EVENT-DRIVEN AND PROCESS-ORIENTED ARCHITECTURES

PROJECT PRESENTATION

MICHAŁ CISŁO & RAFFAEL ROT

ABOUT
THE PROJECT

Machines: The Machine microservice lets us manage wood-shaving machine easily. We can turn it on/off and control the start of the production line.

Warehouse: Responsible for managing the logistics and inventory within the manufacturing facility. It facilitates the confirmation of transport orders, quality checks, stock level updates, and the storage of goods.

Factory: The Factory microservice coordinates various tasks including initiating the production line, monitoring machine fill levels, publishing real-time inventory data, and orchestrating complex process.

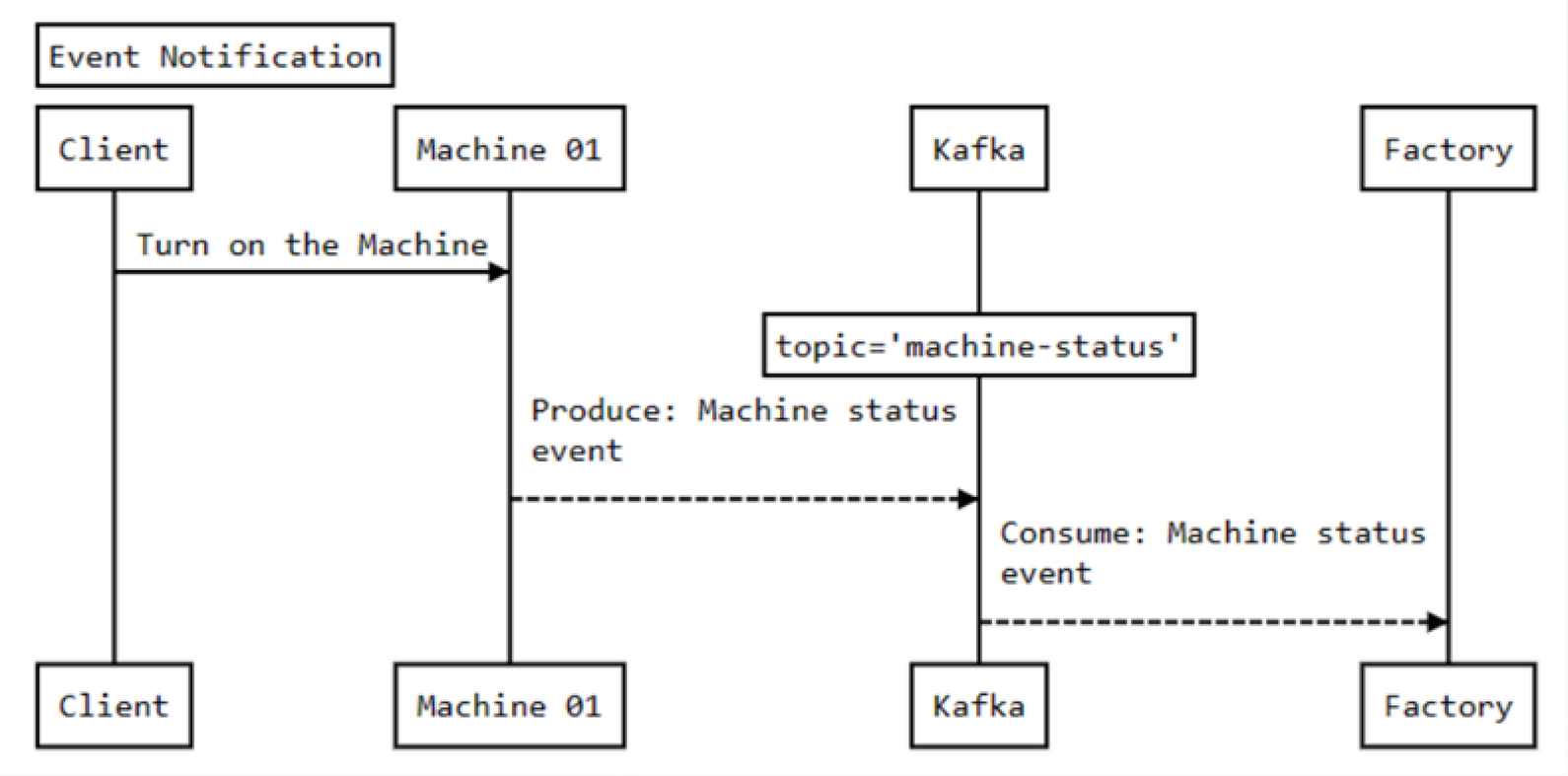
Logistics: Deals with transporting goods. It schedules transfers, picks up goods from the factory, and delivers them to the warehouse. This microservice plays a role in ensuring delivery of goods, optimizing supply chain efficiency.

EMPLOYED CONCEPTS

IN THE PROJECT

EVENT NOTIFICATION

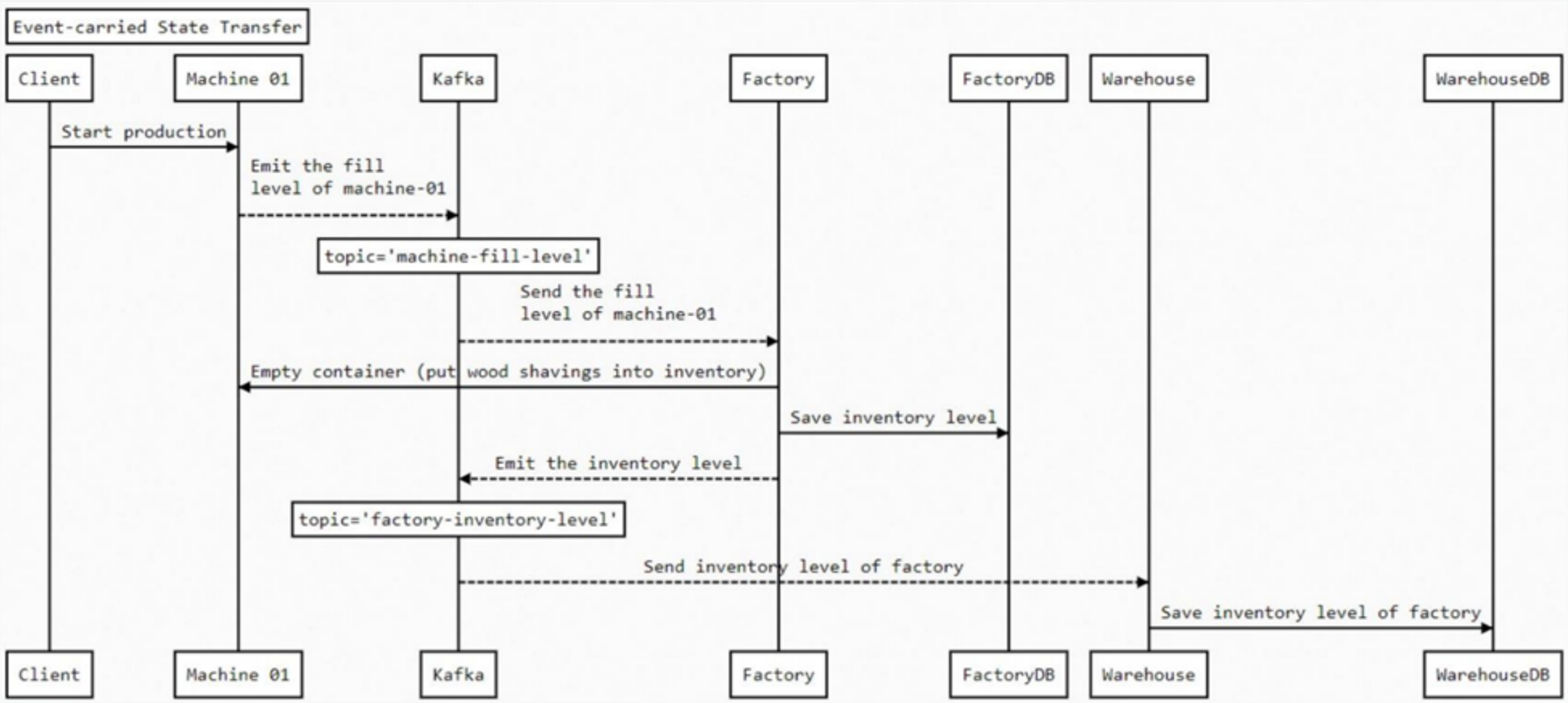
This event-driven approach enhances system responsiveness and agility, enabling real-time updates without necessitating continuous polling or manual intervention.



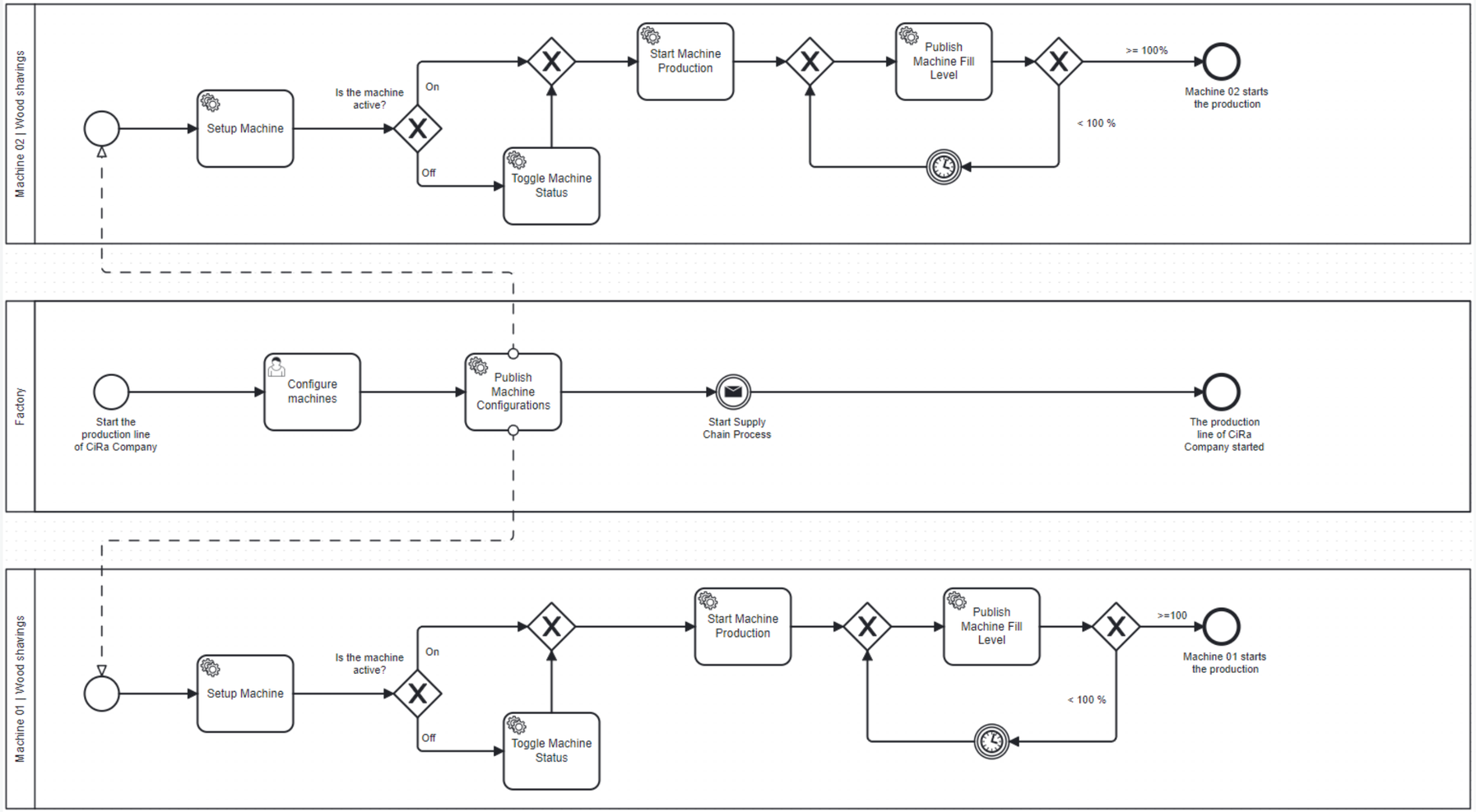
EVENT-CARRIED STATE TRANSFER

Trade-offs:

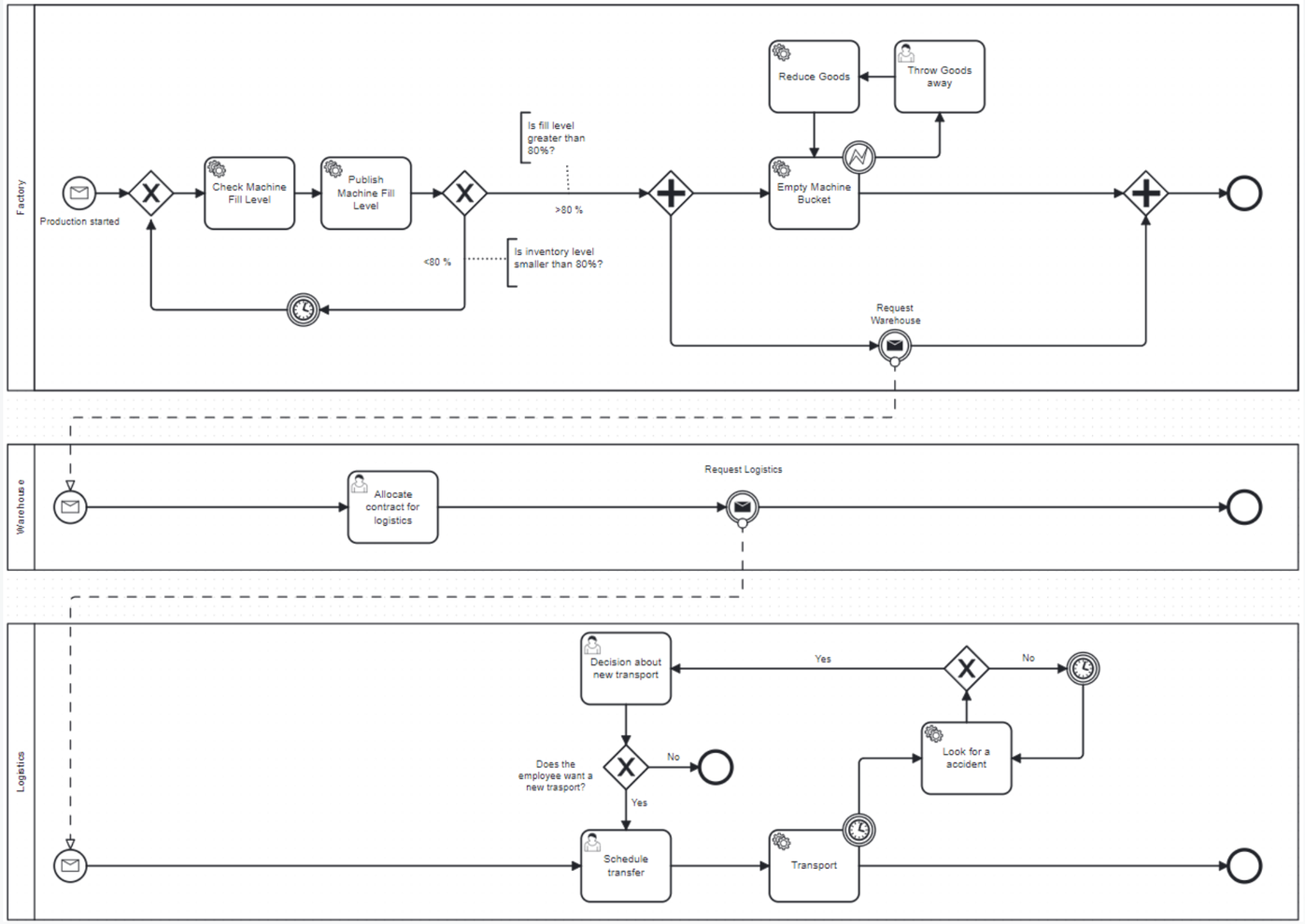
- Decoupling and reduced load on Factory
- Replicated data and eventual consistency



PARALLEL SAGA

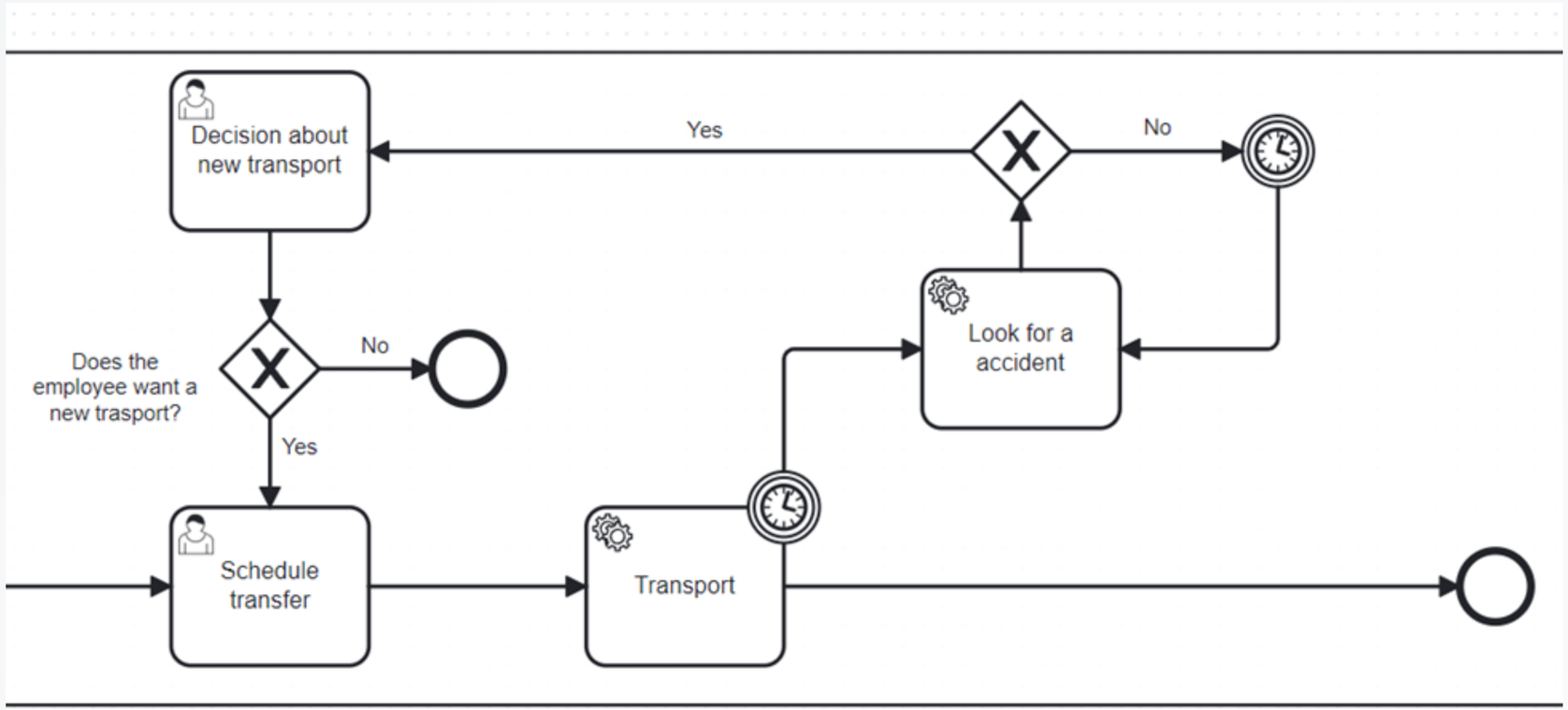


ANTHOLOGY SAGA



HUMAN INTERVENTION

In the Human Intervention pattern the system is designed to gracefully handle exceptional situations by involving human intervention when necessary.



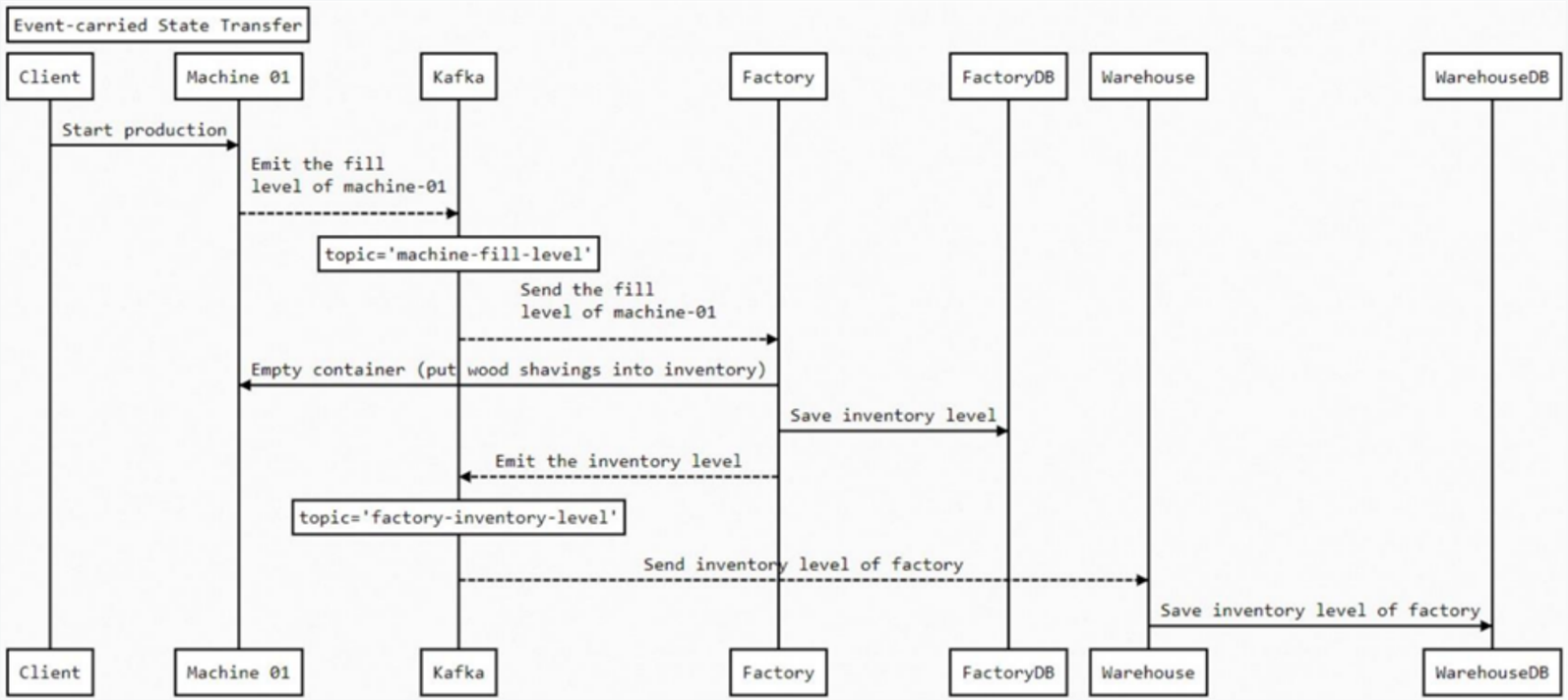
STATEFUL RETRY

In the Stateful Retry pattern within event-driven architecture, resilience is achieved by implementing a stateful mechanism for retrying failed service tasks. Specifically, each service task is retried up to three times upon encountering a failure. This approach aims to improve the robustness and fault tolerance of the system by allowing failed tasks to be automatically retried.

TRADE-OFFs AND ADRs

ASYNCHRONOUS COMMUNICATION - EVENTUAL CONSISTENCY

Asynchronous communication allows systems to operate without waiting for immediate responses, leading to higher throughput and reduced latency. However, this approach requires handling eventual consistency, meaning that data might not be instantly synchronized across all parts of the system.



PARALLEL SAGA AND ANTHOLOGY SAGA

Parallel saga

Parallel saga enables concurrent execution of tasks, allowing systems to respond to user requests or events, thus enhancing the user experience.

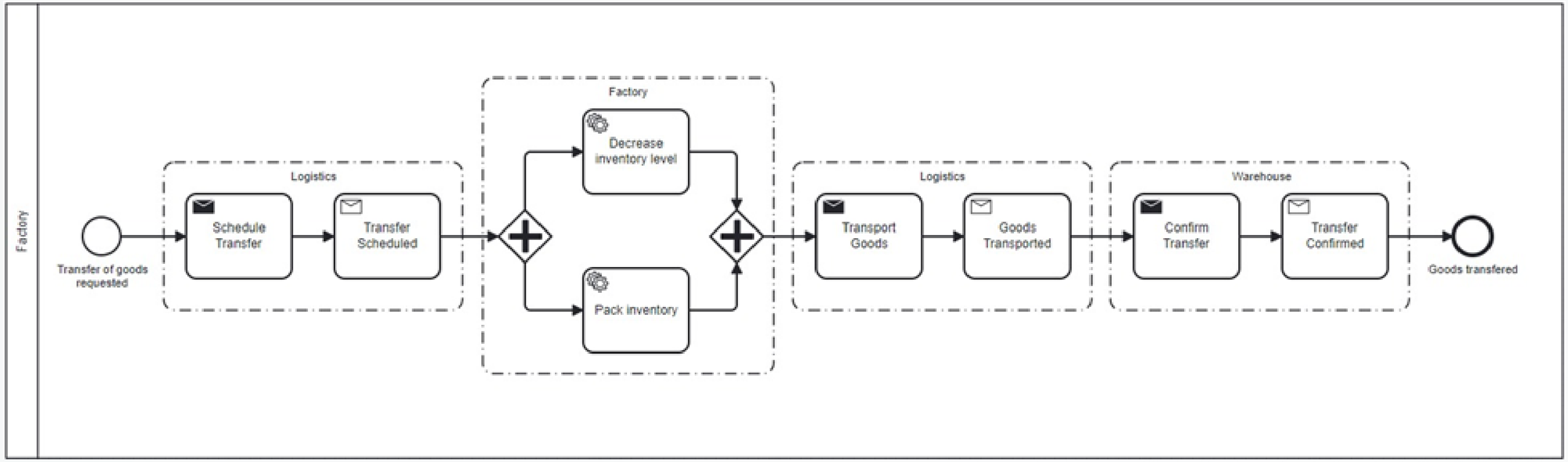
Anthology saga

Anthology saga facilitates horizontal scaling, ensuring the system can efficiently handle growing workloads or dynamic demands without compromising performance.

This combination of parallel and anthology sagas optimizes responsiveness and scalability, laying the foundation for robust and adaptable event-driven architectures.

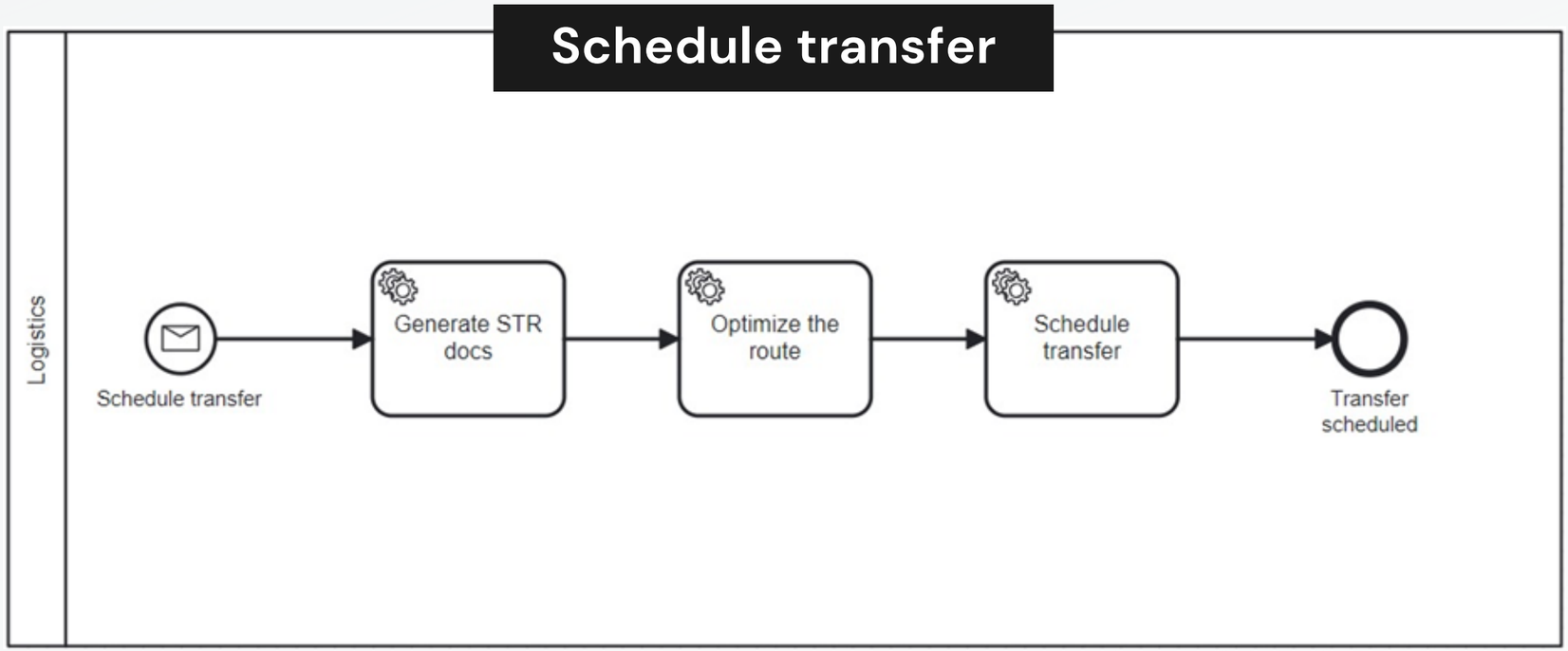
BALANCING ORCHESTRATION AND CHOREOGRAPHY

In our system, we faced an important trade-off between orchestration and choreography. To address the challenges posed by a complex process, we employed a combination of commands and events. By utilizing events for communication and commands for control, we've achieved a delicate balance between centralized coordination and decentralized autonomy. Our approach is based on an understanding of responsibilities, guiding us in determining whether to use commands or events.

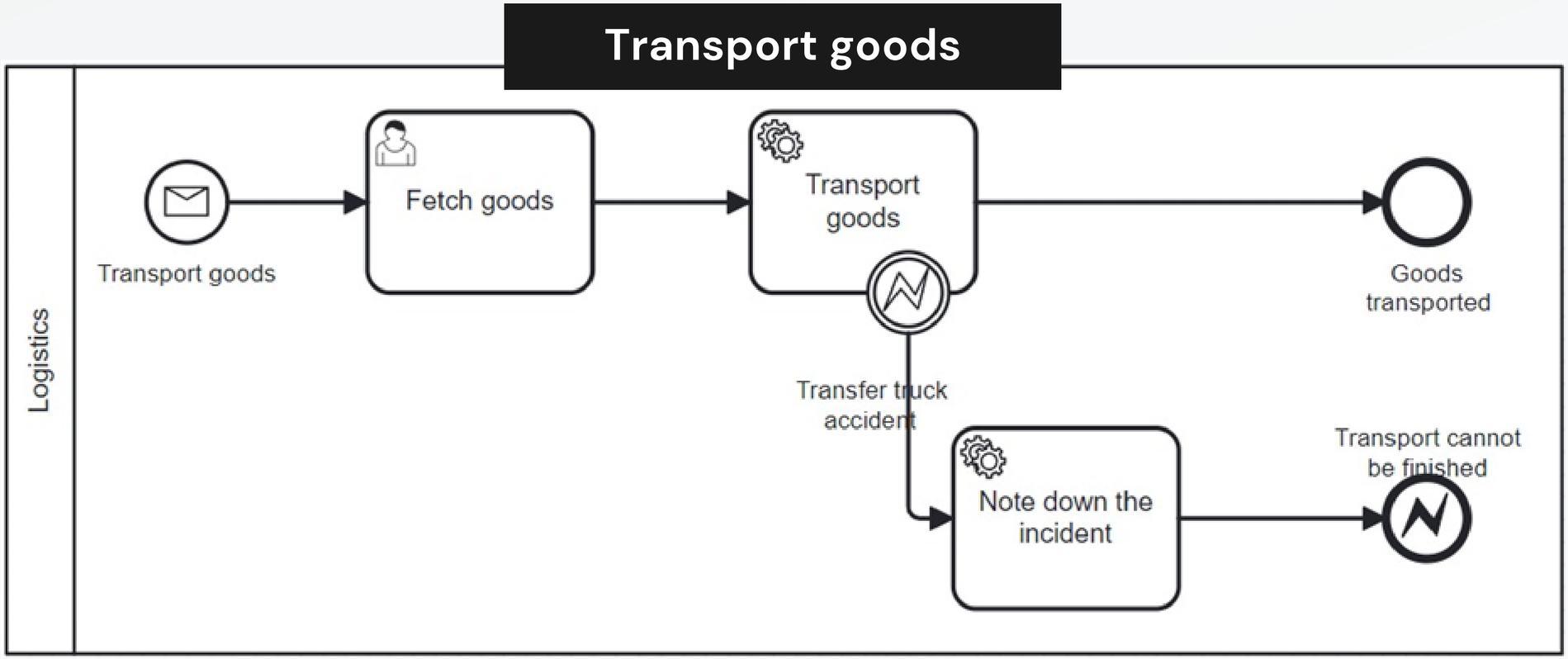


RESPECT BOUNDARIES AND AVOID PROCESS MONOLITH

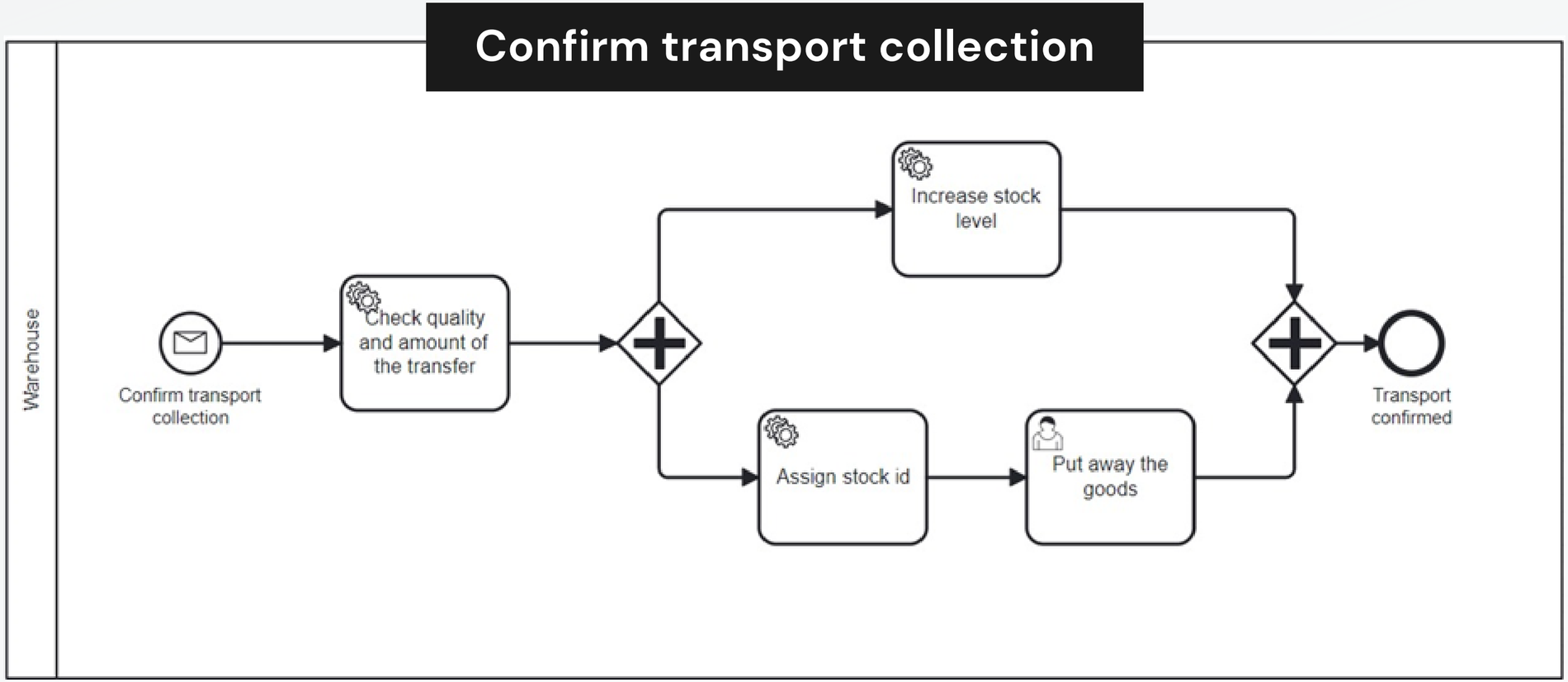
Avoiding process monoliths involves breaking down complex systems into smaller, manageable components with well-defined boundaries. This approach promotes modularity, allowing for independent development, deployment, and scaling different parts of the system.



RESPECT BOUNDARIES AND AVOID PROCESS MONOLITH



RESPECT BOUNDARIES AND AVOID PROCESS MONOLITH



ADR
CAMUNDA 7 OR CAMUNDA 8

6. Continue with Camunda 7 or Camunda 8

Date: 2024-04-05

Status

Accepted

Context

We must decide whether to continue with our current version, Camunda 7, or migrate to the newer version, Camunda 8. This decision requires careful consideration of the benefits, features, and alignment with our team goals and individual skills.

Decision

After evaluation, we have decided to migrate from Camunda 7 to Camunda 8. This decision is based on several factors outlined below:

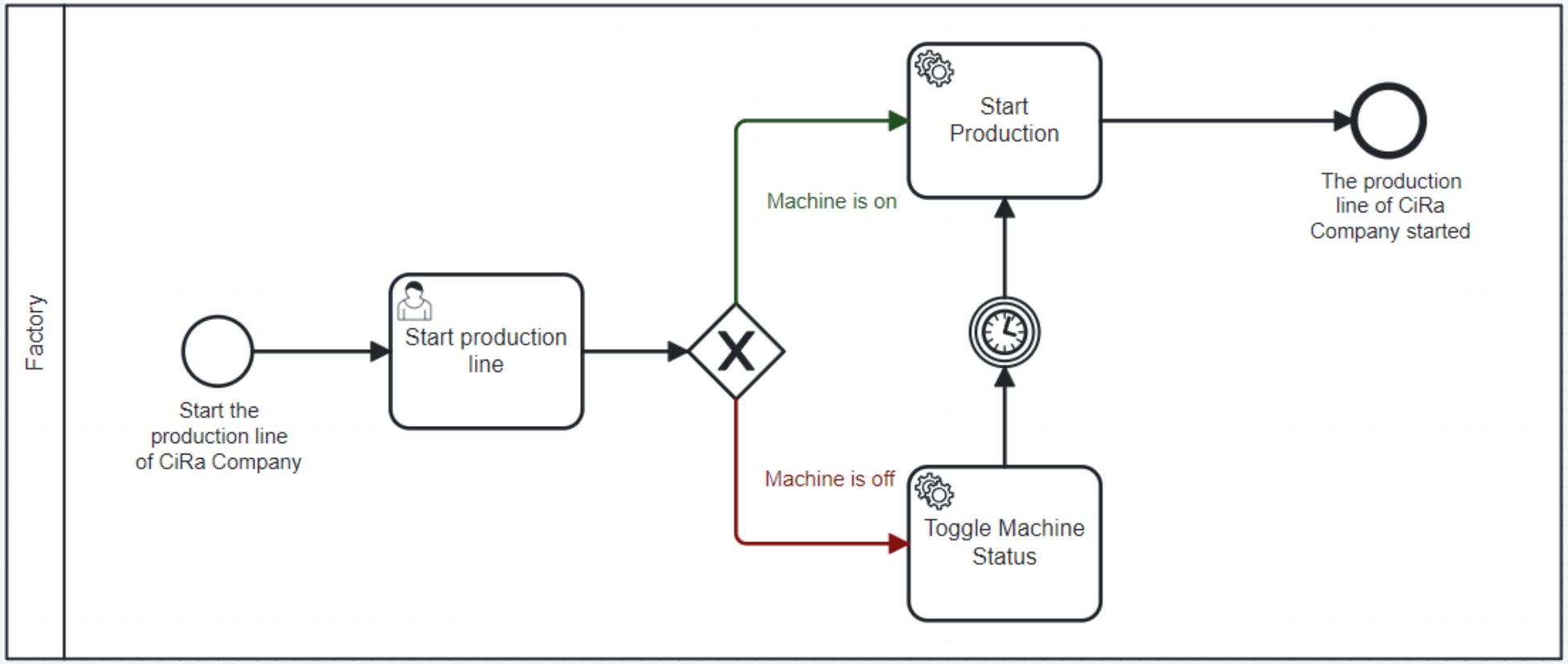
1. **Desire to experiment:** We had been using Camunda 7 since the beginning of the semester, so the moment we had the opportunity to use Camunda 8 and learn about the changes and differences between the versions, we decided to migrate.
2. **New skills:** Camunda 8 introduced new developments to its engine. We decided to switch to Camunda 8 in order to gain more skills in using the platform and the new developments in the code.
3. **Future-proof solution:** After considering the scalability and architectural differences between the Zeebe engine in Camunda 8 and the Camunda 7 engine, we've decided to use to Camunda 8. This decision is based on the scalability of Zeebe, does not rely on a relational database, eliminating its major bottleneck for scaling the engine.

Consequences

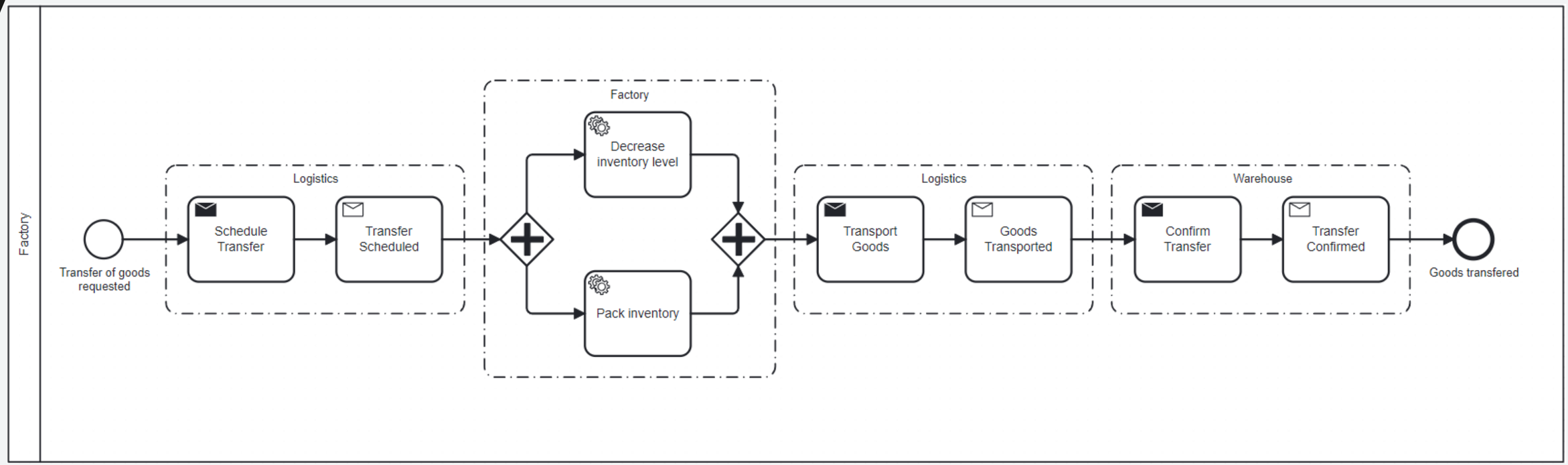
- Integration with existing systems and workflows may require adjustments to ensure smooth transition and minimal disruption.
- Adoption of new features and technologies may entail a learning curve.

PROCESSES DIAGRAMS

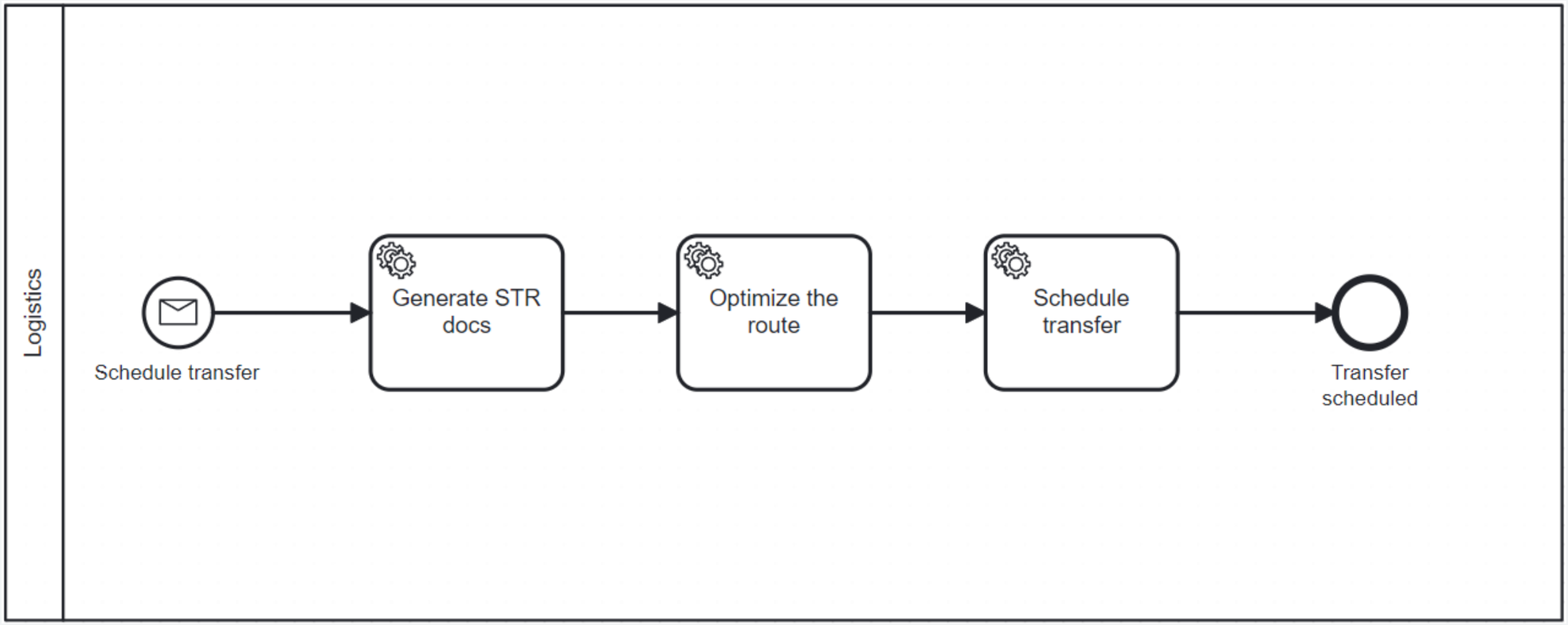
START PRODUCTION PROCESS



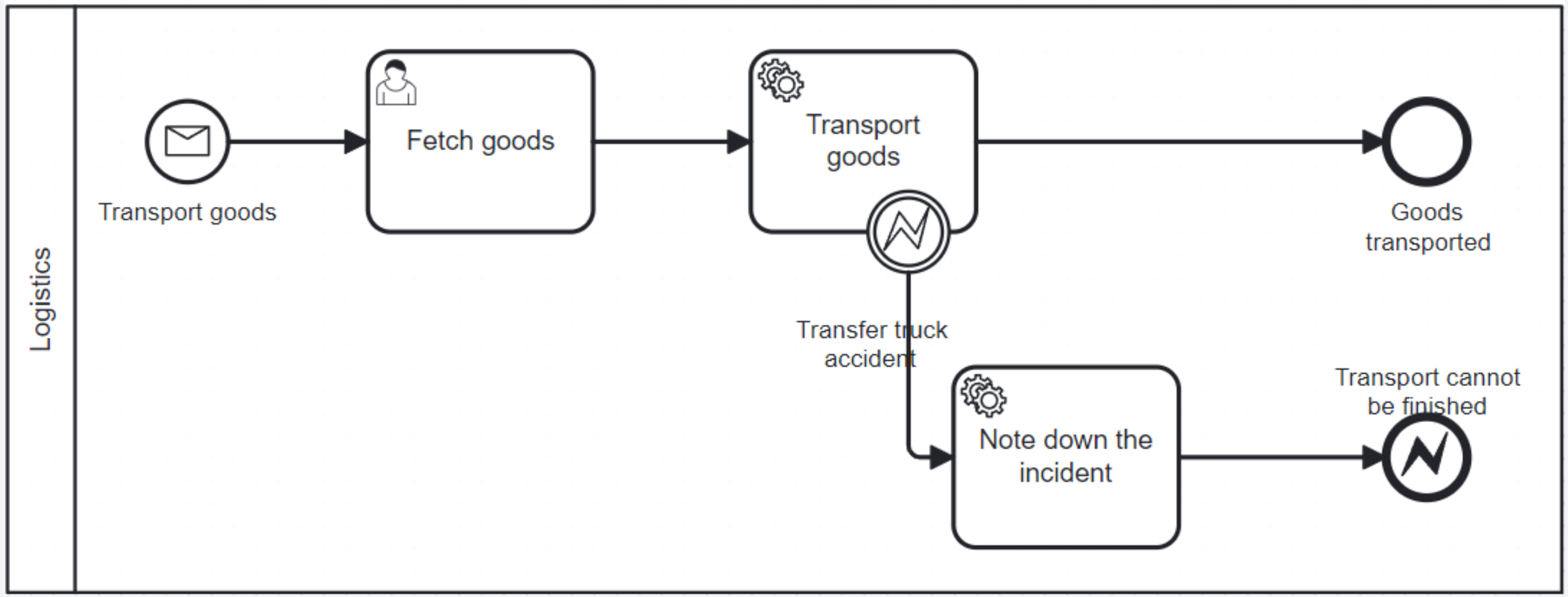
TRANSFER GOODS PROCESS



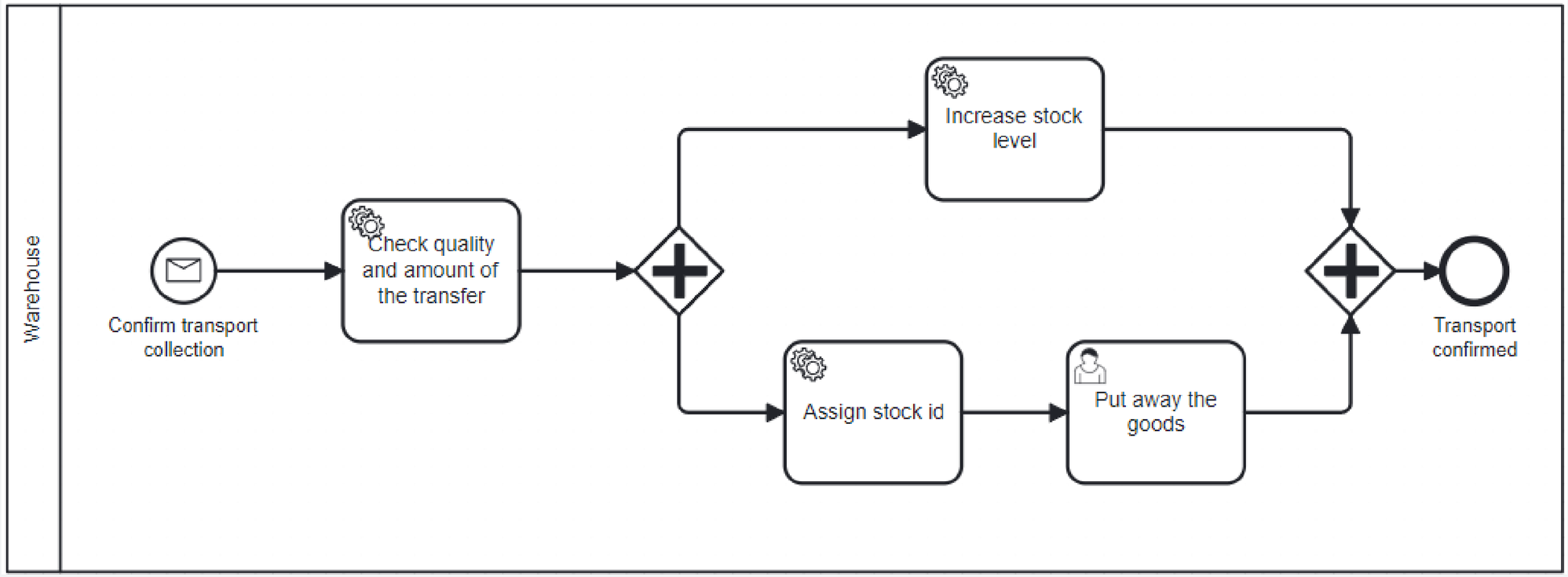
SCHEDULE TRANSFER PROCESS



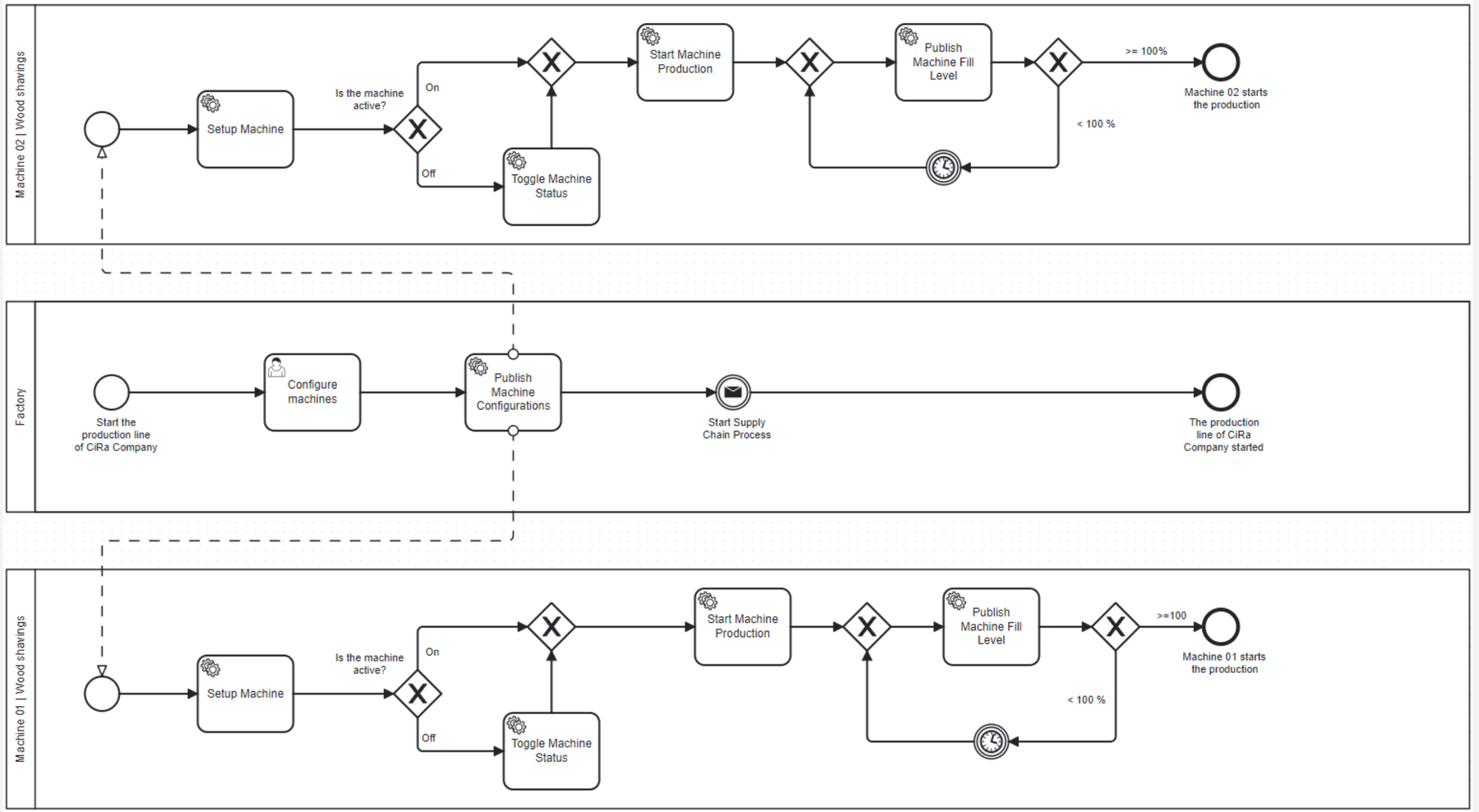
TRANSPORT GOODS PROCESS



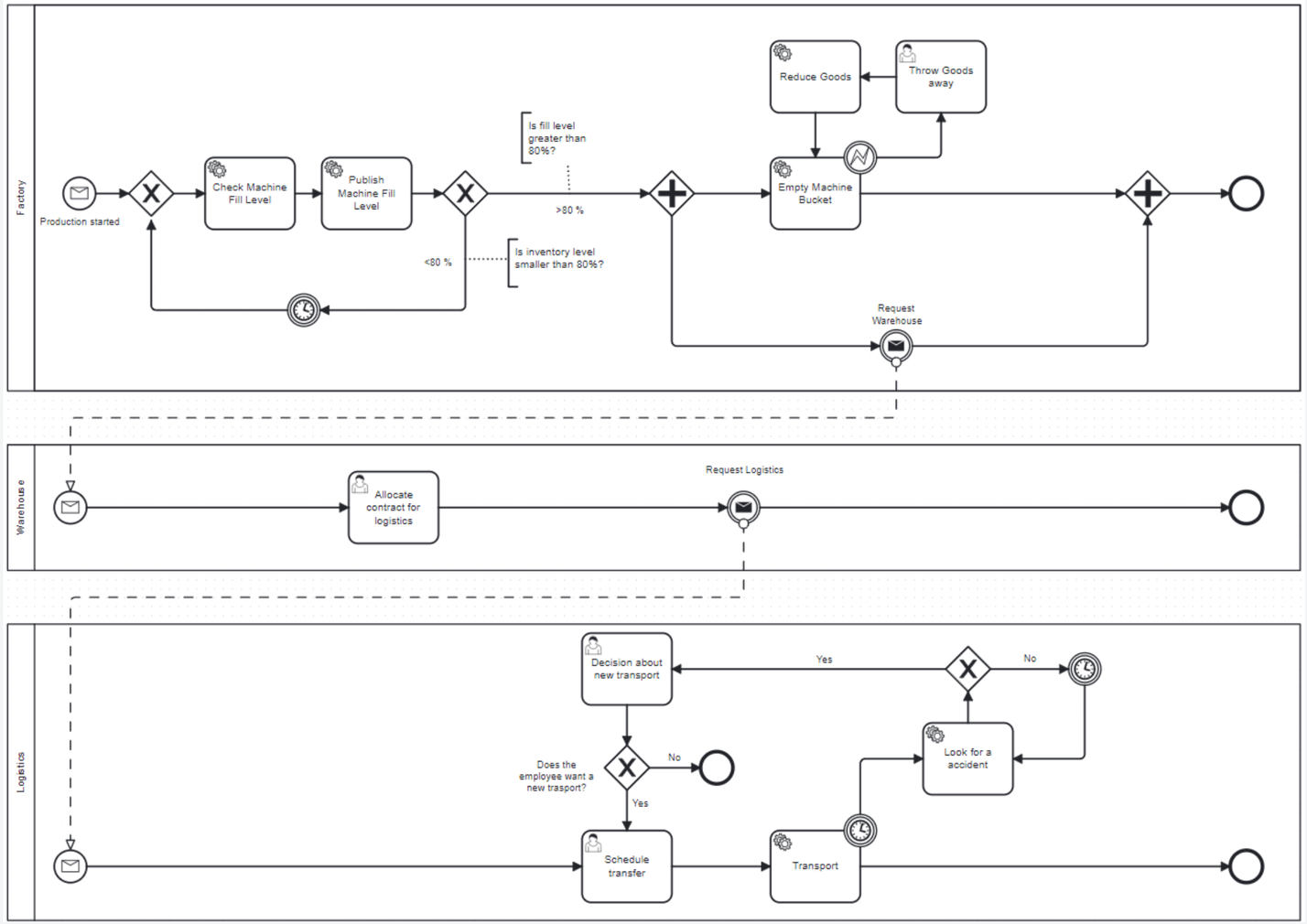
CONFIRM TRANSPORT PROCESS



START PRODUCTION LINE PROCESS



SUPPLY CHAIN PROCESS



DEMO

THANK YOU