

Viral Images!

System of Integrations



university of
groningen

for the course of Enterprise Application Integration

Authors

Swastik Nayak

Anil Mathew



Filipe Capela

Mark Soelman

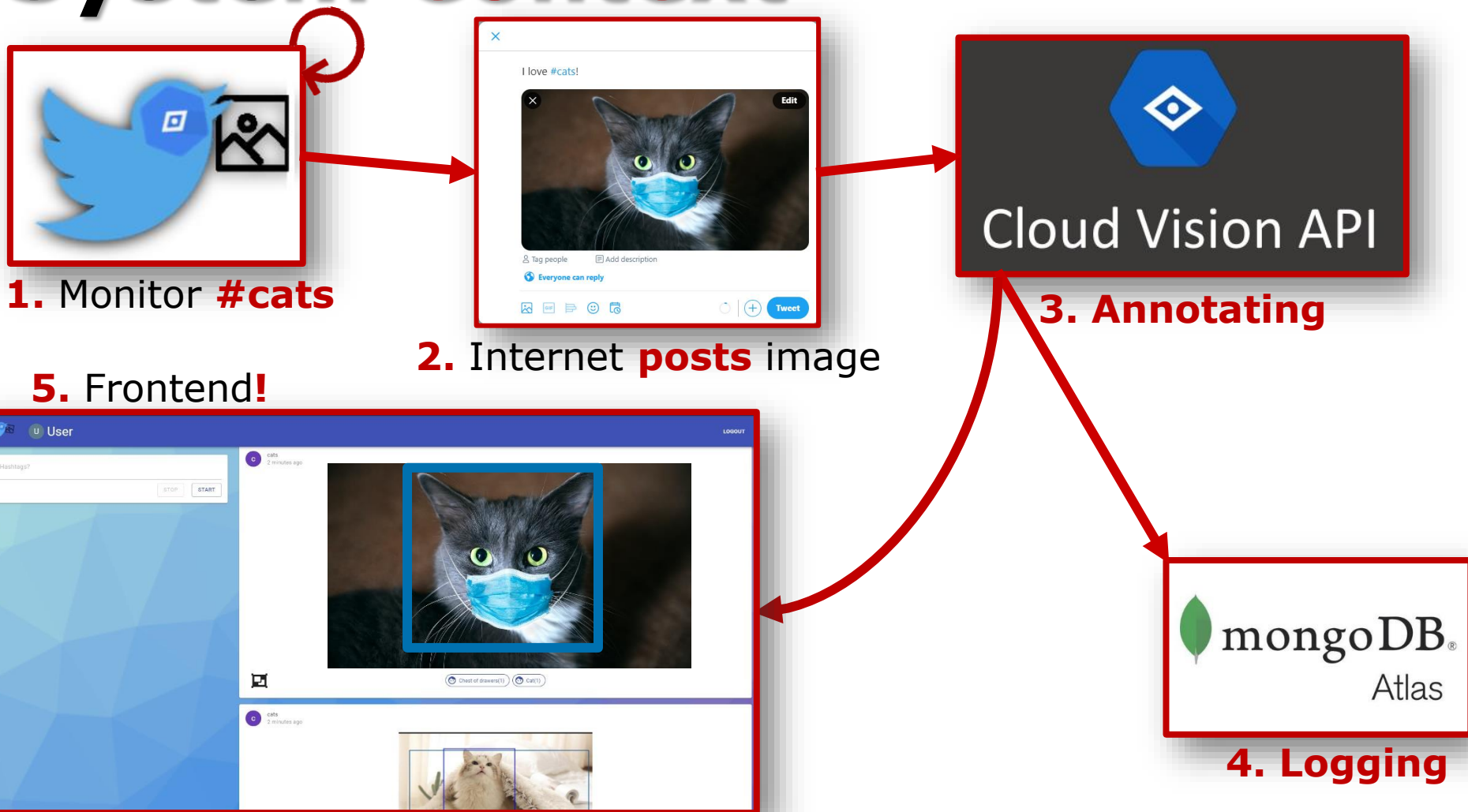


Table of contents

- System Context
- Integrated Systems
- Tiered Architecture
- Design Decisions
- Architectural Diagrams
- Demo

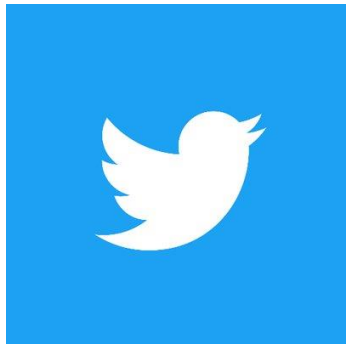


System Context





Integrated Systems



mongoDB®
Atlas

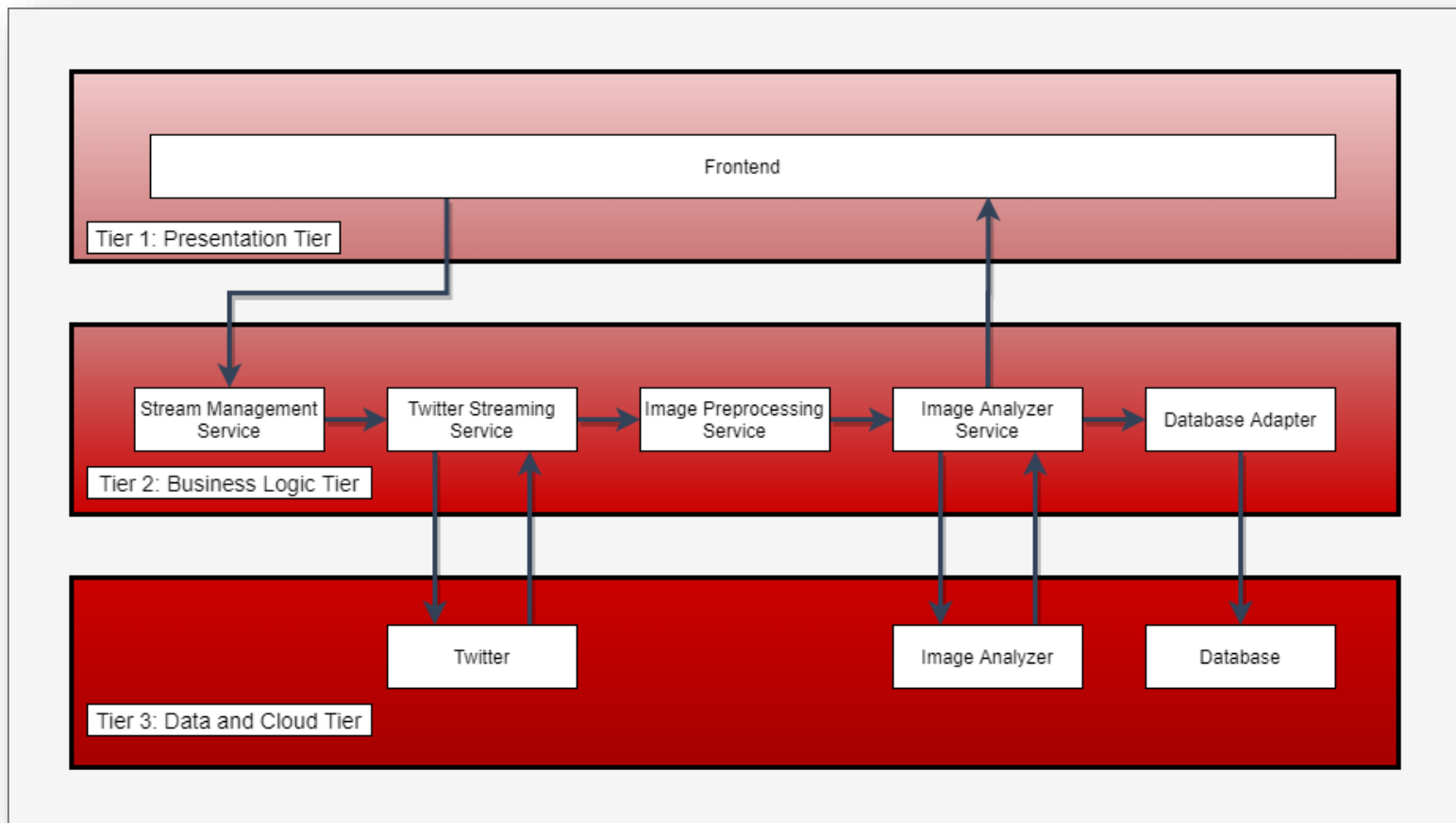


Google Cloud
Vision API





Tiered Architecture





Technologies

Stomp 



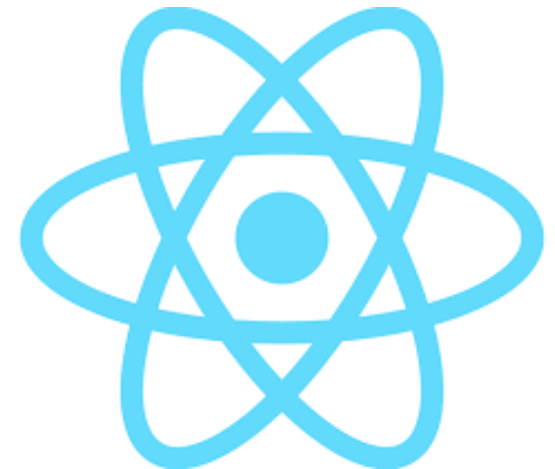
Spring Boot



ACTIVEMQ



docker





Problem #1

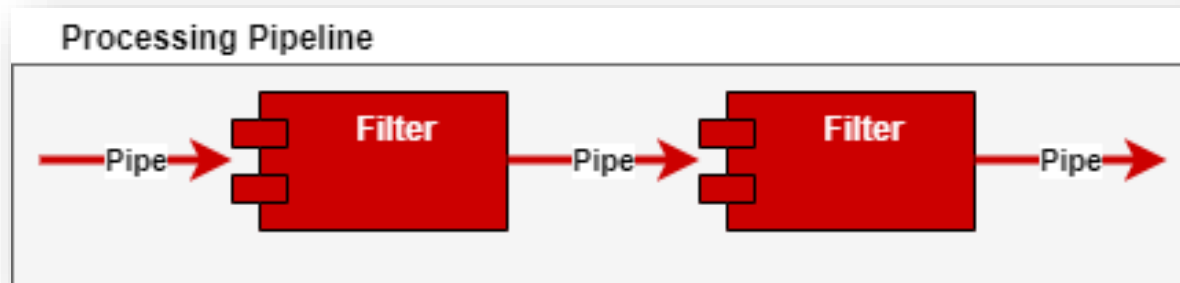
How to get from **Twitter** -> **Output** ?

Assumptions/Constraints

- Multiple processing **stages** required
- Transferred data (images) may be **large**
- Analysis may take a **long time**



Integration Style in-between processing pipeline components (Pipes and Filters)



Chosen

- Pipes and Filters
- Messaging
- Point-to-Point Channel
- Document Messages



Problem #2

When to **start** and **stop** data ingestion from **Twitter**?

Assumptions/Constraints

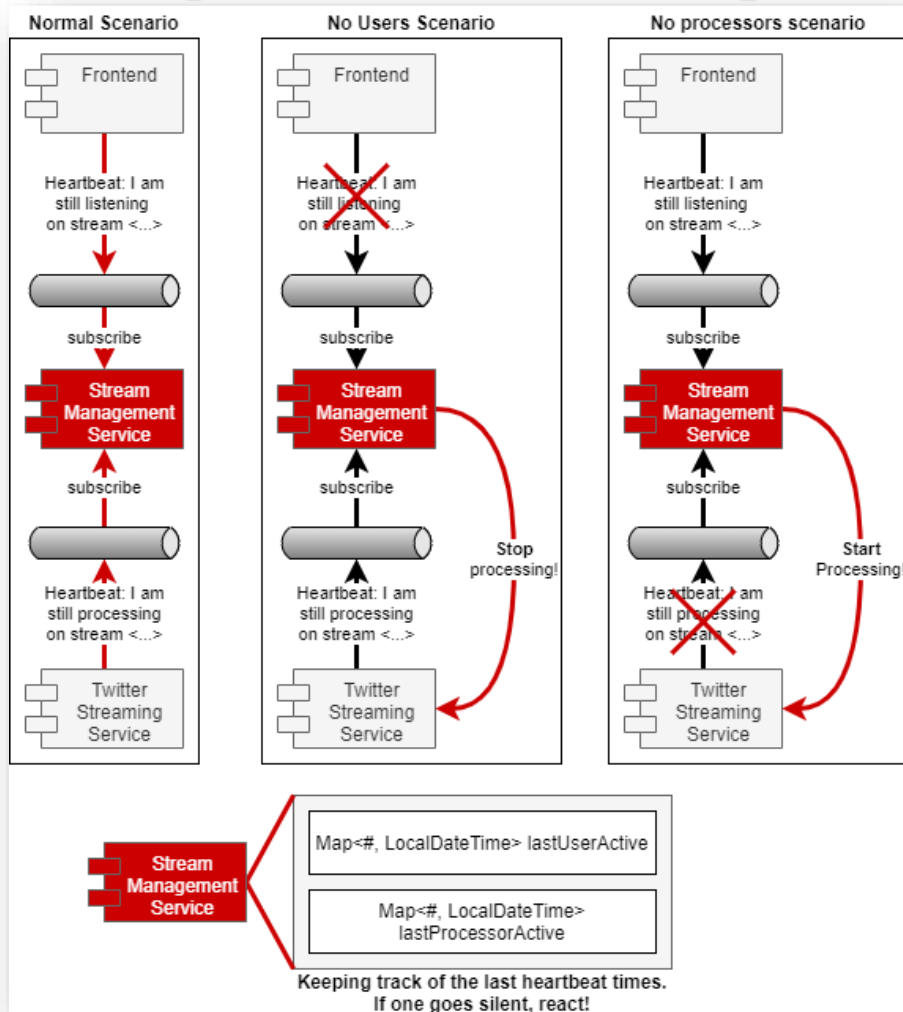
- N users watch #hashtag -> **1 data ingestion** on #hashtag
- 0 users watch #hashtag -> **0 data ingestion** on #hashtag
- Users are **unreliable**. Will not always click "**stop**"
- What if... **Twitter** ingestion app crashes?



Pipeline Lifecycle Management

Chosen

- Observer
- Event Messages
- Splitter
- Command Message
- Correlation Identifier
- Idempotent Receiver
- Content-based Router





Problem #3

Decoupling between **Messaging** <-> **Business Logic**

Assumptions/Constraints

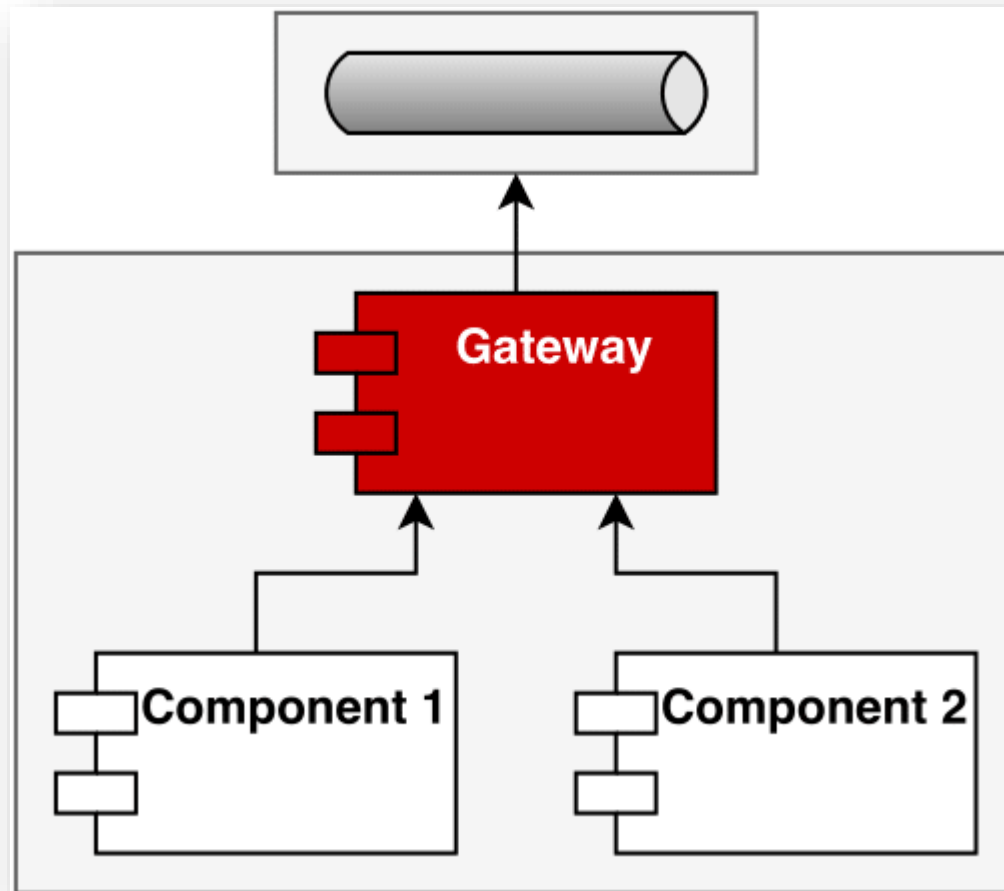
- Business logic should be 100% agnostic to communication
- Messaging System may be replaced



Coupling between code and messaging system (Gateway)

Chosen

- Gateway





Problem #4

Agreement on **Data Format**

Assumptions/Constraints

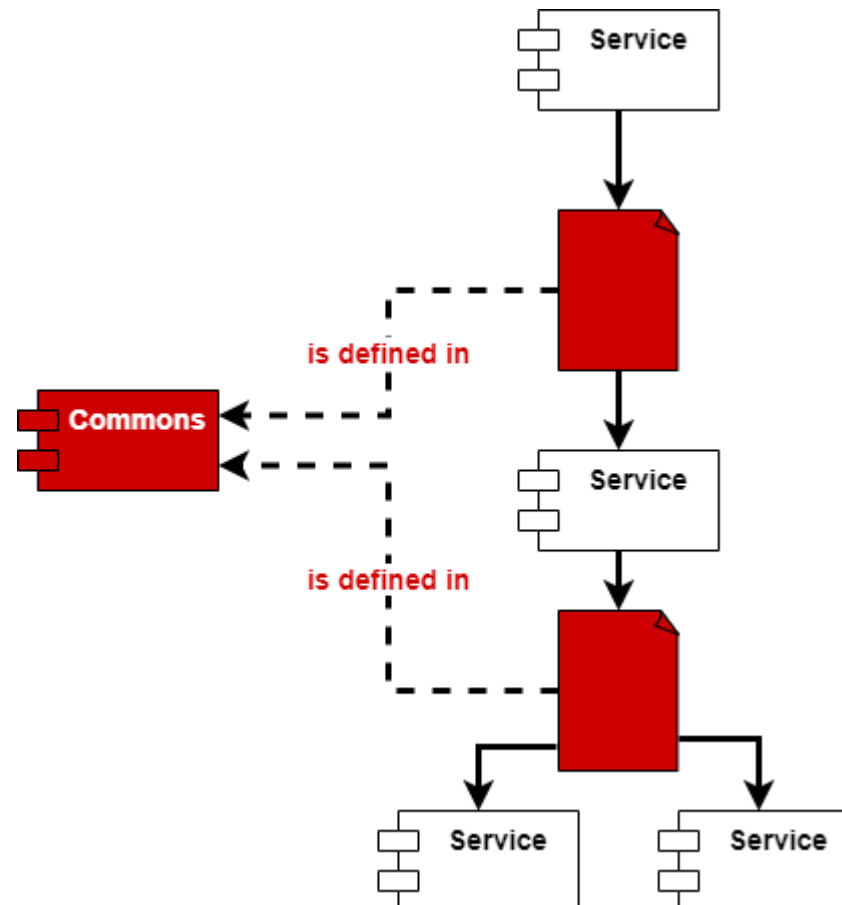
- Components communicate data
- Different data formats used throughout application
- Applications may need to modify data formats
- Applications should allow independent compilation



Cross-application data model sharing (Canonical Data Model)

Chosen

- Canonical Data Format



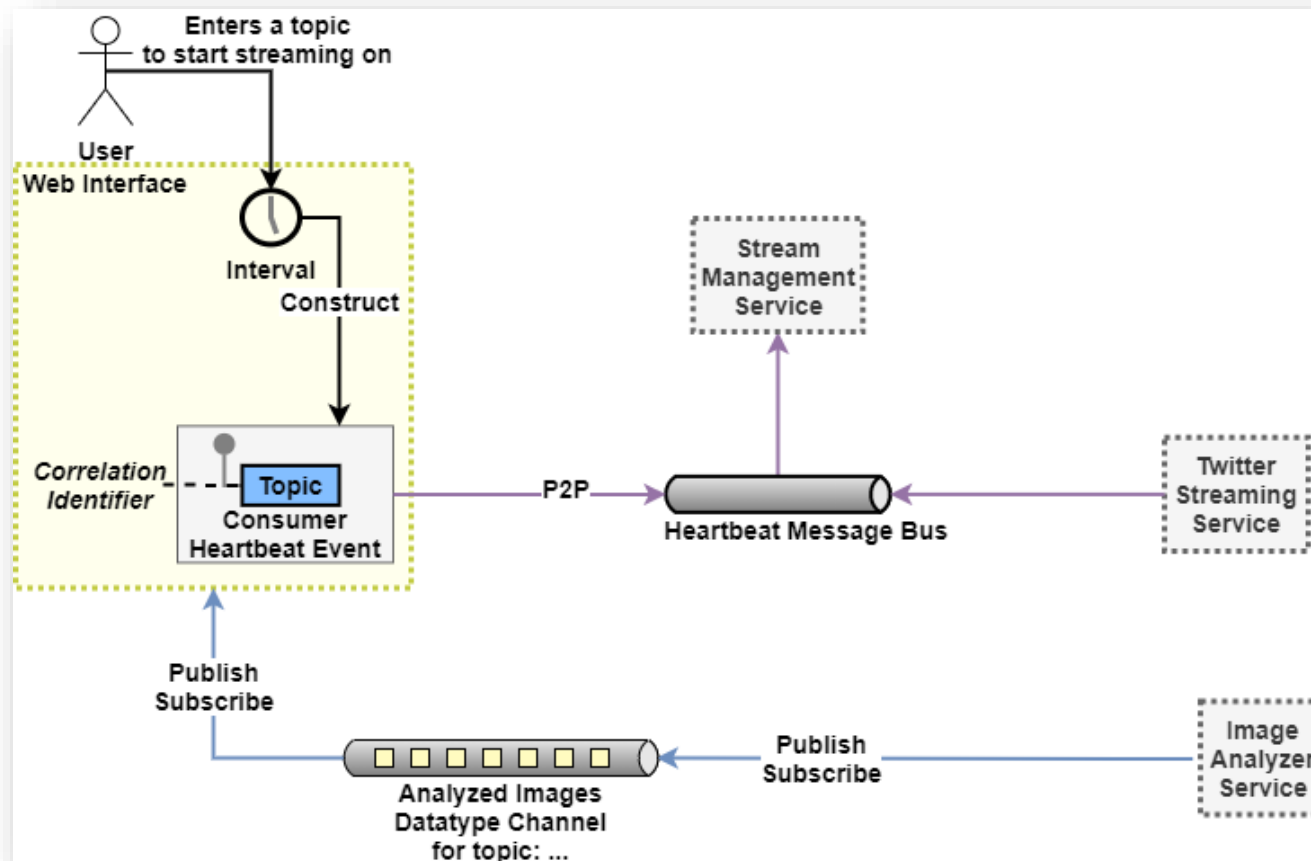


Design Decisions

- **Integration** Style in-between processing pipeline components (Pipes and Filters)
- Pipeline Lifecycle **Management**
- **Coupling** between code and messaging system (Gateway)
- Cross-application **data model** sharing (Canonical Data Model)
- **Delivering Analysis** Results to the Frontend (Publish-Subscribe)
- **Filtering** results and removing noise (Content Filter)
- Object **Detection** in images (Content Enricher)

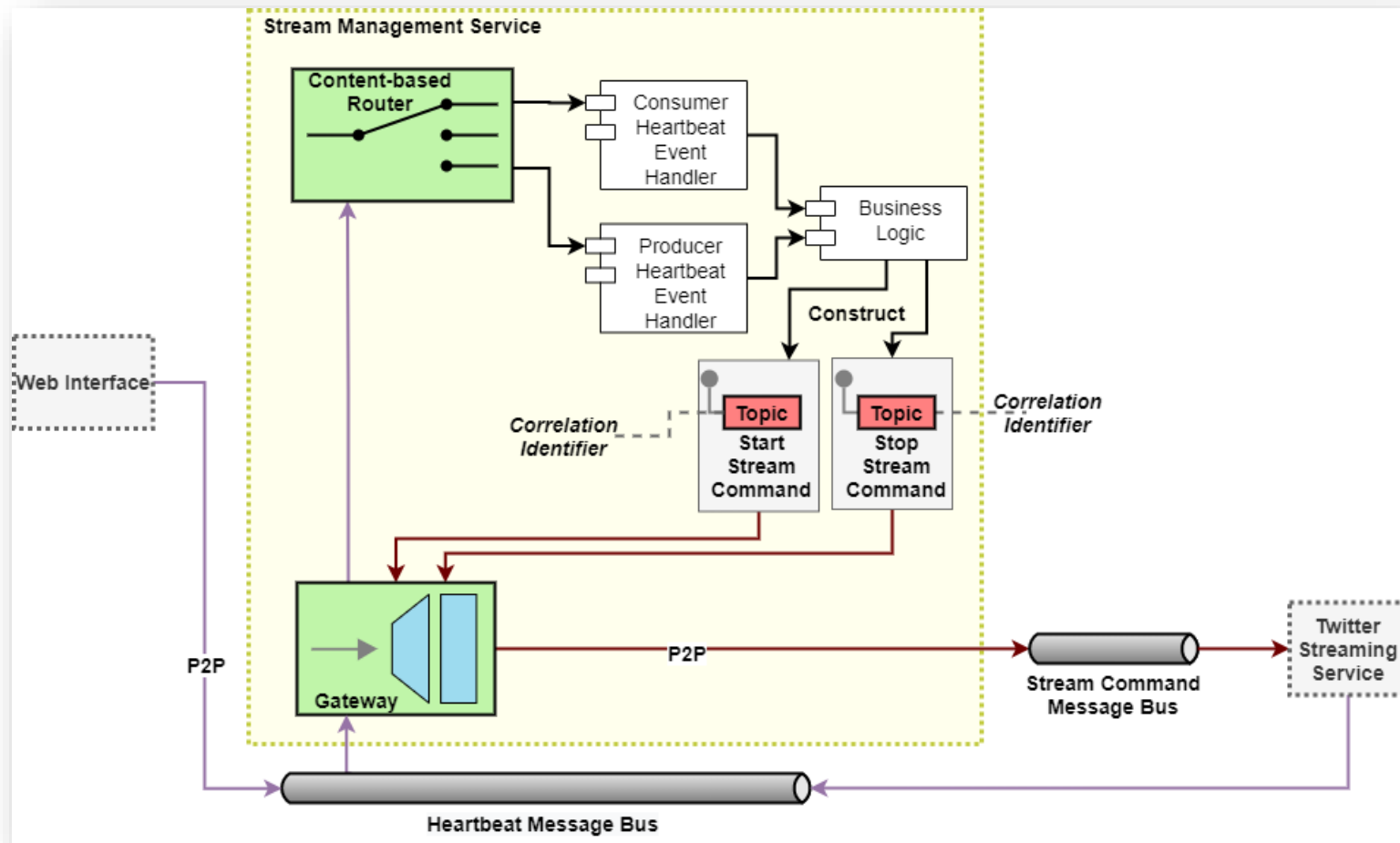


Architectural Diagram (Web Interface)



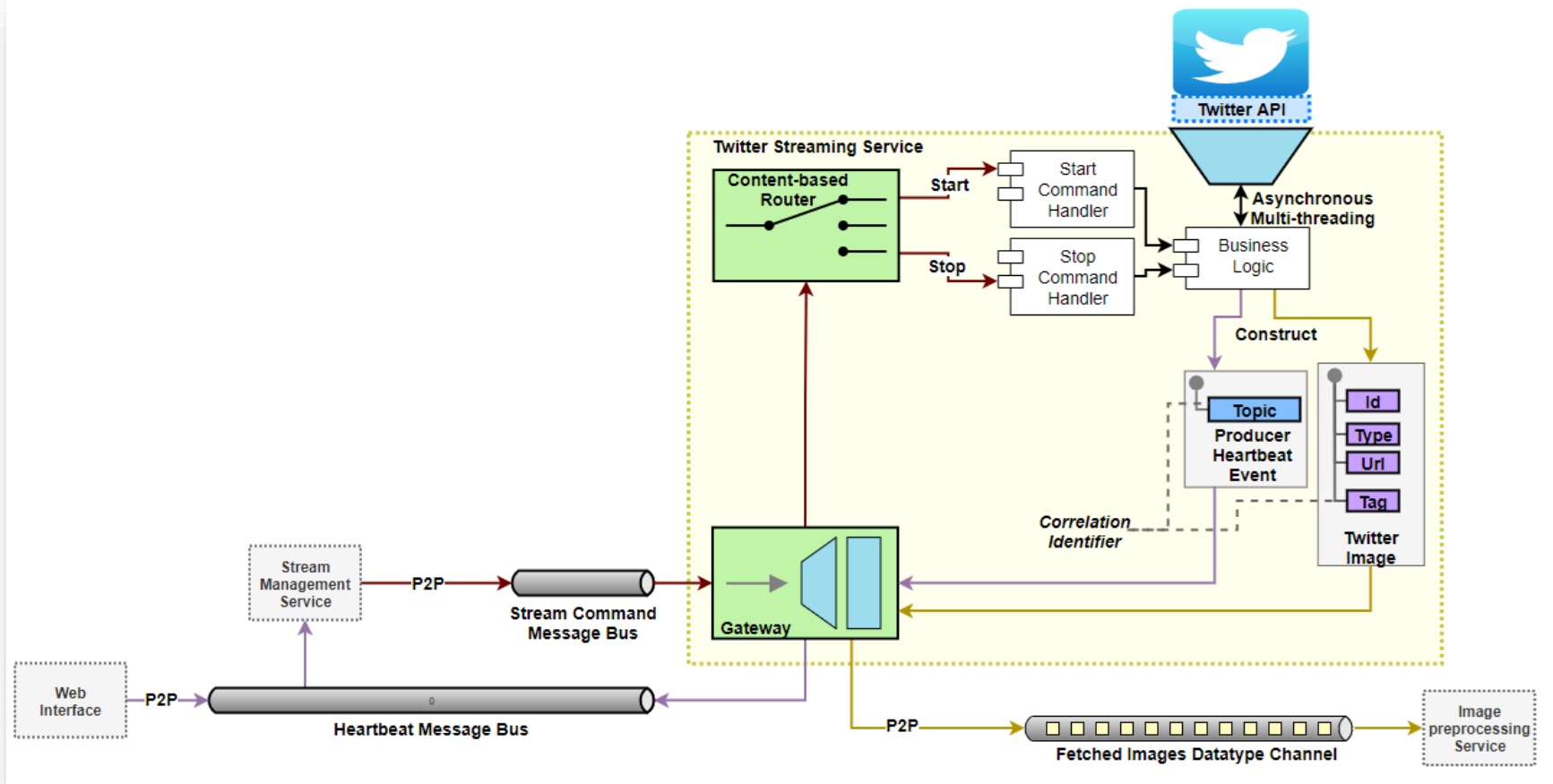


Architectural Diagram (Stream Management Service)



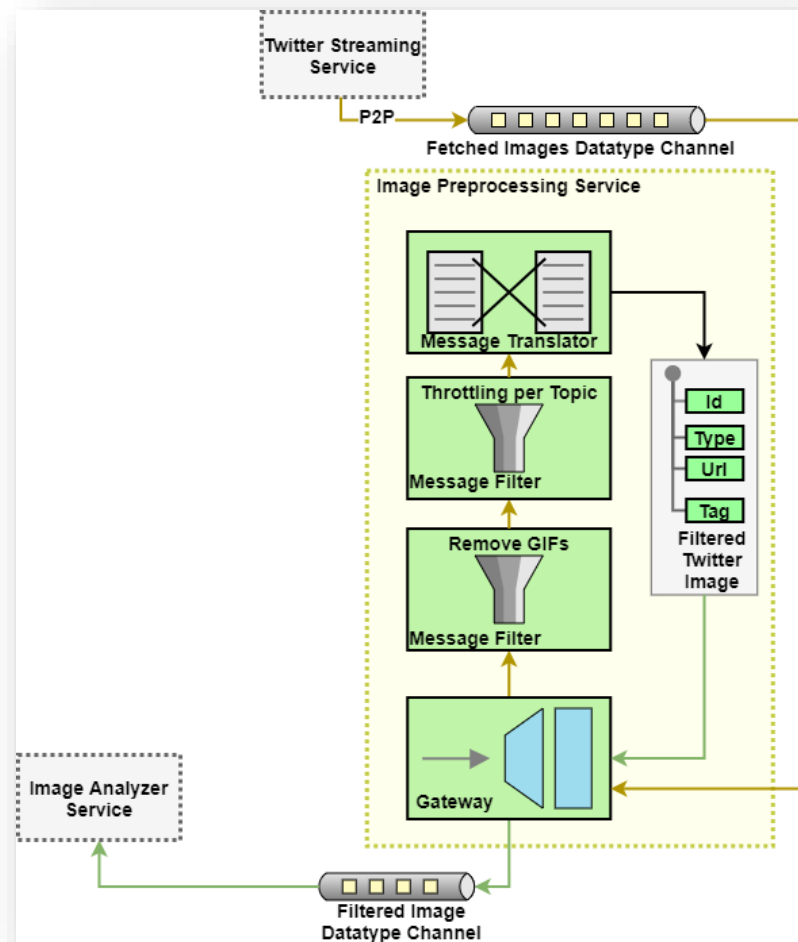


Architectural Diagram (Twitter Streaming Service)



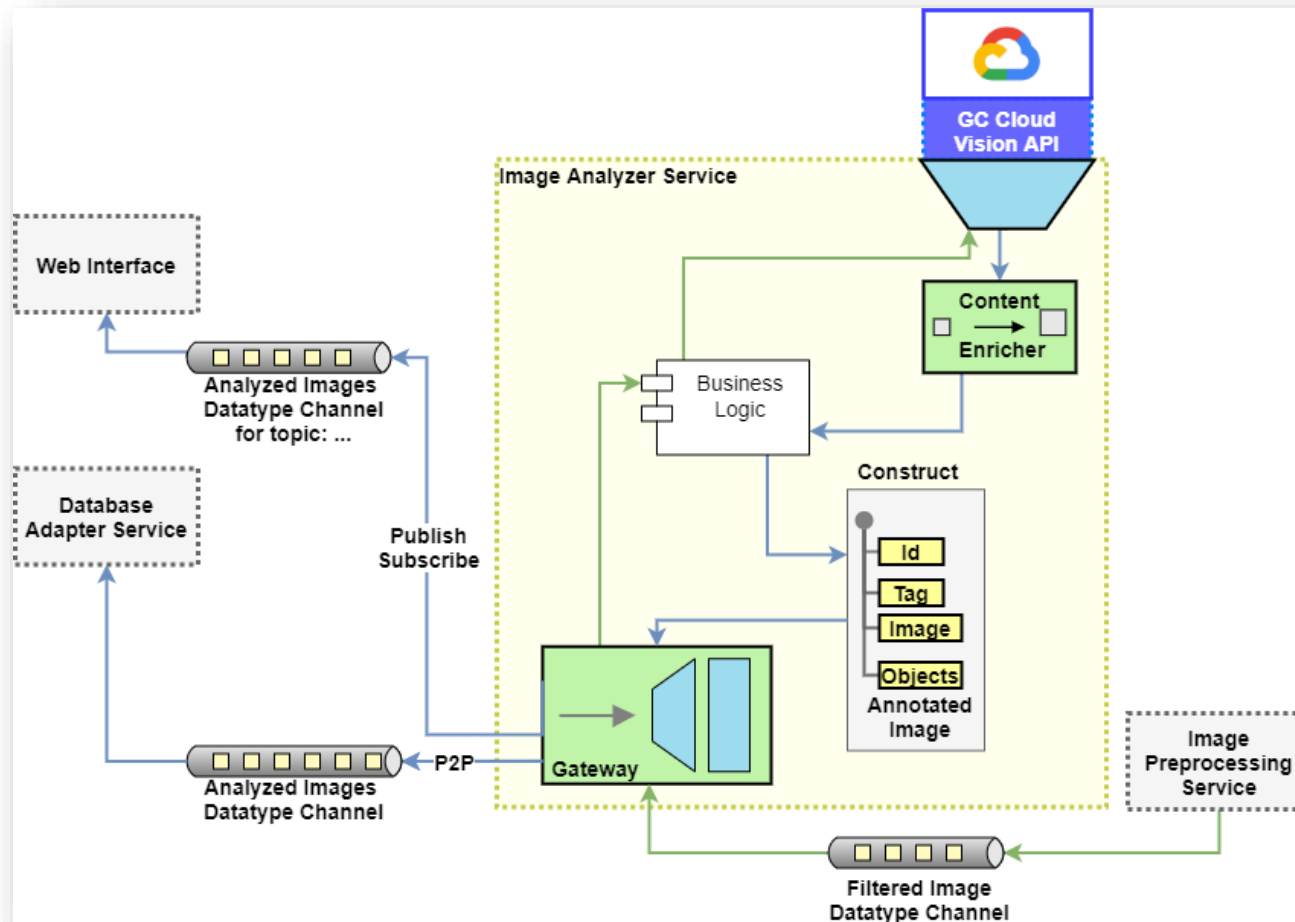


Architectural Diagram (Image Preprocessing Service)



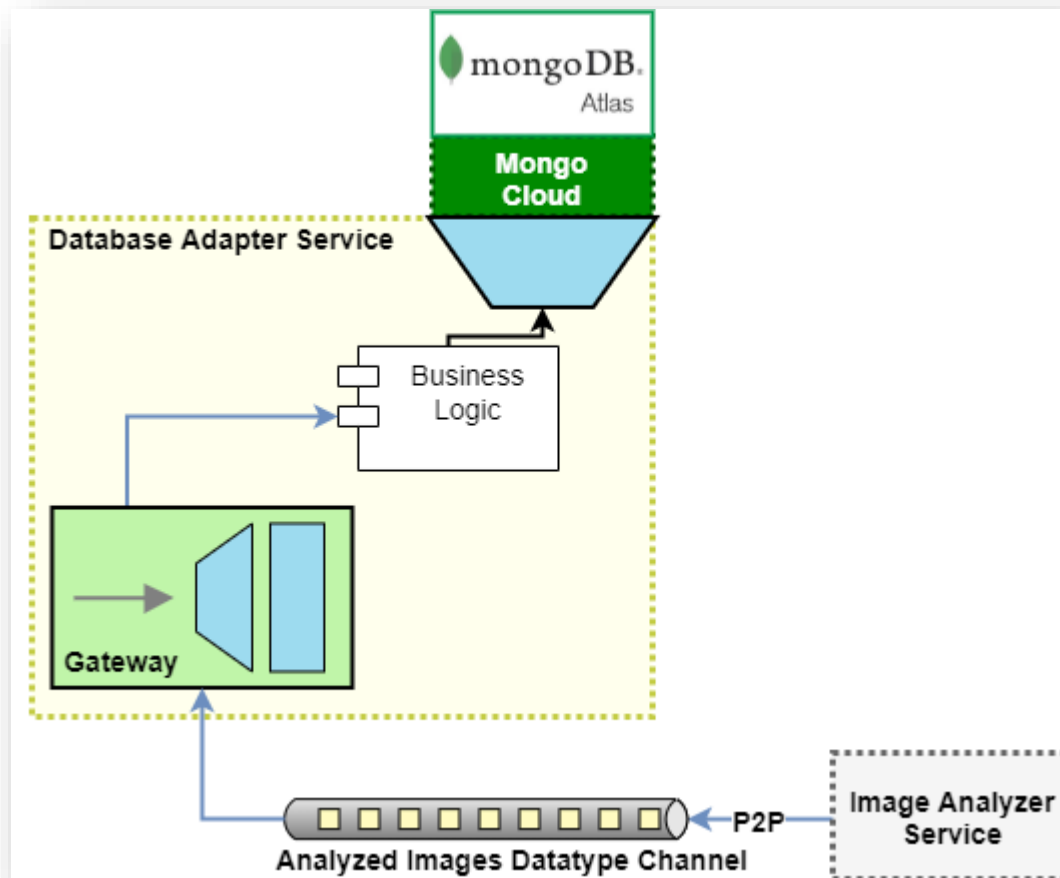


Architectural Diagram (Image Analyzer Service)





Architectural Diagram (Database Adapter Service)





Architectural Diagram

Final presentation of our group's project

Monday, January 11, 2021 | 22

