FRAUD DETECTION IN A GOVERNMENT AGENCY (MLOPS)

INTRODUCTION

- CONTEXT: Increasing number of fraud suspicions in government agency program
- PROBLEM: Thousands of applications, manual fraud detection not possible
- GOAL: Automated fraud detection system, adaptable to changing data
- APPROACH: Apply MLOps principles like iterative model training and retraining, monitoring, experiment logging and versioning (Luu et al., 2024, pp. 2-4)
- DATASET: Credit Card Fraud Detection (Credit Card Fraud Detection, 2025)

OUTLINE

- Requirement Analysis
- System Architecture
- Fraud Detection Model
- MLflow Tracking & Logging with S3 Backups
- Retraining Scenarios
- REST-API
- Simulation of One Year
- Challenges and Limitations
- Conclusion

REQUIREMENT ANALYSIS

- Automated fraud detection
- Easily adaptable and retrainable
- Monitored for reliability and performance
- RESTful API serves integration
- Secure Acces Control
- Cloud-ready / scalable

SYSTEM ARCHITECTURE

Data Flow:

- Online applications → weekly CSVs stored in S3 → Drift Watchdog → Retraining → Random Forest → Flask API

Docker Container:

- hosts scripts & API
- ensures consistent environments (McKendrick & Gallagher, 2018, pp. 4–8)

MLOps:

- GitHub Actions automates retraining

Security:

- API token for access control

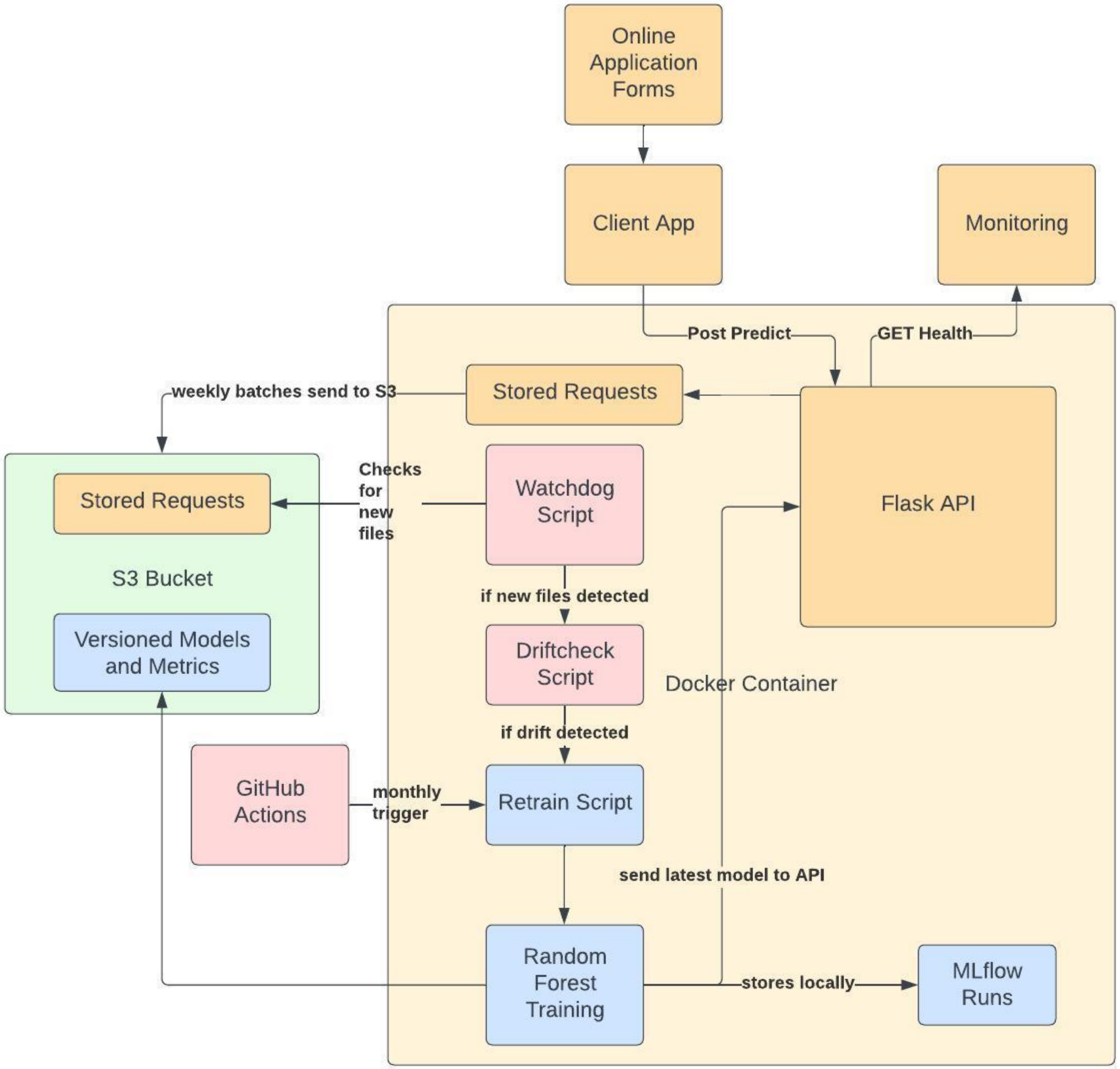

Image Source: Author

FRAUD DETECTION MODEL: RANDOM FOREST

- “Slim” tree for faster training & retraining
- Robust, handles imbalanced data, reduces false negatives (Mihali & Niță, 2024, p.111)
- Tuned hyperparameters: 25 trees, max depth 8, min samples split 5, min samples leaf 3
- Trained on the last 12 weeks of data from S3
- Predicts fraud probability for new applications

MODEL VERSIONING AND LOGGING

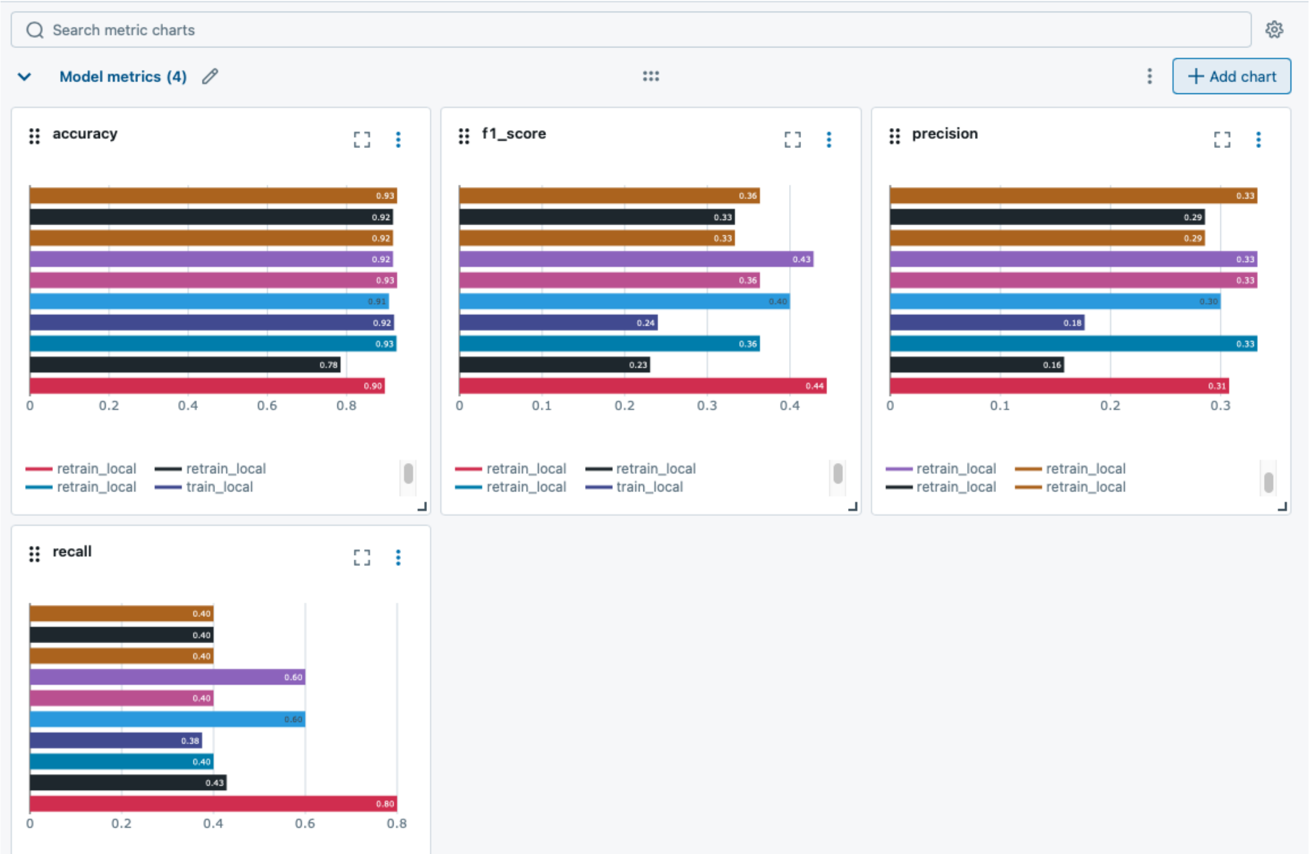

Image Source: Author

- **MLflow:**
  - tracks experiments inside Docker Container
  - Logs training metrics, model artifacts (Luu et al., 2024, pp. 136-150)
- **S3-Backup:**
  - Models and metrics versioned and stored in S3
- → This Setup enables performance monitoring

SCHEDULED MONTHLY RETRAINING

- Retraining triggered automatically every month via GitHub Actions
- Uses the last four weeks of data from S3
- New model logged in MLflow and backed up in S3 and deployed to Flask API
- Ensures system adapts to evolving patterns over time

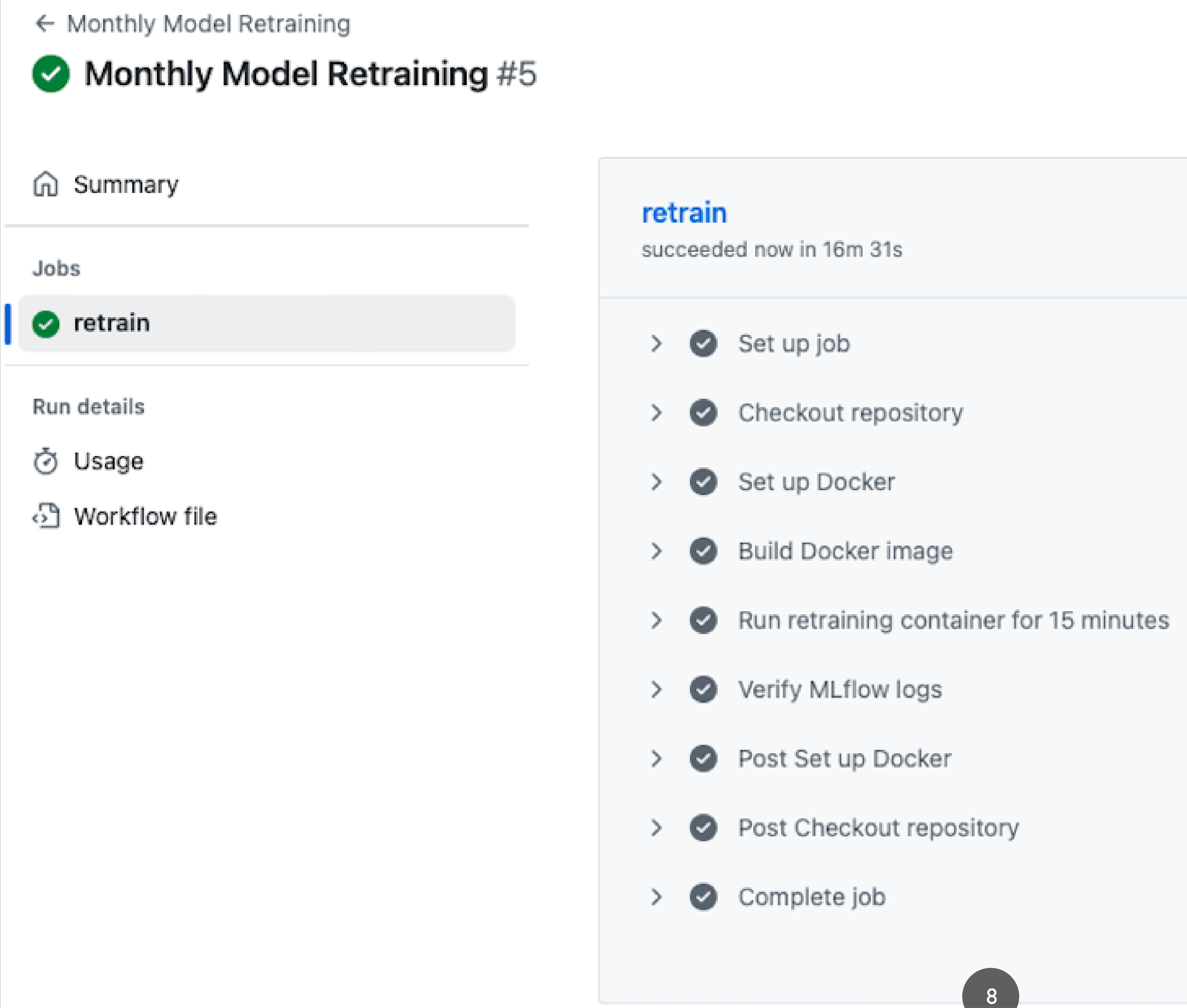

Image Source: Author

DRIFT BASED RETRAINING

- All API requests are stored locally and uploaded weekly to S3.
- **Drift Watchdog script** detects new files
- For each new file:
  - Compare feature distributions using two-sample-t-test (Kutner et al. 2013, pp 278-288)
  - Bonferroni correction applied: α divided by number of features to control overall false positives (Kutner et al. 2013, pp 155-156)
  - Corrected p-value < 0.05 → feature marked as drifted
- **Drift detected** → Trigger retraining in Docker
- **MLflow** logs new model and metrics
- **S3 backup** stores versioned model, metrics, input example
- **API** updated with latest model

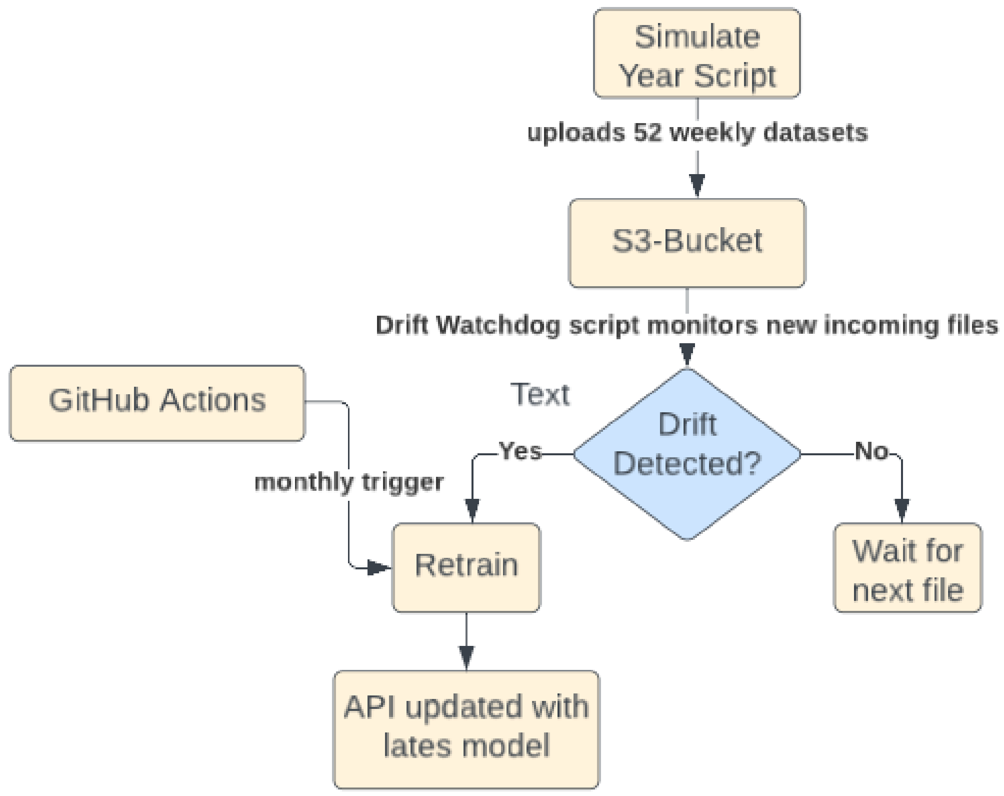

Image Source: Author

SIMULATION OF ONE YEAR

- Simulation script continuously uploads new files to S3
- Drift Watchdog script runs in Docker, monitoring new files in real-time
- Validates that MLflow logging, S3 backups, and API updates happen as expected

REST-API

Available Actions:

- **POST** /predict → submit new application data, get fraud probability
- **GET** /health → check API and system status

Process Flow:

- Client sends request to the Flask API
- API accesses the latest Random Forest model
- Model returns predictions as JSON

CHALLENGES AND LIMITATIONS

Challenges:

- S3 connection: credentials & paths
- Deploying on EC2 was challenging

Limitations & Next Steps:

- **Scaling & Deployment:**
  - Single container limits scalability and security → consider Docker Swarm for better isolation and encrypted communication (Ganne, 2022, p.4)
- **Compute & Performance:**
  - t3.micro EC2 is slow for training/retraining → upgrade instance
- **Model & Monitoring:**
  - Simple drift detection, small Random Forest → implement advanced drift detection, consider larger ensemble models

CONCLUSION

- System successfully detects fraud automatically using Random Forest
- Retraining triggered monthly or on data drift
- Models logged in MLflow and versioned backups stored in S3
- REST API provides seamless integration with client applications
- **Next Steps:** Deploy on EC2, consider Docker Swarm, enhance monitoring

BIBLIOGRAPHY

Credit Card Fraud Detection. (2025). [Dataset]. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Ganne, A. (2022). CLOUD DATA SECURITY METHODS: KUBERNETES VS DOCKER SWARM. *International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/IRJMETS32176

Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2013). *Applied Linear Statistical Models* (5th ed.).

Luu, H., Pumperla, M., & Zhang, Z. (2024). *MLOps with Ray: Best practices and strategies for adopting machine learning operations*. Apress L. P.

McKendrick, R., & Gallagher, S. (2018). *Mastering docker: Unlock new opportunities using Docker's most advanced features, third edition* (3rd ed). Packt Publishing.

Mihali, S.-I., & Niță, Ștefania-L. (2024). Credit Card Fraud Detection based on Random Forest Model. *2024 International Conference on Development and Application Systems (DAS)*, 111–114. https://doi.org/10.1109/DAS61944.2024.10541240

LIST OF FIGURES