BLUESENTRY

CASE STUDY: Database Modernization - Formpiper

FormPiper

THE CHALLENGE

FormPiper, a B2B Software-as-a-Service company, provides tools to make the completion of consumer finance applications quick and easy. End users enjoy a streamlined experience, able to apply to several financial institutions using one set of form inputs, while corporate customers are able to automate what was previously a very labor intensive process. At launch, FormPiper chose to build its product initially using MySQL as its backend data platform for ease of use and a minimum of configuration for their development staff. Once they partnered with Blue Sentry and started working with our Transformational Engineering teams, we identified many areas for further improvement of the platform.

THE SOLUTION

Among the improvements was an initiative to update and modernize the data platform for cost savings, increased performance, and enhanced reliability and scalability. Moving from MySQL to native AWS services seemed to be the most logical choice, given the way FormPiper's application was utilizing MySQL. Also, by utilizing NoSQL for a lot of their storage needs, a good portion of their data could be better served with increased performance and simplification of the relational schema where that was required. DocumentDB was selected and spun up and data migrated by the Blue Sentry team. Step one was tested and deemed successful and ready for the move to Production, but in discussions with Blue Sentry engineers, it was decided to take things one step further. Given FormPiper's use case and the focused need for the relational database, which is bound to set operating schedules, the decision was made to move the data that still required a relational database framework to PostgreSQL running on Aurora serverless. Blue Sentry once again took the reins and built the Aurora cluster; then, working with the FormPiper development team, migrated the necessary data to this new platform and tested it extensively.

THE BENEFIT

FormPiper had realized increased performance by utilizing the proper database models given its application needs. Data that lends itself to storage in a NoSQL system is now stored in DocumentDB, lessening the load on the relational database platform. At the same time, data that still requires the relational database framework now resides in a resilient and performant system that has reliability and scalability built into its DNA. The serverless nature of the platform also allows FormPiuper to automatically scale down relational resources when not in use, maximizing cost savings and with no performance or reliability downside.