# Live Dashboard - Real-Time AppsFlyer Campaign Analysis

## Agenda

### Architecture

- Overview of data flow
- Tech Stack
- End result

### Environment Setup

- Docker-compose setup for Kafka, MongoDB, NiFi, Spark, JupyterLab
- Port mappings and access
- Starting all services

### Real-Time Data Pipeline

- Streaming from NiFi to Kafka
- Kafka consumer with Python
- Aggregation logic and MongoDB persistence

### Dashboard

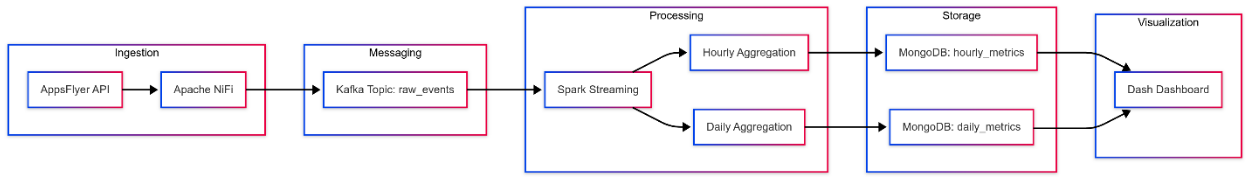
- Visualization with Streamlit
- Reading from MongoDB
- Auto-refresh and campaign insights

### Code walkthrough

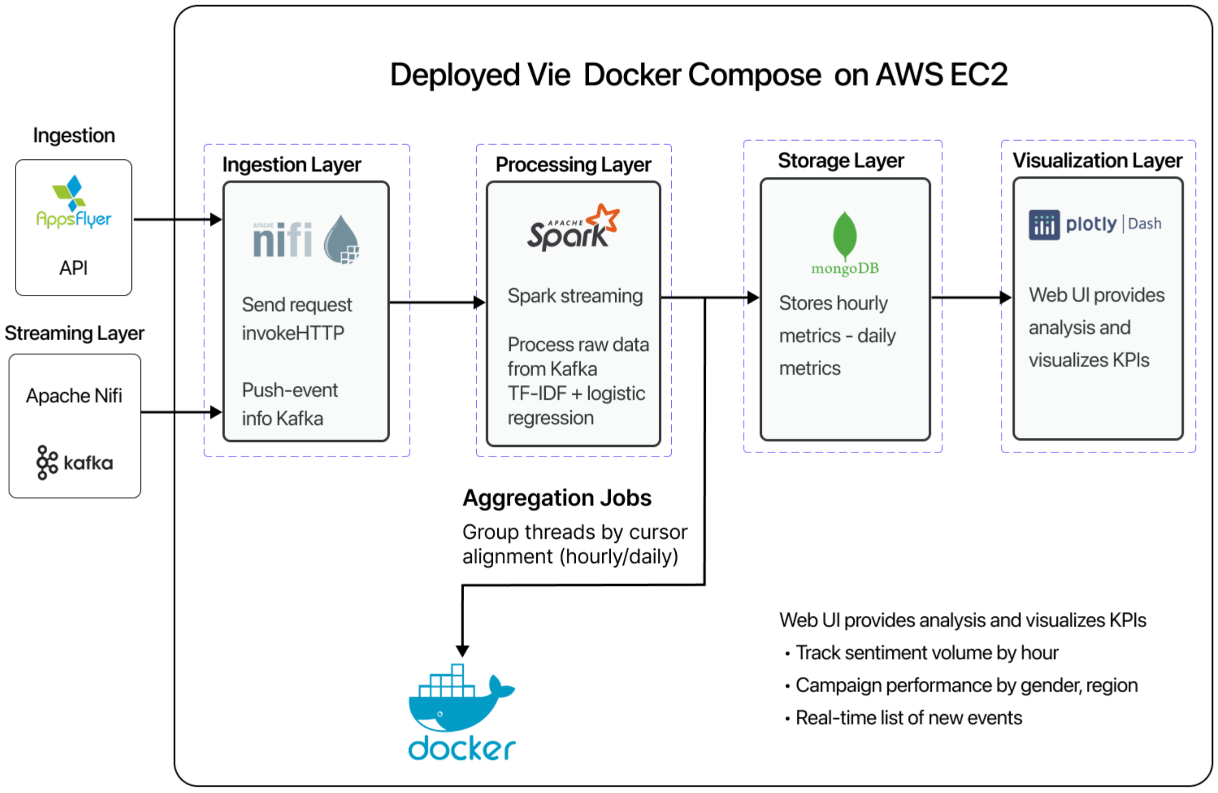
- Docker Compose
- realtime_pipeline.py
- dashboard.py

## Architecture

### Overview of data flow



## Architecture Diagram


Deployed Vie Docker Compose on AWS EC2
Ingestion
AppsFlyer
API
Streaming Layer
Apache Nifi
kafka
Ingestion Layer
nifi
Send request invokeHTTP
Push-event info Kafka
Processing Layer
Spark
Spark streaming
Process raw data from Kafka TF-IDF + logistic regression
Storage Layer
mongoDB
Stores hourly metrics - daily metrics
Visualization Layer
plotly | Dash
Web UI provides analysis and visualizes KPIs
Aggregation Jobs
Group threads by cursor alignment (hourly/daily)
docker
Web UI provides analysis and visualizes KPIs
• Track sentiment volume by hour
• Campaign performance by gender, region
• Real-time list of new events

## Tech Stack

- Apache NiFi
- Kafka
- MongoDB
- Python
- Streamlit
- Docker / docker-compose

## End result

**Real-Time Aggregation in MongoDB**

- Hourly and daily campaign counts stored

**Live Dashboard**

- Auto-updating charts and tables

# Environment Setup

## Docker-compose setup

**Docker Compose file includes:**

- JupyterLab, Spark Master/Worker, Kafka, Zookeeper, NiFi, MongoDB, Mongo Express

## Port mappings and access

- Kafka exposed on port 9092 (external)
- MongoDB on port 27017
- NiFi: http://localhost:8443
- JupyterLab: http://localhost:4888
- Mongo Express: http://localhost:4141

## Starting all services

```
docker-compose up -d
```

To access individual containers:

```
docker exec -it kafka bash
```

Ensure Kafka topic exists:

```
kafka-topics.sh --create --topic appsflyer-events --bootstrap-server
localhost:9092 --partitions 1 --replication-factor 1
```

# Real-Time Data Pipeline

## Streaming from NiFi to Kafka

- NiFi processors extract and publish AppsFlyer-like events to Kafka topic

## Kafka consumer with Python

- Consumes JSON events from Kafka
- Aggregates installs by hour/day
- Periodically writes aggregates to MongoDB

## MongoDB persistence

- Collections:
  - `hourly_events`: stores hourly installs per campaign
  - `daily_events`: stores daily installs per campaign

## Dashboard

### Visualization with Streamlit

- Reads from MongoDB
- Auto-refresh every 60 seconds
- View hourly or daily campaign performance

### Main Features

- Filter by time
- Sort campaigns by number of installs
- Live data visualization with Plotly

## Code walkthrough

### docker-compose.yml

- Services: NiFi, Kafka, MongoDB, JupyterLab, Spark, Mongo Express
- Port mappings and volumes

### realtime_pipeline.py

- KafkaConsumer setup
- Hourly/daily aggregation logic
- Periodic flush to MongoDB

### dashboard.py

- MongoDB queries
- Streamlit layout and controls
- Plotly charts with auto-refresh

*This document outlines task A05: setting up a full real-time data pipeline for AppsFlyer campaign analytics with automated visualization and Docker-based orchestration.*