SWITCHING CLOUD ARCHITECTURES WITH CONFIDENCE AND ZERO DOWNTIME

Summary

The customer was attempting to migrate a loan approval engine from legacy code to a new team, code base, and AWS Lambda architecture. The software was running in production and constituted the core of the company's business, so it was critical that this fundamental rearchitecting would not lead to downtime or logic bugs.

Corporation Type: Credit Lending Technology Company
Industry/Sector: Finance
Project type: Credit Policy software
Technology used: AWS EC2, Haskell, Ruby, Microservices, PostgreSQL, AWS Lambda

Project Requirements

The company had decided that it needed to revamp its legacy code base to implement new features quicker. On the **technical** side, the plan was to:

- Move from a legacy Ruby+Haskell software running on EC2 servers to a serverless architecture on AWS Lambda, implemented from scratch in Haskell.
- Make it a "clean rewrite": The new solution needed to be proven to produce *exactly* the same results as the legacy code, given the same inputs.
- Switch from manual deployment to modern DevOps methodology, introducing Continuous Integration and Continuous Delivery into the project.
- Have zero production downtime and low number of bugs, because any problem in the automatic loan processing would require a costly credit expert review.

At the same time, the company went through significant **organizational** changes:

- **Leadership changes:** Following the closure of one of the company's two office locations, the company switched CEO and CTO.
- **Staffing:** The original developer of the legacy software left the company, and the project manager went on parental leave.
- **Corona Crisis:** Amidst the project, the new Corona virus forced the company to go fully remote.

FP Complete had to jump in to take over engineering leadership, project management, design, and implementation, and assist in the hiring and training of new Haskell engineers.

The Solution

To prevent the Credit Policy engine from drifting into chaos, the company appointed FP Complete to take over the engineering management, project management, and programming until the project had stabilized.

- We arranged a knowledge transfer with the employees leaving the company, ensuring that the current code and legacy workflow were well understood before making any changes. We did a thorough write-up of this knowledge to remove the existing single points of failure.
- We enacted a policy of "full output equivalence", meaning that code refactorings to the project were not allowed to change computational results. To ensure this, we used generative testing to generate millions of input-output samples, covering well the probability distribution of possible loan applications, and built them into the Continuous Integration pipeline. This warned us of any accidental logic changes.
- With this safety harness in place, we rewrote the legacy code incrementally and carefully over the course of multiple months, without causing a single unexpected bug.
- We helped the company fill their open developer positions by conducting technical interviews for their candidates. A few highly skilled programmers were hired based on our recommendations and we performed project-specific training to speed up onboarding. This included sharing our years of experience with fully remote work, helping the company to stay organized during the Corona virus lockdowns.
- With the legacy problem solved by the clean rewrite elaborated above, the newly built team could perform the architecture switch from VMs to AWS Lambda. Because the cleaned code was written in a Functional Programming (FP) style that Lambda fundamentally embraces (benefitting from statelessness and referential transparency), this migration went through without surprises.
- We incorporated common DevOps techniques FP Complete employs routinely to reduce the risk of downtime. Git-push-to-auto-deploy as our Continuous Delivery framework, linear version history for regression bisection, structured logging, and property-based integration testing were especially useful on this project.

New challenges for FP Complete

Through the course of the project, multiple obstacles had to be overcome, including:

- Unexpected departure of company employees, including our contact points.
- Additional responsibilities beyond the originally planned code implementation: Project management, product development, and engineering management.

- Navigating a wildly grown network of legacy databases and microservices that our project had to interface with.
- Protecting a large amount of Personally Identifiable Information (PII) during the advent of new EU privacy regulations (GDPR).

Conclusion

The clean rewrite and cloud architecture switch were executed as expected, despite the project starting on shaky grounds and suffering through staffing changes throughout.

During one year of development, the microservice that we worked on had no unplanned downtime, and the company's customers did not notice this fundamental rewrite going on in the background.

At the conclusion of the project, the customer was handed over not only a working service running modern, reliable code, but also a strong and competent new team ready to tackle further company projects.