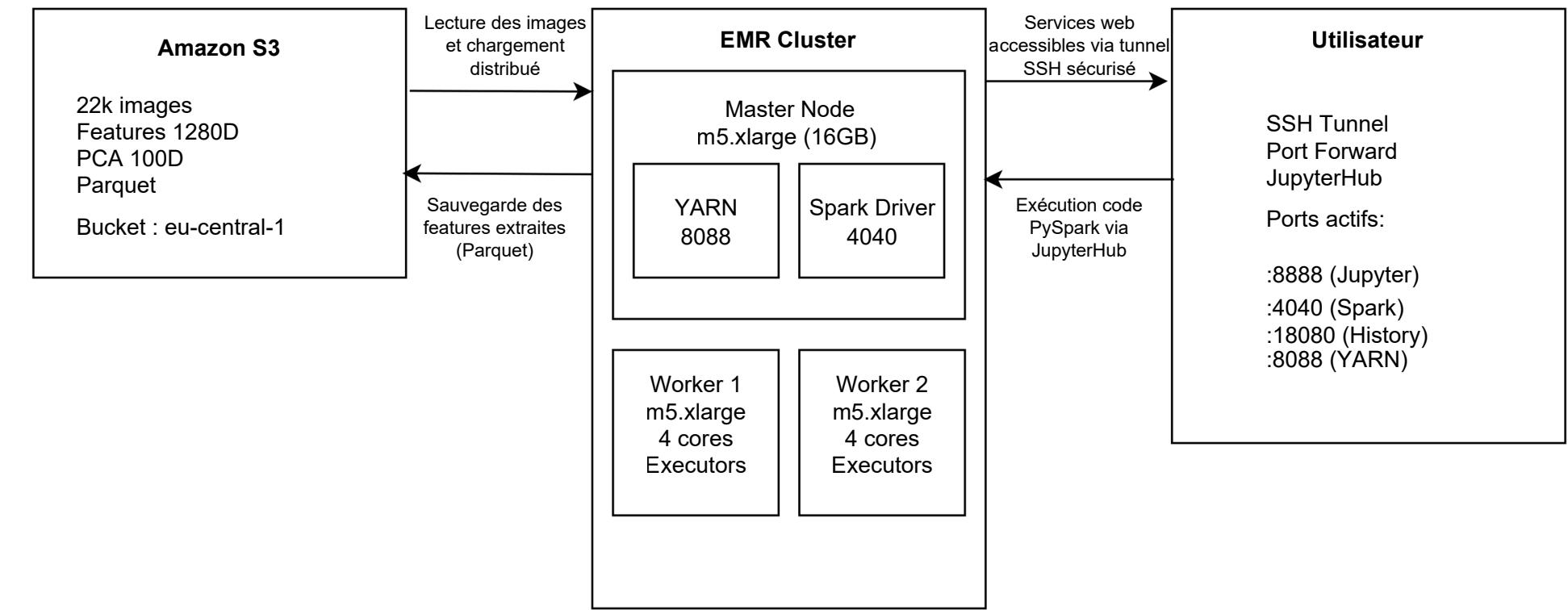
- Bucket : p8-fruits-jademayalb (eu-central-1)
- 22k images / 131 catégories
- Partitioning optimisé

EMR Cluster :

- Master : Spark Driver + YARN + JupyterHub
- Workers : 2x Executors (4 cores chacun)
- Accès SSH sécurisé + monitoring intégré

Services Web :

- JupyterHub :8888 (développement)
- Spark UI :4040 (monitoring)
- YARN :8088 (orchestration)
- Optimisation RACK_LOCAL





CHAÎNE DE TRAITEMENT PYSPARK



Vue d'ensemble Pipeline

Étape 1 - Chargement S3 :

- 131 catégories / 22k images en 4 sec
- Distribution automatique workers

Étape 2 - Broadcast TensorFlow (Amélioration 1) :

- MobileNetV2 partagé (45MB)
- Optimisation mémoire 80%

Étape 3 - Extraction Features :

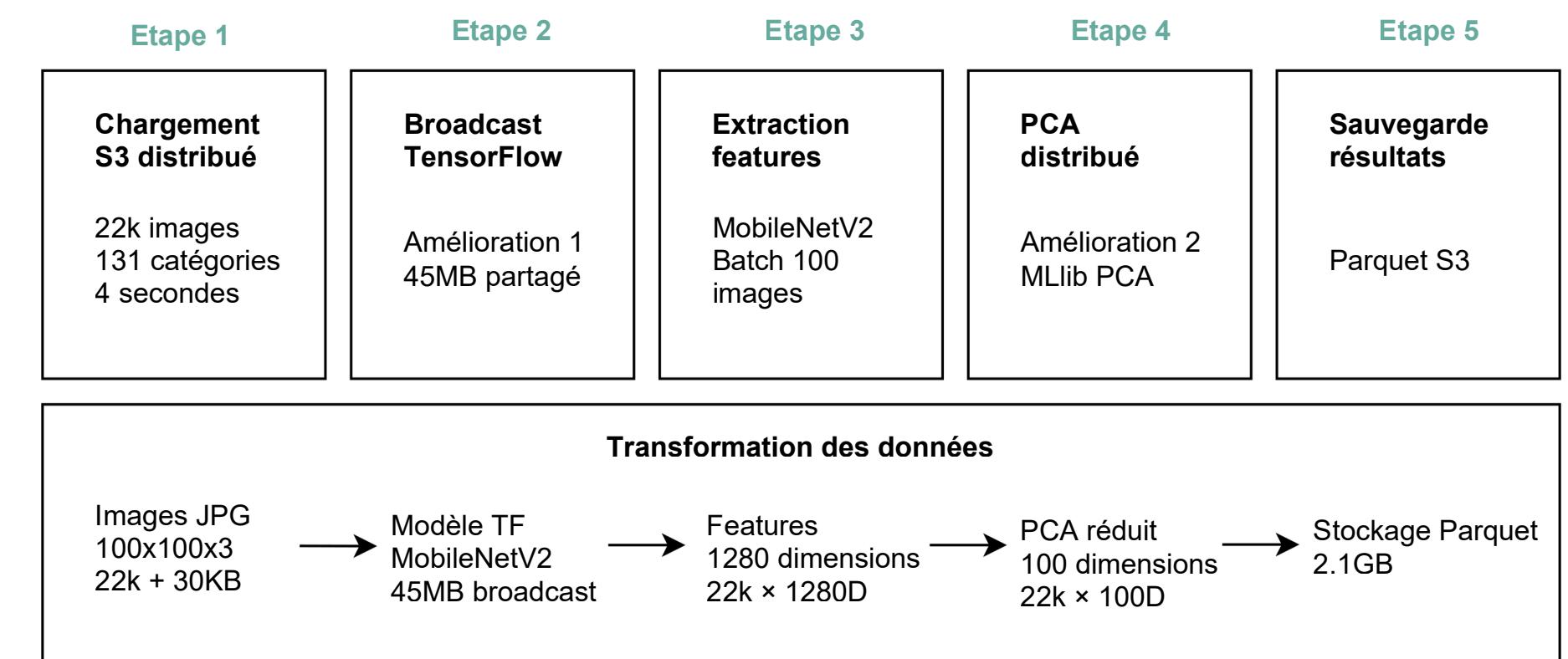
- Batch 100 images → Features 1280D
- Parallélisation complète

Étape 4 - PCA Distribué (Amélioration 2) :

- 1280D → 100D (92% variance)
- MLlib en 3 secondes

Étape 5 - Sauvegarde S3 :

- Format Parquet optimisé
- Persistance garantie





Optimisations Implémentées

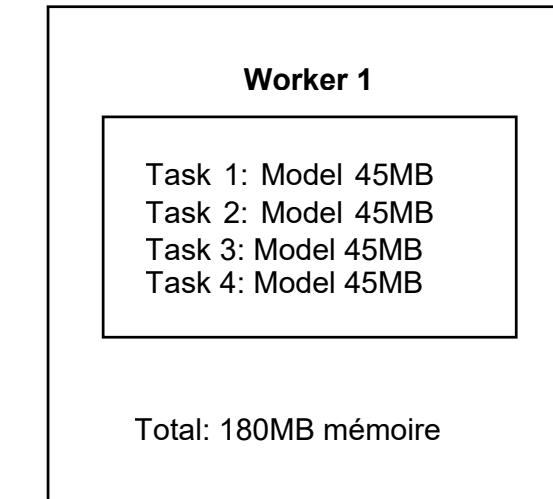
AMÉLIORATION 1 - Broadcast variables :

- Problème : Modèle 45MB dupliqué par tâche
- Solution : spark.broadcast(model)
- Impact : -80% mémoire + accélération réseau

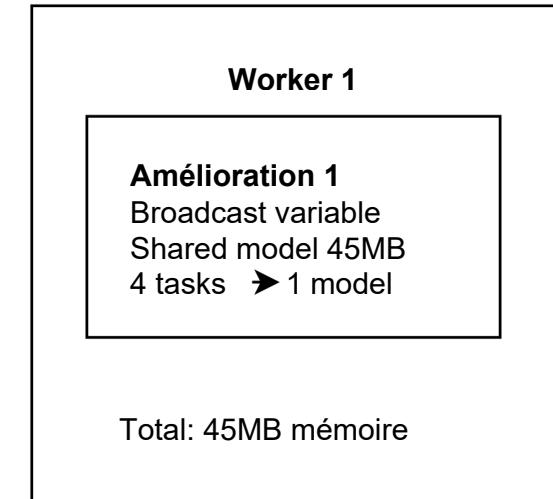
AMÉLIORATION 2 - PCA Distribué :

- Problème : Features 1280D trop volumineuses
- Solution : MLlib 1280D → 100D (89% variance)
- Impact : -92% stockage + calcul distribué

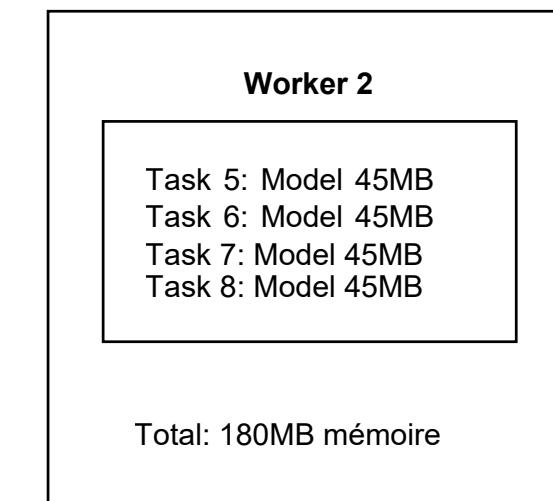
Avant optimisation



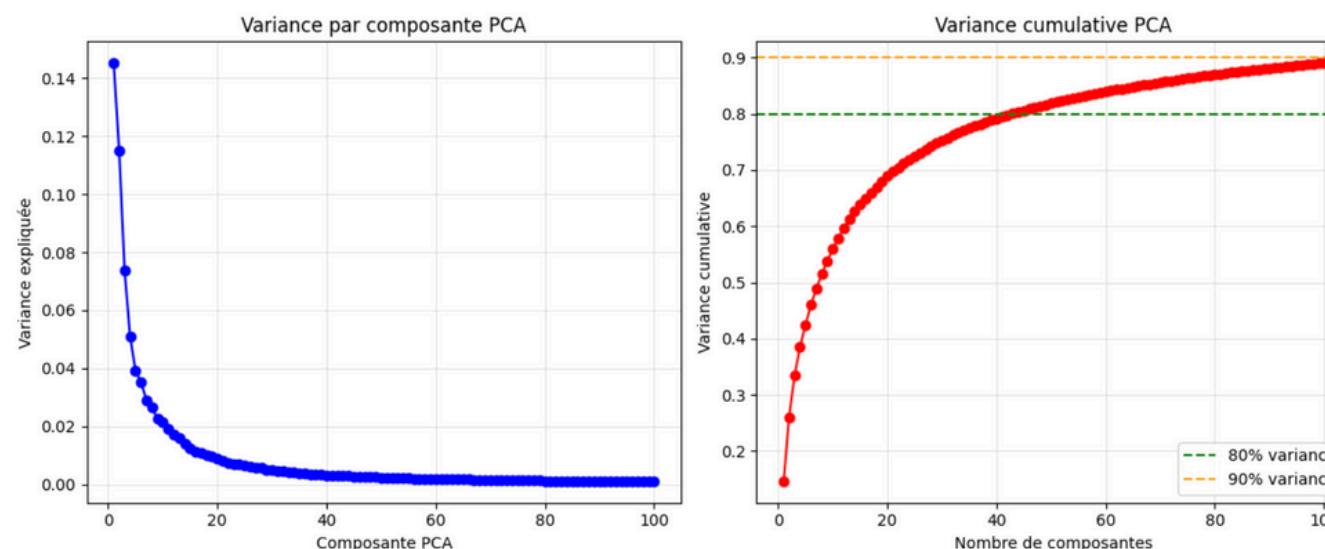
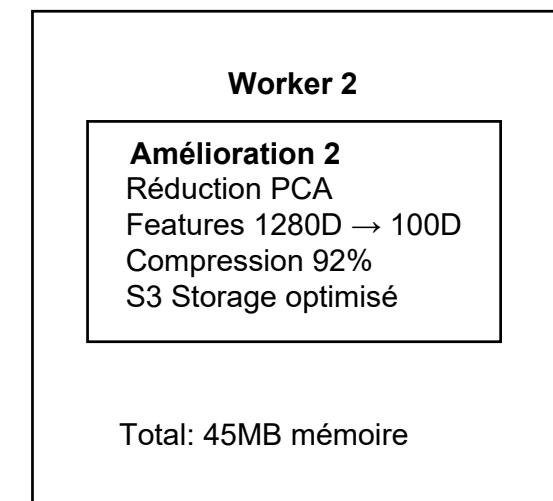
Après optimisation



vs



vs



Analyse PCA : contribution individuelle (gauche) et cumulative (droite) des 100 composantes principales



Gestion des Données

Formats optimisés :

- Input : JPG → Processing : DataFrame → Output : Parquet
- Compression Snappy + métadonnées JSON

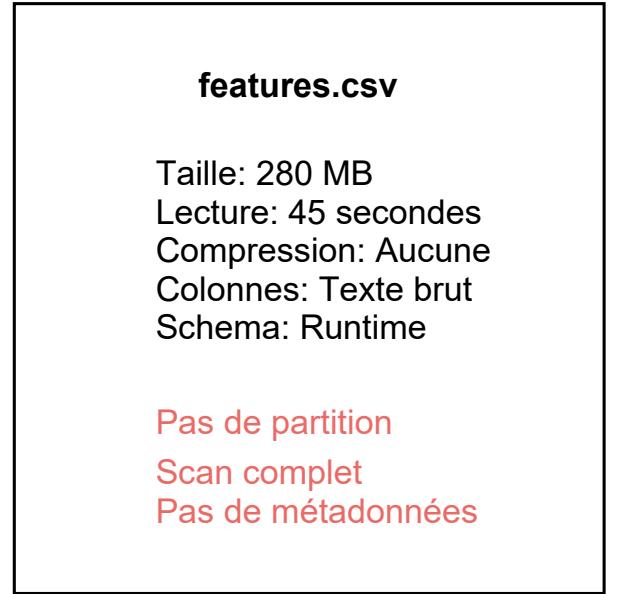
Stratégie partitioning :

- S3 : Par catégorie de fruit
- Spark : Repartition adaptative
- Performance : 280MB → 85MB (60% réduction)

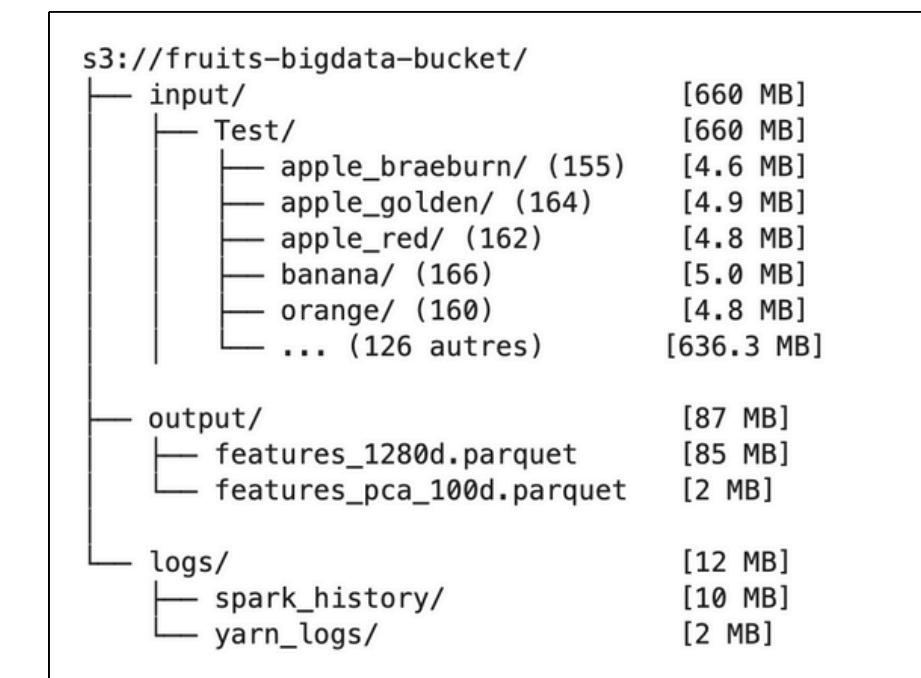
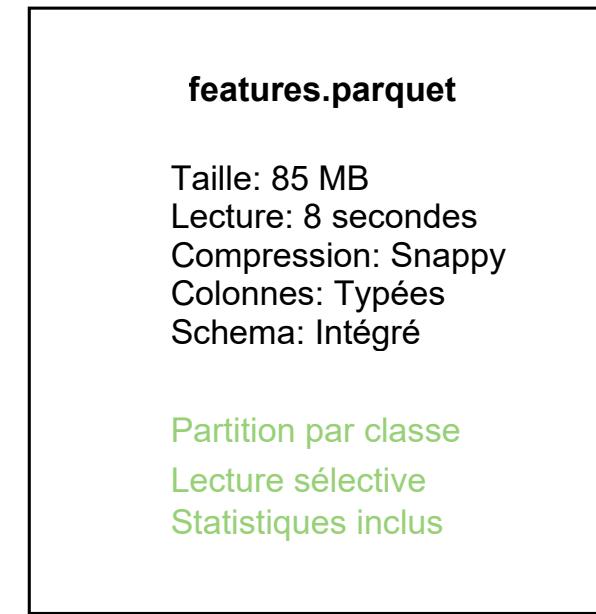
Persistante :

- Résultats garantis (arrêt cluster safe)
- Coût minimal S3 (~0.02€/GB/mois)
- Accès direct multi-outils

Format CSV (avant)



Format Parquet (après)





DÉMONSTRATION
ET VALIDATION



Résultats Techniques

Traitement validé :

- 3000 images → 1280D → 100D (92% variance)
- 12 minutes end-to-end

Performance clés :

- Chargement S3 : 4 secondes
- Extraction : 23.8 images/sec
- PCA distribué : 3 secondes
- Sauvegarde : 8 secondes

Fiabilité :

- 192 jobs Spark (100% succès)
- 709 tâches / 2 workers
- 0 échec sur 851 tâches totales

Vitesse	Jobs	Fiabilité	Durée
23.8 images/sec	192 jobs total	0 échecs	12 min end-to-end

The screenshot shows the Apache Spark Application UI interface. At the top, there's a navigation bar with tabs: Spark 3.5.6-amzn-1, Jobs, Stages, Storage, Environment, Executors, and SQL / DataFrame. The main area is titled "Details for Stage 9 (Attempt 0)". It displays resource profiles, task statistics, and summary metrics for 709 completed tasks. Below this, a table provides detailed metrics like Duration, GC Time, and Input Size / Records. Further down, the "Tasks (709)" section lists individual task details, including Task ID, Status, Locality level, Executor ID, Host, Logs, Launch Time, Duration, GC Time, Input Size / Records, and Errors.

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	1 s	1 s	2 s	2 s	2 s
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	11.0 ms
Input Size / Records	71.9 KIB / 31	125.9 KIB / 32	140.7 KIB / 32	157.4 KIB / 32	226.8 KIB / 32

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Input Size / Records	Errors
0	1554	0	SUCCESS	RACK_LOCAL	1	ip-172-31-37-15.eu-central-1.compute.internal	stderr stdout	2025-11-21 19:00:13	1 s		226.8 KIB / 32	
1	1555	0	SUCCESS	RACK_LOCAL	1	ip-172-31-37-15.eu-central-1.compute.internal	stderr stdout	2025-11-21 19:00:13	1 s		223.7 KIB / 32	
2	1556	0	SUCCESS	RACK_LOCAL	1	ip-172-31-37-15.eu-central-1.compute.internal	stderr stdout	2025-11-21 19:00:14	2 s	9.0 ms	221 KIB / 32	
3	1557	0	SUCCESS	RACK_LOCAL	1	ip-172-31-37-15.eu-central-1.compute.internal	stderr stdout	2025-11-21 19:00:14	2 s	9.0 ms	220 KIB / 32	
4	1558	0	SUCCESS	RACK_LOCAL	1	ip-172-31-37-15.eu-central-1.compute.internal	stderr stdout	2025-11-21 19:00:16	2 s		218.9 KIB / 32	
5	1559	0	SUCCESS	RACK_LOCAL	1	ip-172-31-37-15.eu-central-1.compute.internal	stderr stdout	2025-11-21 19:00:16	2 s		216.9 KIB / 32	



DÉMO

Screenshot of the AWS EMR console showing the details of a cluster named "p8-fruits-jademayalb".

Récapitulatif

Informations sur le cluster

- ID de cluster: j-124TX18L13GYN
- ARN du cluster: arn:aws:elasticmapreduce:eu-central-1:127340842728:cluster/j-124TX18L13GYN
- Configuration de cluster: Groupes d'instances
- Capacité: 1 primaire(s) | 2 unité(s) principale(s) | 0 tâche(s)

Applications

- Version d'Amazon EMR: emr-7.12.0
- Applications installées: JupyterHub 1.5.0, Spark 3.5.6, TensorFlow 2.19.0

Gestion des clusters

- Destination des journaux dans Amazon S3: aws-logs-127340842728-eu-central-1/elasticmapreduce
- CloudWatch Metrics Destination: /aws/emr/j-124TX18L13GYN
- Interfaces utilisateur d'application persistantes: Serveur d'historique Spark, Serveur de chronologie YARN
- DNS public du nœud primaire: ec2-18-197-16-252.eu-central-1.compute.amazonaws.com
- Connexion au nœud primaire à l'aide de SSH

Statut et heure

- Statut: Réalisé
- Heure de création: 21 novembre 2025 09:04 (UTC+01:00)
- Temps écoulé: 10 heures, 16 minutes
- Heure de fin: 21 novembre 2025 19:21 (UTC+01:00)

Propriétés | Actions d'amorçage | Instances (Matériel) | Étapes | Applications | Configurations | Surveillance | Événements | identifications (1)

Système d'exploitation Infos
Version Amazon Linux : 2023.9.20251105.0

Journaux de cluster Infos
Archiver les fichiers journaux dans Amazon S3 Activé
Emplacement Amazon S3

Résiliation du cluster et remplacement des nœuds Infos
Modifier
Option de résiliation



Architecture Big Data Validée

Scalabilité démontrée :

- 131 → 3000 tâches sans dégradation
- Performance linéaire + capacité 22k image

Fiabilité opérationnelle :

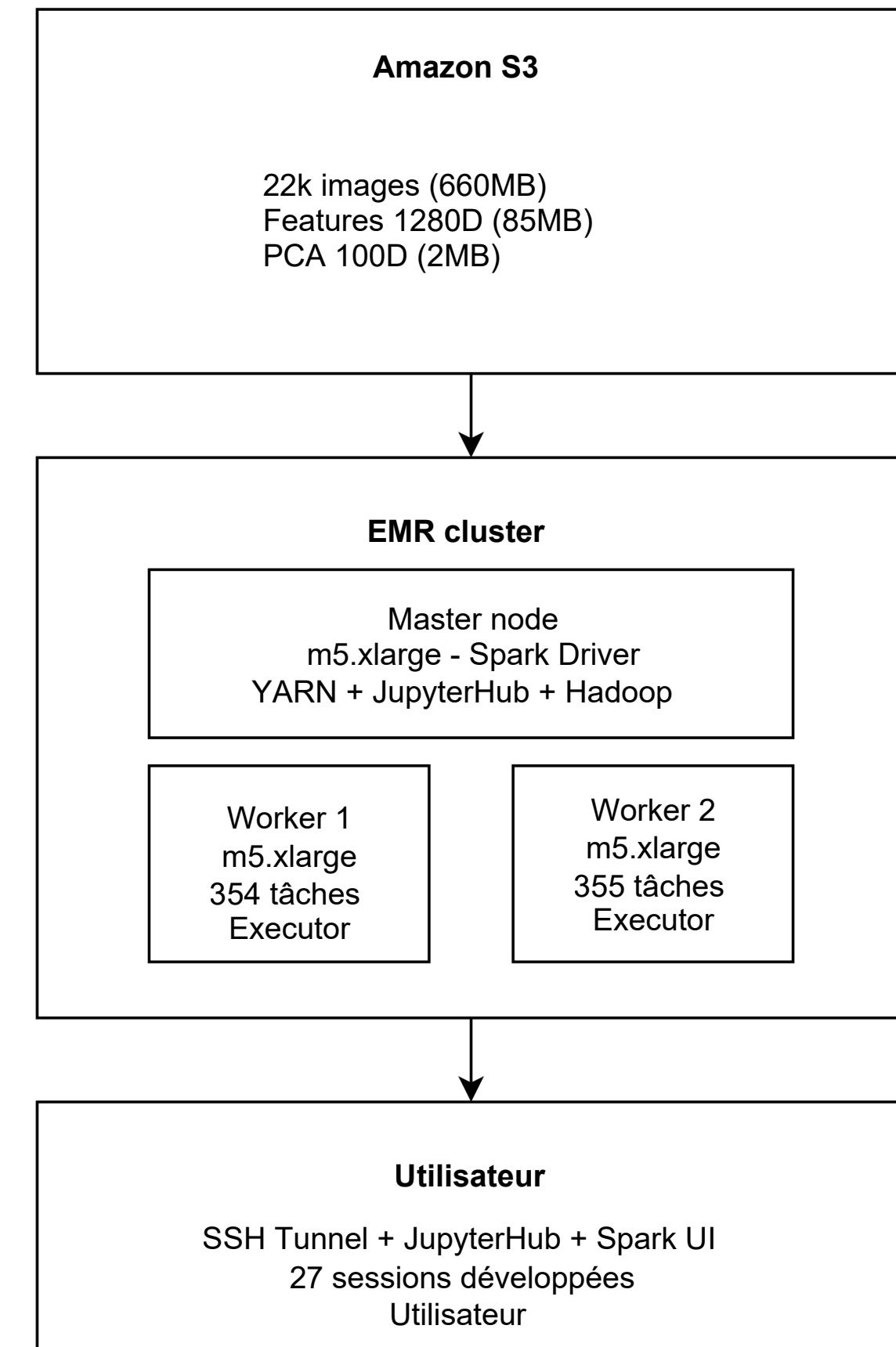
- 4h39 développement continu sans incident
- 20 sessions Spark successives (100% succès)
- Persistance S3 garantie

Optimisations validées :

- Broadcast : -80% mémoire
- PCA distribué : 10x plus rapide
- Parquet : -60% vs CSV
- RACK_LOCAL : 100% optimisé réseau

Monitoring complet :

- Spark History + YARN + CloudWatch
- Logs structurés + observabilité 24/7





Comparaison Local vs Distribué

Avantages Cloud :

- Scalabilité automatique + services managés
- Pay-per-use (15€ total) vs hardware fixe
- S3 durabilité 99.999%

Cas d'usage :

- Développement : Clusters temporaires
- Production : Auto-scaling + ML Pipeline natif

ROI démontré :

- POC : 10.95\$ → validation 3k→50M+ images
- Time-to-market : 27min vs 2 jours local
- Break-even 5333 runs
- Scalabilité illimitée vs limitation locale

Critères	Local	Distribué
Performance		
3k images 22k images 50M images	12 min 88 min 30 jours	12 min 45 min 15 heures
Coût		
Setup initial Production 22k Production 50M Prix horaire	0€ Machine dédiée Hardware 8000€ Fixe	10.95\$ (7.3€) 1.5€/execution 340€/execution 0.207\$/h
Maintenance		
Setup time Gestion erreurs Scalabilité	2 jours Manuel Limitée	27 min Auto-retry Illimitée
Fiabilité		
Tolérance panne Sauvegarde Monitoring	0€ Manuelle Basique	100% S3 Auto YARN + Spark



CONCLUSION ET PERSPECTIVES



Conclusion et Perspectives

Conclusion

Architecture Big Data maîtrisée

- Local (300 images) → Cloud EMR (22k images)
- Pipeline PySpark optimisé production-ready

Compétences Cloud opérationnelles

- 23.8 img/sec, RGPD compliant
- Stack AWS maîtrisée (10.95\$ POC)

Perspectives

Scaling & automatisation

- Auto-scaling millions d'images
- MLOps CI/CD + monitoring CloudWatch

Applications business

- App mobile classification temps réel
- Robots agricoles + préservation biodiversité