Data Architecture Design

Tommy Lim, 2024

Table of contents

Exploration: Self-managed and managed solutions in the market

- Message-passing / “event bus”:
  - Apache Kafka (Kafka Producer API)
  - AWS SQS/SNS
  - GCP Pub/Sub
  - Azure Service Bus
- Stream processing layer:
  - Apache Kafka (Kafka Streams)
  - Apache Flink (GCP Dataproc, Amazon Managed Service (MSK), HDInsight on AKS)
  - Apache Spark Streaming (GCP Dataproc, AWS EMR, Azure databricks)
  - AWS Kinesis
- Storage:
  - Short/medium term storage:
    - Redis (GCP Memorystore, AWS Elasticache, Azure cache)
    - Postgres (GCP Cloud SQL, AWS RDS, Azure database)
    - MySQL
    - MongoDB
    - DynamoDB
    - AWS Redshift
  - Long term data storage:
    - HDFS or other filesystem
    - Cloud storage (GCS, S3, Azure Blob Storage)
- Visualization / analytics layer, eg.
  - Real-time analytics:
    - Apache Druid
    - Apache Flink (Flink SQL)
  - Batch analytics:
    - GCP BigQuery
    - AWS Redshift
  - Visualization:

- Grafana
- Metabase
- Tableau
- Power BI
- Looker Studio
- Custom, eg. Node.js (Sockets.io, Highcharts)

There are also end-to-end fully managed solutions in the market, such as Confluent Cloud, Databricks, Snowflake.

What are the assumptions and constraints

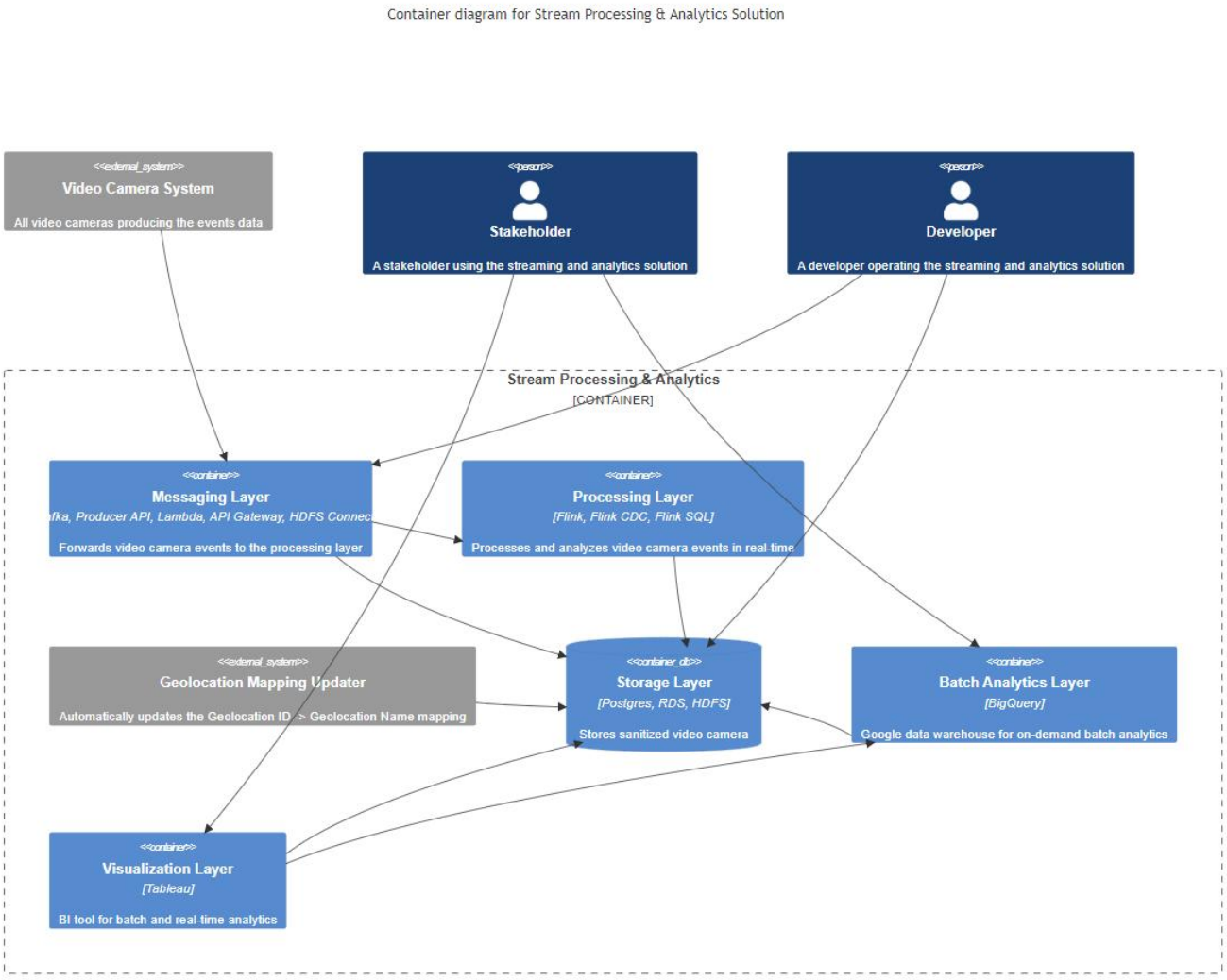
Based on the fact that potentially sensitive data from video camera needs to be processed and with the possible data security concerns in HTX context, I will make the following assumptions before deciding on a tech stack:

1. The video cameras are physically limited to a certain locale (ie. Singapore only).
2. Each video camera will capture events with the same schema, similar to Dataset A's schema.
3. Dataset B (mapping of geolocation ID to geolocation names) can be independently managed and is relatively small.
4. We have an acceptable encryption/decryption method for PII (personally identifiable information) or other sensitive data, if we need to mask data on one of the cloud providers.
5. We have sufficient resources and project timeframe to bring this solution from proof-of-concept to production.

What is the recommended tech stack

Tech stack	
Message-passing / "event-bus"	Kafka Producer API (AWS Lambda, Amazon API Gateway)
Stream processing	Flink CDC (AWS EC2 or EKS)
Storage	Postgres (AWS RDS) HDFS (on-premise) (via Kafka HDFS Connector)
Real-time analytics	Flink SQL (AWS EC2 or EKS) (Optional) SNS or HTTP webooks for custom alerts
Batch analytics	BigQuery (Postgres -> BigQuery via Data Transfer Service)
Visualization	Tableau (connected to Postgres and BigQuery)

Architecture diagram (C4 Container diagram):



Why this tech stack?

Message-passing / ”event-bus”: Kafka has a strong reputation in the data industry, and can be deployed on any cloud infrastructure. This helps to significantly reduce our vendor lock-in, and allows for some flexibility to change clouds for pricing considerations or security considerations).

Stream processing: From my research and exposure in previous roles, Flink is industry standard for the typical needs for stream processing- windowing, joining, aggregation. It can be hosted on cloud or on-premise (eg. by Confluent). One of the real-time operations I would expect to be executed through Flink is the anonymize any PII (personally identifiable information) before writing to Postgres database.

Real-time analytics: Flink SQL allows for SQL-like queries on realtime data (in this case, data in Postgres database). This may not be required for the immediate use-case, but its good to have this option, eg. when items detected related to crime need to be reported immediately.

Storage: Postgres is chosen as the primary database and sink from Flink CDC because it will perform well in the situation where we are generally going to be write-heavy; presumably because video camera data will constantly stream in 24/7.
We will also store the raw video camera events data in an on-premise HDFS, since this is sensitive data that may need to be retrieved even years later (possibly for crime evidence, audits, etc.) If infrastructure and infosec is well planned for, on-premise will reduce our vulnerability to data breaches since we would have more control over how the data is replicated (if at all), and where exactly it is stored (the physical location of the on-premise hardware).

Batch analytics: If ingress costs into GCP is tolerable, BigQuery is the ideal option for batch analysis, because of its ubiquity among data analysts, data scientists, data product managers). DTS is Google's recommended way to transfer from GCS to BigQuery (and other sinks). Invoking DTS in Cloud Function (equivalent of a Lambda) would be straightforward. In contrast, setting up Airflow DAGs + GCS to BigQuery operators is likely to be overkill.
Once the anonymised data is in BigQuery, it can be connected to Tableau or any dashboard / BI tool with a BigQuery connector.

Visualization: Though important in its own right, from an architecture perspective this is the least costly component (in terms of effort) to swap out. I chose Tableau not based on personal experience but based on widespread industry acceptance. It also fulfills the potential future need to support real-time dashboards (ie. via Postgres connector).

Tradeoffs of this architecture

+ Partially cloud agnostic since it relies heavily on open-source software which can generally be deployed to any cloud
+ True raw data is stored in an on-premise HDFS (via Kafka HDFS Connector); this enables higher level of security and a source of truth when needed
+ In terms of data privacy, we can confidently sanitize/anonymize/filter out whatever we don't want exposed to any cloud provider. Because the source of truth is maintained in the on-premise HDFS
+ Deduplication is handled out of the box by Flink CDC
+ Costs are in theory minimized by deduplicating early in the data flow
+ Real-time analysis is supported via Flink SQL
+ BigQuery for batch analysis easy to learn and a popular choice; among analysts, data scientists, and PMs.

- More infrastructure and self-managed disaster recovery is needed, since we are intentionally not going for end-to-end solutions, eg. Snowflake / Databricks
- This choice of stack leans towards complex: AWS, GCP, and on-premise are all used
- Less negotiation power for cloud prices in the long term, compared to if we locked-in with a single cloud provider or managed solution
- By choosing to partially expose the data to GCP, we have an even bigger need to manage schema evolution, masking of data, being compliant in terms of deleting data both on-premise and in cloud (eg. If there is a case where sensitive data is requested to be deleted by authorities)
- From experience, it may be harder to get discounts from GCP compared to AWS. We would need to be conscious of both BigQuery storage and compute costs (ie. Planning for data archival / deletion and planning for BigQuery slot reservations)

- No data contracts / data quality assertions / data quality monitoring out of the box. If we want it, then we have to integratae tools like dbt and Monte Carlo.

Further considerations

- If HTX is already heavily locked-in to Azure and is satisfied with Azure's security certifications, should we consider a full Azure-based stack?
  - Maybe. From experience (albeit with AWS, not Azure), negotiating for cloud discounts is much easier when a larger part of the company's architecture is using the same cloud. It also depends on the hiring strategy (do we think more graduates or applicants will be familiar with Azure?) But if pricing is not a high priority and infrastructure support can be counted on, I would favour sticking to open-source and more self-managed solutions.

- Why both AWS and GCP? Won't that make it harder for developers to be competent at maintaining/expanding all parts of this solution?
  - AWS is a reasonable choice for production-ready software: from cloud market share, economies of scale achieved by Amazon, and many companies use AWS for production (at least compared to GCP. I have no experience with Azure)
  - GCP is good mainly for BigQuery. BigQuery is one of the best parts of GCP and is also highly recognized in the data industry; almost any dashboard provider will want to support visualization of BigQuery data. It will be easy to hire people who can write good SQL in BigQuery, compared to RDBMS, NoSQL, GraphSQL, kSQL, etc. Stakeholders might think they need very fancy tools, but they will be surprised at what you can do with basic data modelling / schema design and a straightforward tech stack. And now with BigQuery ML, even fancier analysis can be done. Most importantly, there currently is no strong equivalent in AWS (unless we think Redshift is going to become much better or popular).

- Why not 100% on-premise or in-house solutions?
  - Given the popularity of Cloud in both schools and in the tech industry, we may struggle because of the availability of developers and relevant expertise.
  - It will be impractical and unrealistic to think we can challenge the production quality of N years of these open-source solutions. I'm confident that the requirements can be fulfilled without 100% on-premise and in-house solutions.
  - If we find weaknesses of any aspects of the tech stack, we should still consider an in-house solution. For example, if there is some standardization around the video camera hardware / software, perhaps its possible to create a better event-bus. (eg. Something that is resilient to internet outage / VPC outage?)