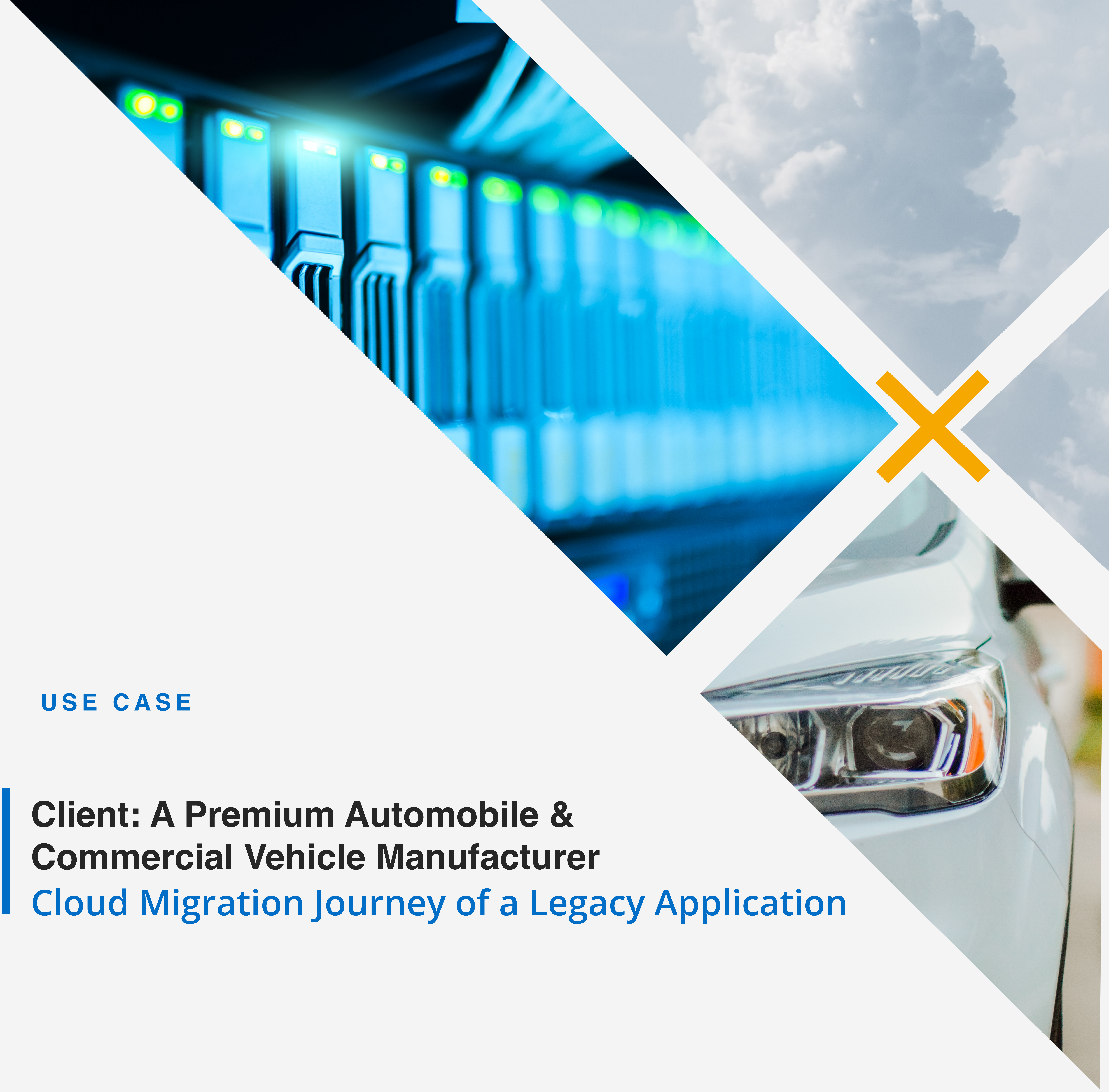

USE CASE

# Client: A Premium Automobile & Commercial Vehicle Manufacturer
# Cloud Migration Journey of a Legacy Application

adesso business. people. technology.

# Cloud Migration

## Use Case

### Project Goals

- Improving the flexibility and agility of delivery
- On-demand and fast paced cloud service
- Utilizing DevOps solutions
- High level automation
- Improving delivery quality & security in concepts

### Point of Initiation

Migration of the data maintenance application to cloud platform.

### Company Profile

Our customer is one of the biggest premium car producers and the world's biggest commercial vehicle manufacturer.
The company runs production facilities in many countries all over the world.

## Overview

## Cloud Migration Journey of a Legacy Application

### Goals

Improve the flexibility and maintainability of the application without compromising the security, and reduce the total cost of ownership. Integration with the IT tech stack was another issue to be addressed.

Operating on cloud unfolds outstanding benefits for organizations.

>Decreasing governance related costs

>Introducing on demand cloud services

>Improving time to market

>Improving delivery quality and security

### Challenges

Cloud migration introduces its own challenges. Dev. team was responsible from migrating an application that runs on Websphere, uses DB2 as database, and implemented with Java EE platform technology to a modern environment.

The team had to take the following into account

> Deciding on the workloads to be moved and the order of migration.

> Maintaining the reliability and performance of business-critical services.

> Implementing appropriate levels of security controls and regulatory compliance.

### Solution

With a clearly defined, well-researched and coherent strategy, the development team has initiated the migration project.

Accelerating cloud migration

We have decided to switch to a Spring boot application.

Since DB2 database was not an option anymore they have utilized PostgreSQL as the new database.

## Results

The product was successfully migrated from legacy environment to Cloud Foundry platform.

Application maintenance, development and integration are now much easier and faster.

New features and bug fixes can be deployed to Integration and Production environments very quickly, sometimes on the same day after the request is made. Before it took at least a couple of weeks.

The deployment frequency of the application was improved %67 (from 3 weeks to 1 week in average) in 1 year.

The cost of running the application, dropped by a large margin, since new environment and the services it provides is much cheaper.

# Implementation Steps

## EJB to Spring Boot

**1- Changing the annotations EJB**

EJB has own its individual bean definition annotations for REST scenarios, many REST annotations changed, such as: "@Stateless" bean definition annotation changed as "@Component" or "@Service" or "@Repository"

**2- Interceptor Changes**

The legacy app had its own interceptor annotation as “@InterceptorIcon” for logging or executing some logic after or before related method calls. All annotations changed with custom “@InterceptorIconAspect”. The new annotation used with the “@Aspect” annotation definition on the class level.

**3- Changing the Transaction Management**

The legacy application was run on WebSphere and the EJB application uses Container Managed Transaction. Spring Boot has a separate application server, the embedded Tomcat. Even using embedded or divided tomcat, developers can manage processes with code-level annotations. The "@Transactional" annotation was defined. This definition was able to merge existing transactions or create a new transaction if there is none.

**4- Changing the Authentication & Authorization Services**

The authentication mechanism was redefined using Spring Boot configurations. For restful request authentication JWT has been used. On the login mechanism, the customer SSO login service was implemented.

**5- Updating Spring Batch**

All of batch job configurations and steps were altered to work with new version.

## OpenJPA to Hibernate

The general usage of OpenJPA and Hibernate has the same annotations because of these two ORM frameworks using the “JAVAX” ORM library. The “Hikari” database connection pool was used.

## DB2 to PostgreSQL

All the native SQLs checked and changed to the corresponding PostgreSQL keywords. There are some issues with data type differences between DB2 and PostgreSQL. For example, DB2 uses BLOB or CLOB for byte typed variables but they were converted to Byte. Data migration from DB2 to PostgreSQL. With a tool we’ve developed which takes data from DB2 and checks data types and if datatype using with a different keyword in PostgreSQL, the tool changes that keyword and creates an insert script respectively.

## CI & CD

CI, CD pipelines and auto-deployment functions are implemented using Jenkins. Cucumber and Selenium were used as Test Automation tools. SonarQube also integrated into CI/CD pipeline to continuously check the code quality at expected levels.

adesso

# THANK YOU!

CONTACT

www.adesso.com.tr • info@adesso.com.tr
T +90 212 346 20 02
Maslak Mah. Ahi Evran Cad. Olive Plaza No: 11 34398
Sarıyer/IST