Case Study: Uniformity in CRM and application data

Challenge:

India’s largest Agri Exhibition company Kisan was facing an issue in getting proper leads data from their application admin and the CRM. They were manually saving everyday’s leads from applications to CRM and vice versa. This was leading to multiple challenges like:

1. Due to human errors inconsistent data was on both sides.
2. Time required for manual handling
3. Lead attending duration was high due to that losing of lead chances were high
4. No real time data to see for decision makers from management

Solution:

We created a stand-alone system called the "sync-system" to serve as a go-between for their application and CRM system. Since the two systems are not the same, we constructed a loosely connected system to eliminate the dependencies between them. In order to maintain loose coupling, the system routed the records to the RabbitMQ queue (RQ) (which communicates via the AMQP protocol) rather than directly contacting the CRM APIs. The consumer, or worker, selects the record for processing and validation while listening to the RQ queue continually. They then call the CRM API to sync the data in the CRM. When an API request fails, meaning that the record sync fails, "sync-system" sends it to the failed queue. The failed queue is then periodically synchronised to CRM (every hour, every four hours, depending on the setup).

Likewise, the same procedure takes place when records are added to CRM by the sales agent (i.e., when a lead is obtained through a call or via a third-party system), and vice versa—records are placed in the RQ queue. To save records in the application database, we created a REST API. Records are sent to the application database by these REST APIs after being called by the sync-system,which processes and verifies the data. Similar to this, unsuccessful records are submitted to a failed-queue, where a job is periodically executed to attempt the unsuccessful records.

To prevent endless tries, failing records attempt x (configurable number) times in both scenarios.

The flow explained:

1. Lead is received in the application(signup, form submission)
2. Record sent to queue
3. `sync-system` picks-up the records
4. And send it to CRM by calling it's API
5. Lead record is added to CRM by an agent
6. Lead record sent to queue
7. `sync-system` picks-up the records to process
8. Calls REST api to sync it to application
9. Records synced to application db

Results:

1. Automated the process which removed manual intervention
2. Daily leads followup/attending count increased up to 5x
3. Accurate data across the systems(crm, application)
4. Realtime data is available in CRM and application
5. No inconsistency of records between CRM and application