
VIRTUSLAB

Success story

# Spark 3 migration for the top global retailer

Our client aimed to optimize their data analytics strategy. Doing so would reduce the manhours and other operational costs needed to complete their daily tasks.

VirtusLab prepared and executed a migration of the existing jobs to Spark 3 in a way that allowed our client's platform to operate without disruption. It also enabled our client to update the platform to a more efficient version with a better Developer Experience and maintain its compatibility with legacy code.

## The Challenge

In its current state, our client's Big Data analytics platform relied on legacy code which made it inefficient and difficult to scale and maintain.
The platform was based on Spark 2 and Scala 2.11. Updating the platform required a newer version of Spark and Scala, which prompted our client to update them both. This meant conducting a full-scale migration to Spark 3.

The migration process carried a risk of service disruptions. Any loss of the platform's availability carried a risk of significant revenue drops and reduced customer satisfaction. As a global retailer, our client was determined to avoid that to maintain their strong reputation.

Our client needed additional manpower and expertise to navigate this high-stakes migration as their engineering teams were occupied with their ongoing projects. They also lacked experience with systems transition.

Encouraged by our existing partnership, our client turned to VirtusLab to avoid the common pitfalls of large migration projects.

## The solution

VirtusLab helped our client in executing their high-level migration plan. During the entire process, our engineers advised the retailer on some aspects of the overall migration and were involved in planning, tracking, and implementing 3 large projects within the platform:

- Unifying legacy workflows.
- Offloading from legacy platform components.
- Migrations of Spark 3 and Scala 3.

Additionally, our engineers helped introduce a cross-compilation process to separate the upgrades of hundreds of ETL tasks from the platform's component upgrades.

### Unifying legacy workflows

Our client created jobs within their platform using multiple technologies and workflows. Their unification and refactoring into a common framework became an important part of the platform’s update plan.

The migration plan had a few assumptions on its own. For every piece of infrastructure, it allowed for the automatization of the migration, combined with the adoption of repeatable processes across different versions of the platform. It also added the ability to recover data for various error scenarios.

During the migration, our engineers also introduced additional improvements to the platform, such as optimizing the storage of raw data.

### Offloading from legacy platform components

The client’s platform was partially based on technologies that were no longer supported. It forced our engineers to rewrite some of the legacy jobs, as they were unsuitable for migration.

## Migrating Spark and Scala to higher versions

Before switching jobs to Spark 3.0.2, tour engineers made the necessary preparations to make the deployment as efficient as possible.

- Our engineers planned and performed a large refactoring and offloading project from one of the legacy platform components. They replaced it with a basic version of the medallion architecture approach; this was important because the component was no longer supported in the new versions of the tools.
- They also refactored and upgraded the code to depend only on libraries available for both Spark/Scala versions.
- As part of the rolling upgrade, our engineers backed up all the data related to the given module. In case of any unexpected events during the migration, the data could be easily recovered.
- They also introduced a cross-compilation to allow new Spark version usage, which required a different Scala version than previously used.
- Before deploying production, our engineers deployed workflows to test and validate them on a staging environment.

Once all the prerequisites were met, our engineers could deploy the new version of the platform. They executed the migration in a way that simplifies future migrations and potential upgrades for the client.

## Migration and deployment

Our engineers divided the migration of the existing jobs into smaller modules, which allowed for a shorter feedback loop. It also gave our engineers greater control over the process and made it more transparent for both users and developers working daily on the platform.

Additionally, the team provided proof that the new version was producing accurate results, ensuring no regression would take place during the migration. They achieved it through a combination of local testing and parallel production runs, with comparative analysis of results as needed.

Deployment was carried out in modules, one by one. Our engineers started with simpler jobs to see if the new version of the application operated properly without the risk of the most vital jobs crashing.

The migration of the platform itself was a matter of "flipping the switch" on the CI/CD pipeline and deploying. In the case of more critical jobs, our engineers conducted additional tests post-deployment or ran old and new versions simultaneously to ensure that both of them produced the same results.

## The results

VirtusLab migrated all the jobs to Spark 3.0.2 successfully. The platform stayed operational throughout the process and maintained its high availability. The migration and update of the platform yielded a range of benefits, both on a local and global scale.

**Locally:**

- Automation reduced the time required to add new data ingestions from weeks to mere days.
- For some workflows, the data size was notably decreased as a result of optimization efforts that were impossible before migration. For instance, a single table that previously was written in legacy code occupied more than 2 TB of space. Post-migration, the same table requires only 100 GB.
- The developer experience of our client's teams has improved due to more user-friendly and efficient environments.

**Globally:**

- Teams have migrated to more efficient frameworks, resulting in enhanced efficiency and quality of their work.

Our engineers also provided proof that the new version is producing the correct results (so no regression upon migration happened) by a combination of local and staging tests as well as parallel production runs with comparing results whenever necessary.

Additionally, the adoption of cross-compilation added backward compatibility, meaning that the system works with its legacy version. It also made the platform's code easier to work with and more modularized.

## The tech stack

| | |
|---|---|
| **Build process execution:** | Gradle Kotlin (DSL) |
| **Distributed computing framework:** | Apache Spark 2.3.0 / Apache Spark 3.0.2 |
| **CI/CD process:** | Jenkins / Ansible / custom Bash scripts |

# About VirtusLab

At VirtusLab, we aim to lead in software technology, working consistently to enhance efficiency. Our profound commitment to research and development and a dedicated focus on emerging trends and inspirations fuels an innovative culture. This ethos precisely guides advancing our cutting-edge solutions, inviting collaboration to expand the boundaries of software technology collectively. We welcome you to be a part of this transformative journey.

**Let’s connect**

# Contact Details


**info@virtuslab.com**

POLAND

**Kraków Headquarters**

Virtus Lab Sp. z o.o.
ul. Szlak 49
31-153 Kraków

GERMANY

**Berlin Office**

**+49 30 52014256**

VirtusLab GmbH
Potsdamer Platz 10
10785 Berlin

UNITED KINGDOM

**London Office**

**+44 (0)20 4577 1051**

Virtuslab Ltd.
40 Bank Street HQ3
London E14 5NR