Adrien NORMAND

MISSION 9

Big Data Cloud Processing with AWS EMR

Distributed Image Feature Extraction with PySpark & TensorFlow

January 2026

Project Context

The Challenge

🍎 Fruits! - AgriTech Startup

- Mobile app for fruit recognition
- Users take photos → App identifies fruit
- Growing user base requires scalability
- Current local processing: NOT scalable

⚡ The Big Data Challenge

- Process ~90,000 fruit images
- Extract deep learning features
- Enable future model training
- GDPR Compliance required (EU data)

✅ Our Solution: AWS Cloud Big Data Pipeline

- Infrastructure as Code with Terraform → reproducible, version-controlled
- AWS EMR (Elastic MapReduce) → managed Spark cluster, auto-scaling
- S3 Storage → durable, scalable object storage in EU region
- PySpark + TensorFlow → distributed feature extraction with transfer learning
- One-click deployment → from zero to running cluster in ~15 minutes

🍎 Fruits-360 Dataset

- Authors: Horea Muresan & Mihai Oltean
- Source: Kaggle (Public Domain)

Complete Architecture

Local Development + Cloud Infrastructure

💻 LOCAL MACHINE

🐳 Docker Container

- Terraform v1.6
- AWS CLI v2

📝 Commands

- terraform init
- terraform apply
- terraform destroy

🔑 SSH Key (auto-generated)
mission9-key.pem

AWS API

☁️ AWS CLOUD (eu-west-1 - Ireland) - GDPR COMPLIANT

🔐 IAM

- EMR Service Role
- EC2 Instance Profile

📦 S3 Bucket

- fruits-360/ (input)
- Results/ (output)

🛡️ Security Groups

- Master: port 9443
- Slave: internal only

⚡ EMR Cluster (emr-6.15.0) - Spark + JupyterHub + Hadoop

🖥️ Master (m5.xlarge)

- JupyterHub :9443 (HTTPS)
- Spark Driver
- 4 vCPU, 16GB RAM
- Public DNS access

⚙️ Workers x2 (m5.xlarge)

- Spark Executors
- TensorFlow processing
- 4 vCPU, 16GB RAM each
- Internal network only

AWS Cloud Architecture

Data Flow & Components

1 S3 Input

- fruits-360/Training
- ~68K JPEG
- 100x100 px

2 Spark Read

- binaryFile format
- DataFrame
- 48 partitions

3 TensorFlow

- MobileNetV2
- Broadcast weights
- 1280 features

4 PCA

- Spark ML
- 1280 → 50
- ~90% variance

5 S3 Output

- Parquet format
- CSV export

🔧 Bootstrap Actions

- Install: tensorflow, pandas, pillow, pyarrow
- Install Java in JupyterHub container
- Download & upload dataset to S3

⏱️ Auto-Termination

- Idle timeout: 1 hour
- Cost optimization
- Prevents forgotten clusters

Technology Stack

Infrastructure & Processing Tools

🏗️ INFRASTRUCTURE

- Terraform v1.6
  → Infrastructure as Code
- Docker Compose
  → Containerized deployment
- AWS EMR 6.15.0
  → Managed Spark cluster
- AWS S3
  → Object storage

⚡ PROCESSING

- Apache Spark 3.5
  → Distributed computing
- PySpark DataFrame
  → Data manipulation
- Pandas UDF
  → Vectorized operations
- PyArrow
  → Parquet I/O

🧠 MACHINE LEARNING

- TensorFlow 2.15+
- → Deep learning framework
- MobileNetV2
  → Transfer learning
- Spark MLlib PCA
  → Dimensionality reduction
- ImageNet weights
  → Pre-trained features

🔧 DEVOPS

- AWS CLI v2
  → Cloud management
- IAM Roles
  → Security & permissions
- JupyterHub
  → Interactive notebooks
- Git
  → Version control

📋 Version Summary

Python: 3.10 | Spark: 3.5.3 | TensorFlow: 2.15+ | EMR Release: 6.15.0 | Terraform: ~1.6

PySpark Workflow

10-Step Distributed Processing Pipeline

1. **SparkSession** — Configure with S3 access
2. **S3 Paths** — Define input/ output buckets
3. **Load Images** — spark.read .binaryFile()
4. **MobileNetV2** — Load pretrained model
5. **Broadcast** — Distribute weights to executors
6. **Pandas UDF** — featurize_udf() 1280 features
7. **Repartition** — Optimize for parallel PCA
8. **PCA** — 1280 ⟶ 50 dimensions
9. **Save Parquet** — Write to S3 Results/
10. **Export CSV** — Human-readable format

GDPR Compliance

European Data Residency & Security

🇪🇺 AWS Region: eu-west-1

📍 Location: Ireland

✅ All data stored in EU
✅ All processing in EU
✅ No transatlantic transfer

🔒 Alternative: eu-west-2 (Paris)

🔐 Security Measures

- S3 Public Access Blocked (block_public_acls = true)
- Security Groups: Only port 9443 (JupyterHub) exposed
- IAM Roles: Least-privilege access for EMR & EC2
- SSH Keys: Auto-generated, stored locally only

📋 Compliance Checklist

- ✅ Data residency: All resources in eu-west-1
- ✅ Encryption: S3 server-side encryption available
- ✅ Access control: IAM policies enforced
- ✅ Audit trail: CloudTrail logging enabled

💡 *GDPR Article 44-49: Personal data can be processed within EU without additional safeguards*

Competency 1: Cloud Tools Selection

Select Cloud tools for Big Data processing compliant with GDPR

✅ 2/2 COMPLETE

CE1: Identify Big Data architecture components

- ✅ Storage Layer: AWS S3 - scalable object storage for images and results
- ✅ Compute Layer: AWS EMR - managed Spark cluster for distributed processing
- ✅ Interface Layer: JupyterHub - interactive notebook environment
- ✅ Processing Engine: Apache Spark - distributed data processing framework

CE2: Identify GDPR-compliant Cloud tools

- ✅ Region Selection: eu-west-1 (Ireland) - data stays within EU territory
- ✅ AWS Compliance: AWS holds EU-approved certifications (ISO 27001, SOC)
- ✅ Data Residency: All S3 buckets and EMR clusters deployed in EU region
- ✅ No Cross-Border Transfer: Processing and storage both in same EU region

Competency 2: Big Data Preprocessing & Modeling

✅ 3/3 COMPLETE

Preprocess and model data using Cloud tools with GDPR compliance

CE1: Load files to GDPR-compliant storage

- ✅ Input: s3://bucket/fruits-360/
- ✅ Output: s3://bucket/Results/
- ✅ EU region data residency

CE2: Execute scripts on cloud machines

- ✅ EMR Cluster: 1 Master + 2 Workers
- ✅ Instance type: m5.xlarge
- ✅ Spark distributes tasks

CE3: Write outputs directly to cloud storage

- ✅ features_df.write.parquet(PATH_Result) → writes directly to S3
- ✅ results_csv.write.csv(PATH_CSV) → human-readable export
- ✅ No local storage needed - direct cloud I/O
- ✅ Parquet format for efficient columnar storage

Competency 3: Distributed Computing

Perform distributed computing with appropriate tools & GDPR compliance

✅ 4/4 COMPLETE

CE1: Identify critical scaling treatments

- ✅ Broadcast: sc.broadcast(weights)
- ✅ Repartition: optimize parallelism
- ✅ Pandas UDF: batch processing

CE2: Ensure GDPR compliance

- ✅ S3 bucket in eu-west-1
- ✅ EMR cluster in eu-west-1
- ✅ No data leaves EU territory

CE3: Develop Spark scripts

- ✅ spark.read.format('binaryFile')
- ✅ PySpark DataFrame API
- ✅ Spark ML PCA transformer
- ✅ Parquet/CSV writers

CE4: Execute full pipeline in cloud

- ✅ JupyterHub on EMR master
- ✅ S3 read → Spark → S3 write
- ✅ No local processing needed
- ✅ One-click 'Run All' execution

Key Metrics & Results

Performance and Output Statistics

📊 Dataset: Fruits-360

- Total images: 90,483
- Training set: 67,692
- Test set: 22,688
- Classes: 131 fruits/vegetables
- Image size: 100×100 px
- Format: JPEG

⚡ Processing

- Input features: 1,280
- PCA components: 50
- Variance retained: ~90%
- Partitions: 48
- Batch size: variable
- Output: Parquet + CSV

🖥️ Infrastructure

- EMR: 1 Master + 2 Workers
- Instance: m5.xlarge
- vCPU total: 12
- RAM total: 48 GB
- Region: eu-west-1
- Auto-terminate: 1h idle

✅ OUTPUT DELIVERABLES

📁 **s3://bucket/Results/**
Parquet files with 1280 features per image

📁 **s3://bucket/Results_PCA/**
Parquet files with 50 PCA components

📁 **s3://bucket/Results_CSV/**
CSV export for external analysis

Conclusion & License

Summary and Credits

🎯 Mission Accomplished

- ✅ All 9/9 competency criteria validated
- ✅ GDPR-compliant infrastructure (EU)
- ✅ Scalable Big Data pipeline
- ✅ Infrastructure as Code (Terraform)
- ✅ One-click deployment & execution
- ✅ Auto-termination for cost control

🚀 Quick Start Commands

- docker compose up -d
- terraform init && terraform apply
- Access: https://[DNS]:9443
- Login: jovyan / jupyter
- Run All cells in notebook
- terraform destroy (cleanup)