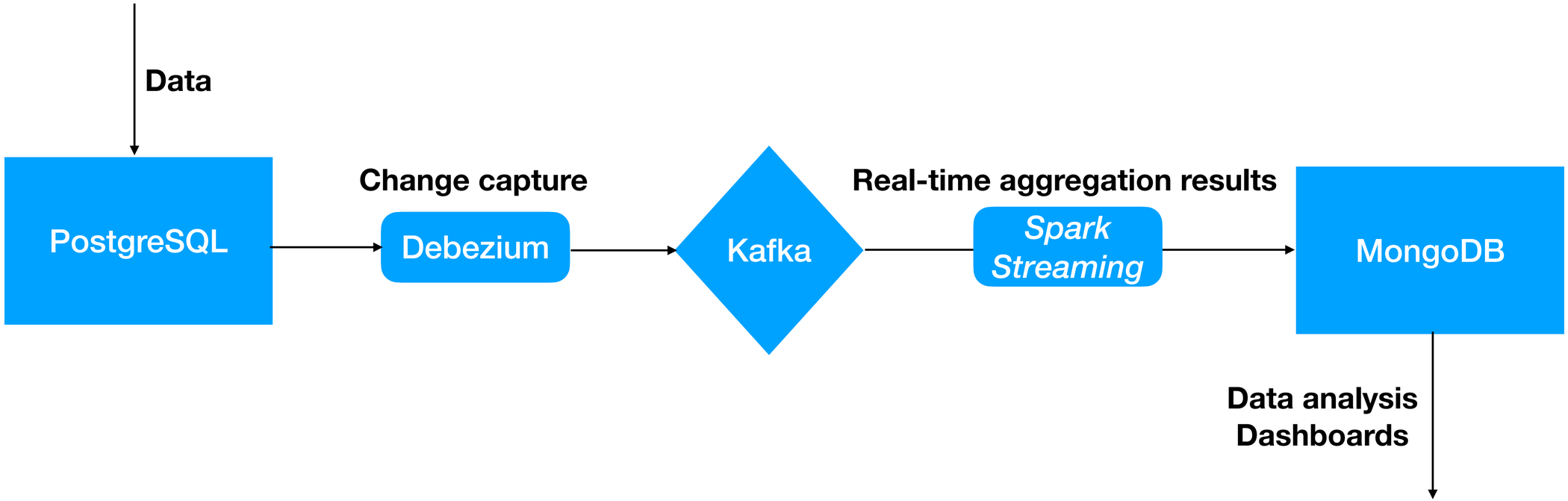
LittleBigCode Technical Test

Solution description

Cherepov Dmitry

Infrastructure



Infrastructure

1) Container running **PostgreSQL**. Init script is provided for the database, where a sample table is created and filled with values. Also, a full replica type is set for the table to make full use of Debezium.

2) Container running **Debezium**. A Postgres Connector is described to track the sample table. All changes are sent to an automatically created Kafka topic.

3) Container running **Kafka** (+ container with **Zookeeper**). A topic is created by Debezium that receives information about all the changes in the sample table.

4) Container running **Spark Streaming application**. The topic is being read, but only new records are being aggregated and written to MongoDB.

5) Container running **MongoDB**. Init script is provided, where admin user is created. Spark Streaming writes aggregated data to a specific collection of the database.

Dashboard thoughts

- Popular solutions (Superset, Grafana) don’t support MongoDB out-of-box, or as a free option, or hard/impossible to containerize
- Possible to use third-party connectors or create our own, where usually a different database is created as the intermediary between MongoDB and the dashboard application
- Possible to use web-based dashboards (access data by API, demonstrate the results in a browser)

Solution limitations/disadvatanges

- Debezium is tracking changes of only one table from Postgres
- The changes are tracked straight from the production database
- Absolutely no security mechanisms implemented
- No restart, or health check mechanisms implemented for the containers
- Hard to scale
- Hard to update containers

Improve solution / run in production

- Replicate main Postgres database and configure Debezium to use the replica
- Use Kubernetes to orchestrate the containers, introduce scalability, updates support, etc.
- Implement security mechanisms (provide environment variables/file or use secrets management solutions)
- Implement error handling, centralised logging
- Implement testing - unit tests, integration tests, end-to-end tests (where possible)
- Implement monitoring and analytics (track the performance and usage)
- If data is not unstructured, use a different target database? E.g. ClickHouse

You’ve been a great audience!