While I was working with a big financial bank, I was part of the IT infrastructure team but from the development side where we were responsible for building web-based tools and reporting for all kinds of service requests, monitoring, etc. We had close to 10 systems all recently build with micro-services styled architecture. The biggest problem with these individual silos was that our business sponsor and the business executive did not have a full 360 view for all the systems together, each system would generate its reports. So, we were tasked to build a data hub kind of solution, so that we can get a unified view of the data.

So, in this case, we did want to have any impact on the existing application services, so had decided to leverage KAFKA to act as a streaming platform. We had leveraged the concept of Change Data Capture (CDC) for database and Kafka connect. In this approach, any database changes like inserts, updates, and deletes are available through Kafka connect, which then pushes CDC as a full message to KAFKA streaming platform where it could be consumed by subscribing application, in this case when we started we had only one subscriber which was the job which would pick up this message, transform it and push it to the centralized operational data store. It looked something like this.

Now we had built data services on top of the centralized data store, these services were specific to business entities, earlier across 10 system we had common entities which had different attributes in different systems, with the new approach of we had a single unified view of that business entity. On top of these services for certain analytics and reporting, we had built a layer of graph QL which gave flexibility of querying underlying APIs across business entities.

Advantages of this approach

1. Data is available to Centralized data store immediately, there is very less latency.

2. Apart from centralized data store as a consumer, it opens possibilities for other use cases, like we can do real-time monitoring through the mutation of data streams. You can build n number of subscribers to listen to Kafka streams.

3. No existing application code changes.

Disadvantages of this approach

1. Schema changes from the source system impact the underlying systems, so they need to be handled synchronously.

2. Change data capture indicates just the change, it does not indicate how data changed, that logic needs to be built in the consumer of the Kafka message, in this case, the job.

3. As the CDC feature is from the database side, it could entail a slight performance impact.

References

Kreps, J. (2015, Feb). *Putting Apache Kafka To Use: A Practical Guide to Building an Event Streaming*

*Platform.* Retrieved from confluent.io: https://www.confluent.io/blog/event-streaming-platform-1/

Moffatt, R. (2018, March 16). *No More Silos: How to Integrate Your Databases with Apache Kafka and*

*CDC*. Retrieved from confluent.io: https://www.confluent.io/blog/no-more-silos-how-to-integrate-your-databases-with-apache-kafka-and-cdc/

*Change Data Capture – What, Why, How.* Retrieved from superlogo.com:

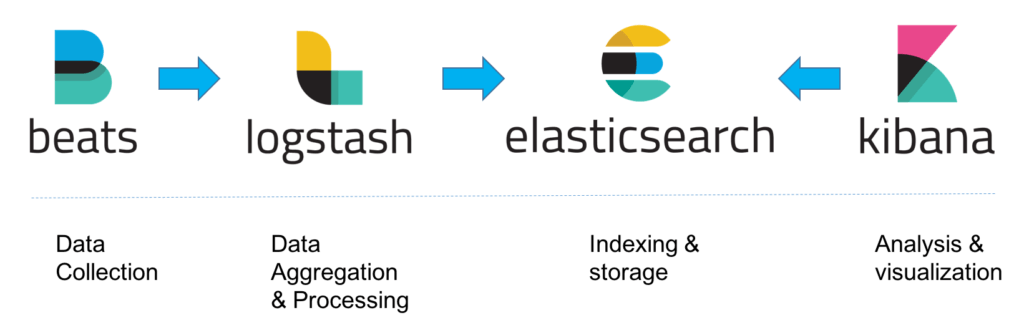
https://supergloo.com/streaming/change-data-ca; pture/

*GraphQL is the better REST.* Retrieved from HOW TO GRAPHQL:

https://www.howtographql.com/basics/1-graphql-is-the-better-rest/

The problem with the integration hub diagram which you have depicted is that multiple applications, storing data in a single data store is simply not realistic in the current age of micro-services. I think you have covered as part of the weaknesses about this integration approach but wanted to elaborate it further. Every application would have a different set of entities and attributes associated with the entities, so pushing the data common data store and then consuming becomes a challenge. Another challenge I see is the new enhancements to the application which will result in database schema change, this must be handled seamlessly in such a manner that it does not impact other applications, this would become a big challenge. I agree with your approach of staging layer between applications, but then I think that would not scale as well, as you are doing transformation between combinations of applications and still storing data in the single data store, this could also lead to redundant data. An approach that I think works is every application has its own datastore and then you build a data pipeline to push data to a data warehouse or data lake, the data pipeline ensures that it is taking appropriate business entities and attributes from each data store wherever the data accuracy is high for that entity. This way the data getting pushed to a warehouse in rich in content and accuracy as well.

Interesting point about Mark logic in comparison to elastic search, this made me think why elastic search cannot be a data hub, I have in fact used it in last 2 projects, in one of the scenarios our primary goal was to give search capability on the data that is been stored in the database, database searches are inherently very slow even after indexing and databases do not provide powerful search capabilities as that of elastic search. In the latest project, we have used this to push the data from 2 data stores to create a common elastic index, based on which we can build analytics and reporting using Kibana data visualizations. ELK (Elastic/log stash/Kibana) provides a good data hub option, where data can be ingested from various sources files/databases using log stash, which then pushes data to the elastic search index, then Kibana can be used to build powerful analytics and dashboards. ELK stack is popular for application log capture and analysis. Elastic has RESTful API’s based on which you can build your custom visualizations as well. ELK stack is part of Open data hub. <https://opendatahub.io/news/2018-12-04/open-data-hub-overview.html>



Compared to Marklogic, Elastic search shines in search space, but I must admit some of the capabilities that Marklogic has are not there in elastic search e.g. it lacks the capability of Semantic triple store. Elastic search even though it is open-source, there are certain features which are available in paid version only, you can find the details over here <https://www.elastic.co/subscriptions>