IRIS News Curator: Intelligent Risk and Incident Sharing – An Enhancement Project

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING 計算機科學及工程學系

TAYDEY, Charline

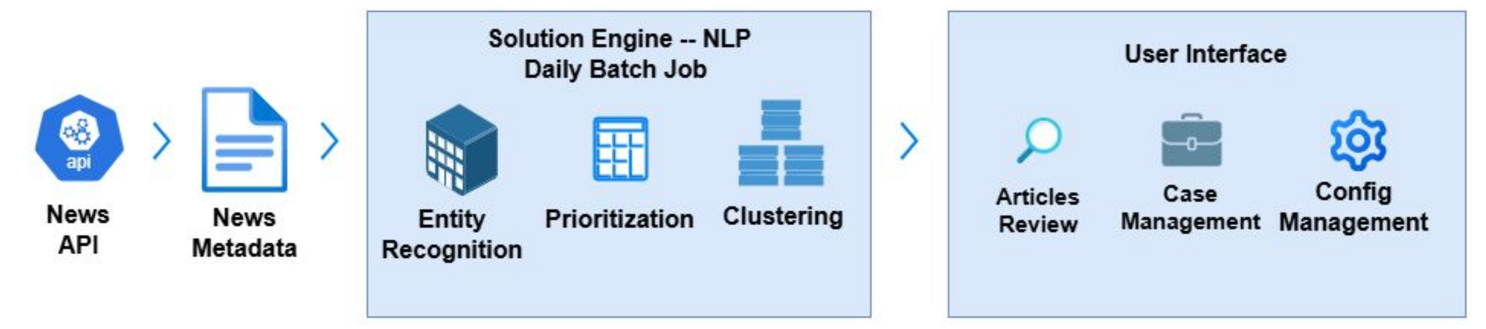
Supervised by Prof. XIAO, Huiru

Deloitte.

Introduction

Money Laundering is a global issue that undermines financial institutions by masking illicit activities through complex financial transactions. Traditionally, **anti-money laundering (AML)** efforts rely on **labor-intensive manual reviews** of news articles, making detection and monitoring challenging and time-consuming.

To address this, **IRIS** is introduced – an **NLP-based news monitoring system** designed to streamline the detection of relevant news articles. IRIS enhances **efficiency** and **accuracy** in detecting potential money laundering activities while **reduces the workload** of AML divisions. To enhance IRIS system's capabilities, **a comprehensive enhancement project has been initiated.**

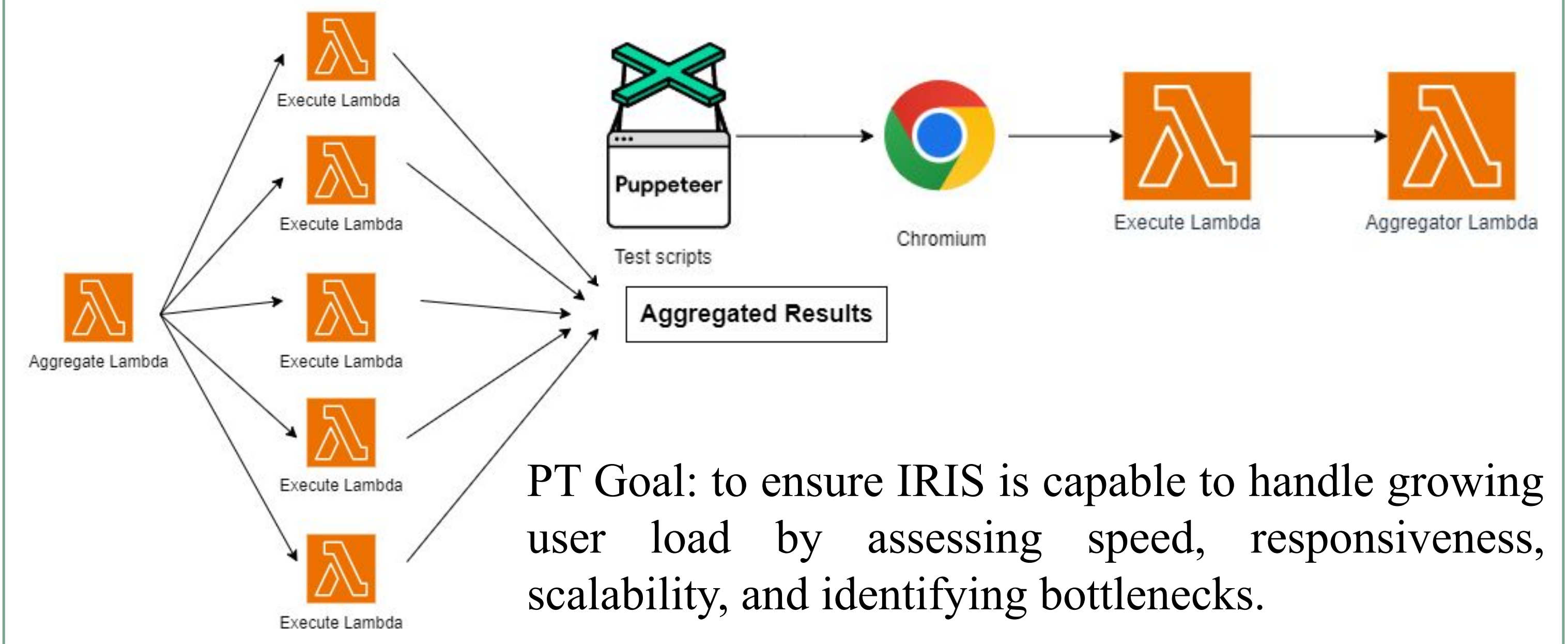


Objectives

- ✔ Implement **an automated framework for performance testing**.
- ✔ Transition the application to a **containerized architecture**.
- ✔ Develop a **customized Named Entity Recognition system.**.
- ✔ Design **Model Feedback Loop** system for model performance analysis.

Design and Implementation

Automated Testing Framework for Performance Test (PT)



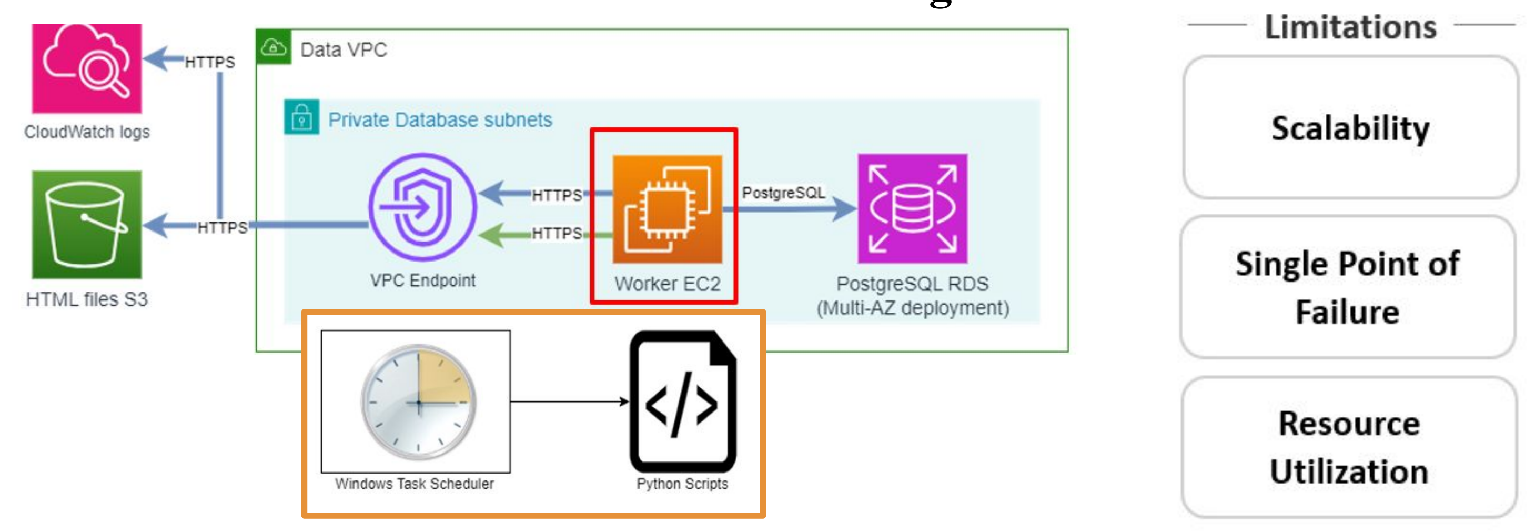
PT Goal: to ensure IRIS is capable to handle growing user load by assessing speed, responsiveness, scalability, and identifying bottlenecks.

Automated Testing Framework for PT – Development Logic

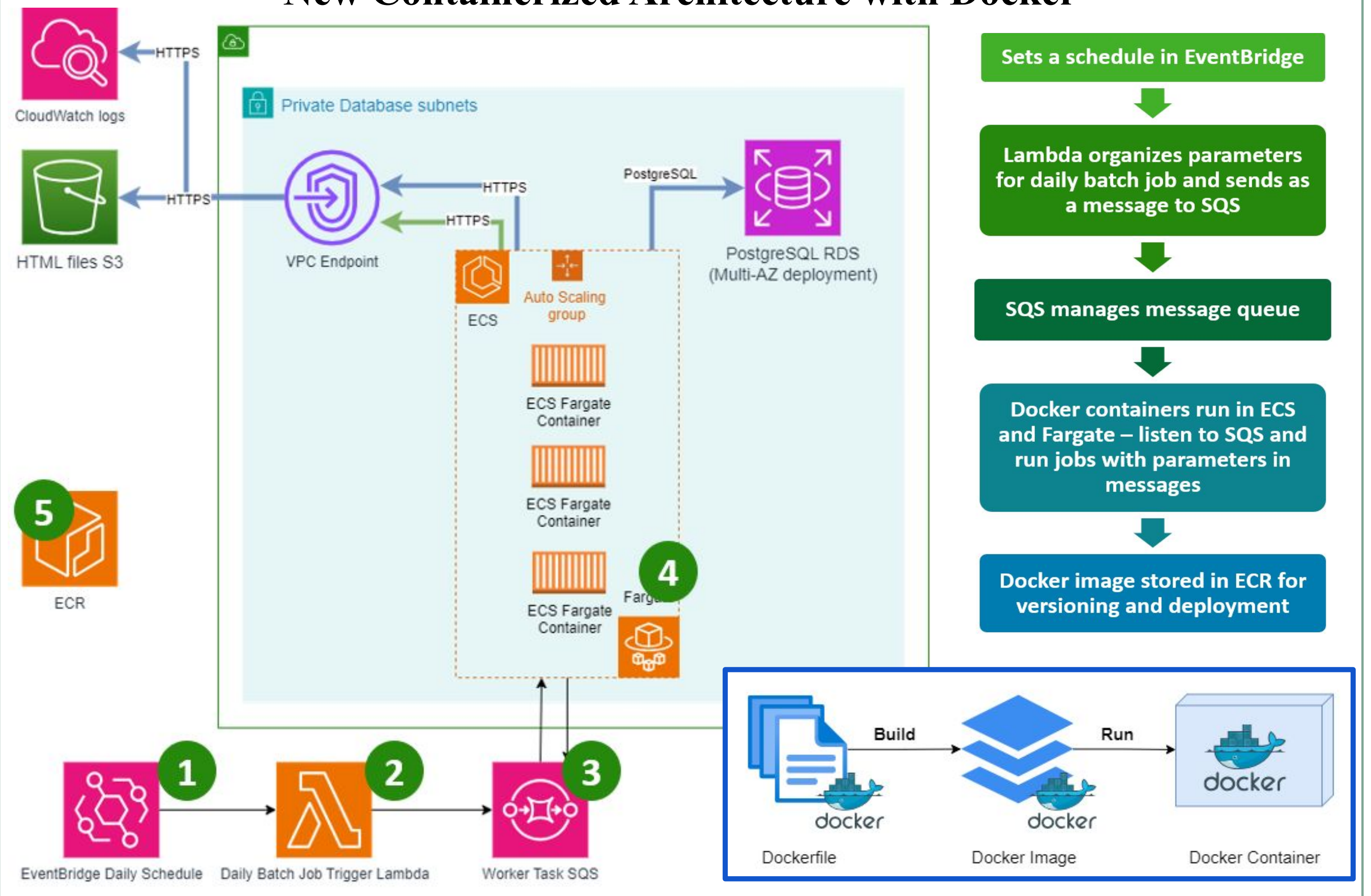
- **Puppeteer** controls browser, simulating one user action.
- Puppeteer scripts deployed to AWS Lambda, called **ExecuteLambda**.
- **AggregateLambda** invokes 50 **ExecuteLambda** - simulates 50 concurrent users on IRIS, tracking speed and response times.

Streamline Data Pipeline for NLP by Re-Architecture

Old EC2-based Design



New Containerized Architecture with Docker


CloudWatch logs
HTTPS
HTML files S3
HTTPS
Private Database subnets
VPC Endpoint
HTTPS
HTTPS
PostgreSQL
PostgreSQL RDS (Multi-AZ deployment)
ECS
Auto Scaling group
ECS Fargate Container
ECS Fargate Container
ECS Fargate Container
Farg
5
ECR
4
1
EventBridge Daily Schedule
2
Daily Batch Job Trigger Lambda
3
Worker Task SQS
Sets a schedule in EventBridge
Lambda organizes parameters for daily batch job and sends as a message to SQS
SQS manages message queue
Docker containers run in ECS and Fargate – listen to SQS and run jobs with parameters in messages
Docker image stored in ECR for versioning and deployment
Build
Run
docker
docker
docker
Dockerfile
Docker Image
Docker Container

First implemented with **Single Stage Build** with Slim Parent Image but docker **image size was too large - not optimal to deploy**. Therefore, we **optimized with Multi Stage Build** to separate build and runtime environments.

Result

Performance Test Contribution to IRIS

Early Testing Phase

- IRIS struggled to handle more than **10 concurrent users, latency issues and occasional failures** when scaling – **Optimization was required.**

Bottlenecks Discovered:

- API Failures identified: **504 Gateway Timeout Error** during high load.
- Optimization steps:
  - Reduced number of SQL joins when retrieving data
  - Increased AWS resources to better handle traffic.

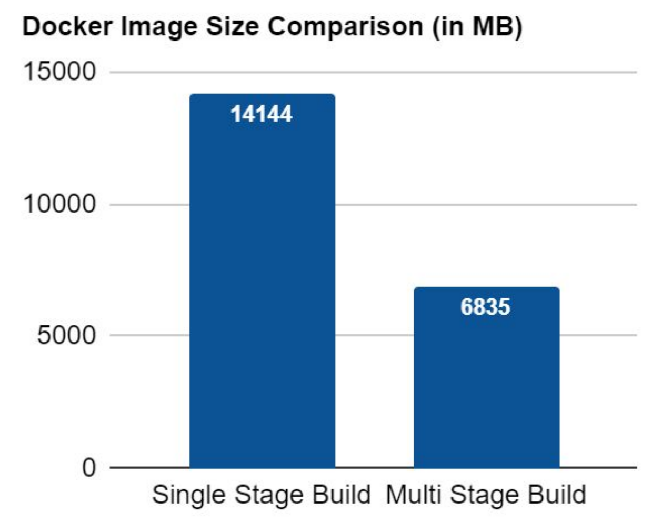
Observation After Optimization

- IRIS was **able to handle 50 users effectively,** optimized speed and time taken.

Containerization and Re-Architecture

Key Components	Single Stage	Multi Stage
Build Tools	✓	✗
Poetry	✓	✗
Development Dependencies	✓	✗
Temporary Files	✓	✗
Application Code	✓	✓
Runtime Dependencies	✓	✓
Size of Final Image (MB)	14144	6835

✓ Included ✗ Not Included in Final Image



By using multi-stage build, docker image size decreased by **~50%** compared to single-stage build.

Conclusion

In summary, our enhancement project has **successfully evaluated IRIS and identified bottlenecks** to support growing operational needs, like increased user loads. Transitioning to a containerized architecture has **improved scalability and reliability**. This change facilitates **horizontal scaling, ensures consistency** across environments, **faster rollbacks** if an update fails, and **potentially lowers costs** with pay-as-you-go pricing model.

Additionally, the two last objectives not shown in this poster have been achieved: improved model results for better article insights and a dashboard for analyzing model performance in IRIS.