An AWS Modernization and Migration to Cut TCO by 97%

CLIENT
Pitney Bowes

SOFTWARE
Postage Payment System

LANGUAGE PAIRING
HP NonStop Tandem COBOL To C# .NET Core on AWS

COMPLETION TIME
6 months

CHALLENGE

Move a Postal Payment Titan Forward

Since 1920 Pitney Bowes has dominated the field of US postal payment metering. It remains a core product in their innovative line. However, in 2021 their postage payment processing system was still in COBOL, a language developed over 60 years ago. This presented limitations that the company would need to overcome, such as:

- **The total cost of ownership** to maintain the systems was millions of dollars per year.
- **A limited, shrinking, and expensive talent pool** of people who still know how to manage and write COBOL.

HIGHLIGHTS

- **Increase Scalability and Agility**
- **Reduce TCO by 97%**
- **Modernize Outdated Technology**
- **Innovate Faster**

SOLUTION

From Monolithic to a Multi-tier, Distributed, and Cloud-Native Architecture

The modernization of Pitney Bowes' Postage Payment System application involved 390,000 lines of HP NonStop Tandem COBOL code and required the transformation of their Tandem Database to the more modern relational Microsoft SQL Server database. This meant moving from a file-based database on a monolithic system, where the application logic and the files are all located on a single piece of hardware, to a distributed environment in a new architecture on the AWS cloud.

Right away, TSRI was able to generate a fully hyperlinked *Application Blueprint*® of the entire legacy COBOL appllication, as well as a *Transformation Blueprint*®, that showed the current state, and the modernized C# .NET Core codebase would look like.

At the same time, TSRI and Pitney Bowes used their automation capabilities of *JANUS* Studio® to transform the code. *JANUS* Studio® identified externals and each of the frameworks in the legacy application, implemented the frameworks, tested, integrated, and deployed the new application and database in the AWS cloud environment.

Following the transformation and AWS migration, TSRI used the refactoring capabilities of *JANUS* Studio® to iteratively identify and remove bugs, code-quality issues, dead and unreachable code, and vulnerabilities that had been present in the original source. The refactoring process resulted in very high-quality code and reduced thousands of bugs to fewer than eight.

Even though Pitney Bowes provided updated legacy code baselines with roughly 40,000 additional lines of code to be transformed during the process, TSRI took in and regenerated the latest and greatest application—without delay or challenge in a matter of minutes. TSRI was able to do this because the TSRI process doesn't involve a line-by-line conversion or code freeze, so changes and additions in scope can be easily made at any point.

RESULT

99.996% Automation and 97% Drop in Total Cost of Ownership (TCO)

The end result was a modernized application in a modern language with modern, multi-tier architecture, microservices, and better performance on the AWS cloud—accomplished using 99.996% automation. TSRI achieved this while managing teams in three global time zones. And Checkmarx scans showed no un-dispositioned medium or above in the generated code.

Pitney Bowes now has a highly advanced cloud-native modern architecture and containerized deployment of their Postage Payment application transformed and refactored from COBOL to C# .NET Core as well as the Tandem database to a modern Microsoft SQL Server database on the AWS cloud. All of this resides on the AWS cloud, using the Well-Architected AWS model. The project also migrated multiple Tandem-specific interfaces to REST and functionally equivalent C# framework.

Finally, since the application was modernized and migrated to AWS, the annual TCO is reduced by nearly 97%.

WHY MODERNIZE WITH TSRI?

- **Migrate to the AWS cloud** with TSRI's automated transformation and refactoring solution.
- **Preserve business logic** through a model-based approach.
- **Reduce code freeze to little or none.** Automation allows baselines to be automatically changed at any time.
- **Improve code quality and maintainability** while reducing technical debt and security flaws inherent in legacy code through pattern-based refactoring.
- **Save time and money,** reducing years-long timelines to months or weeks with near-100% automation.