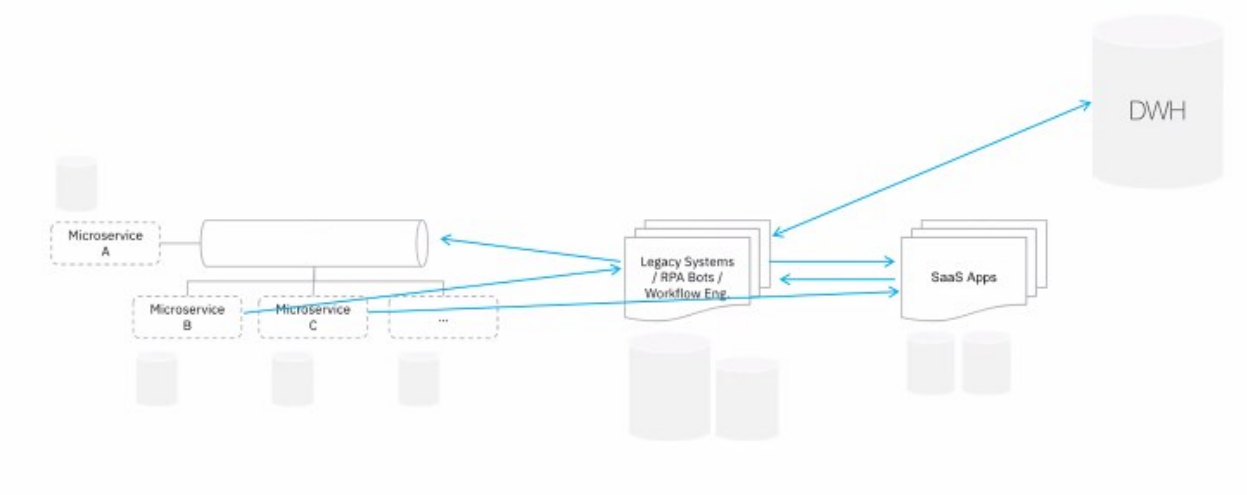
CAMUNDA COMMUNITY SUMMIT 2021

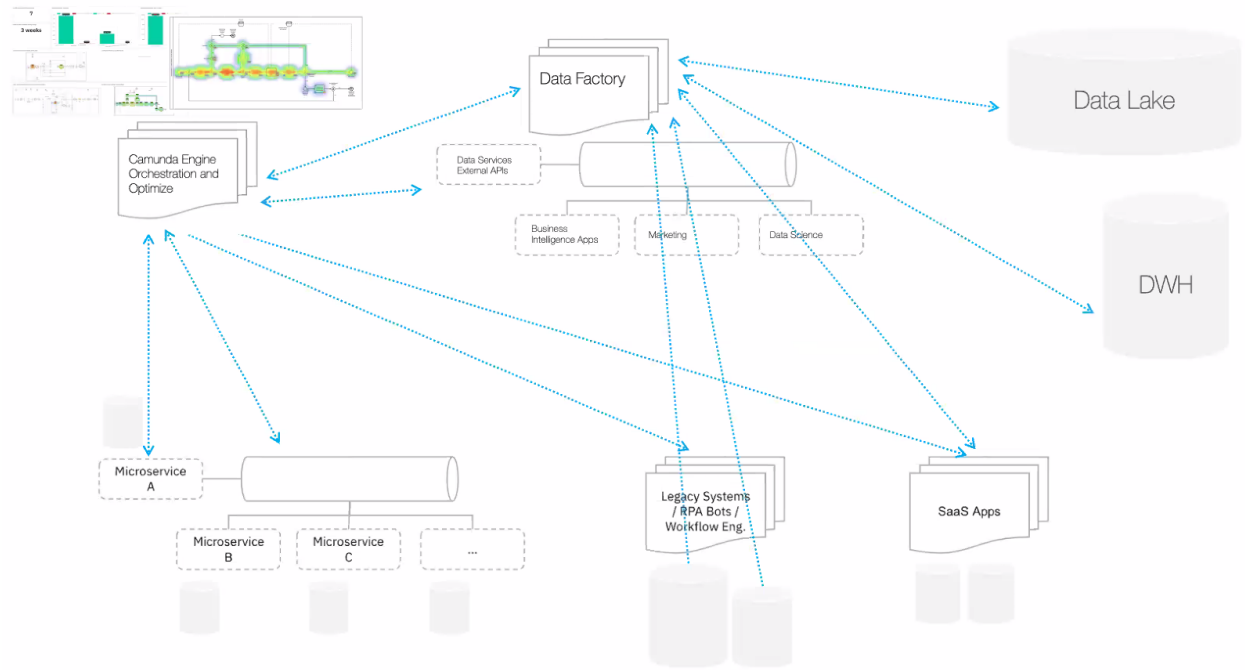
Session Notes:

<Session 3>
Topic: **Microservices Orchestration Using Camunda Platform**
Session Leader: Omid Tansaz
Link to Slides: https://drive.google.com/file/d/15YxEcwJjz4k4aeqyKA0hyCLNP-j7f-PN/view?usp=sharing

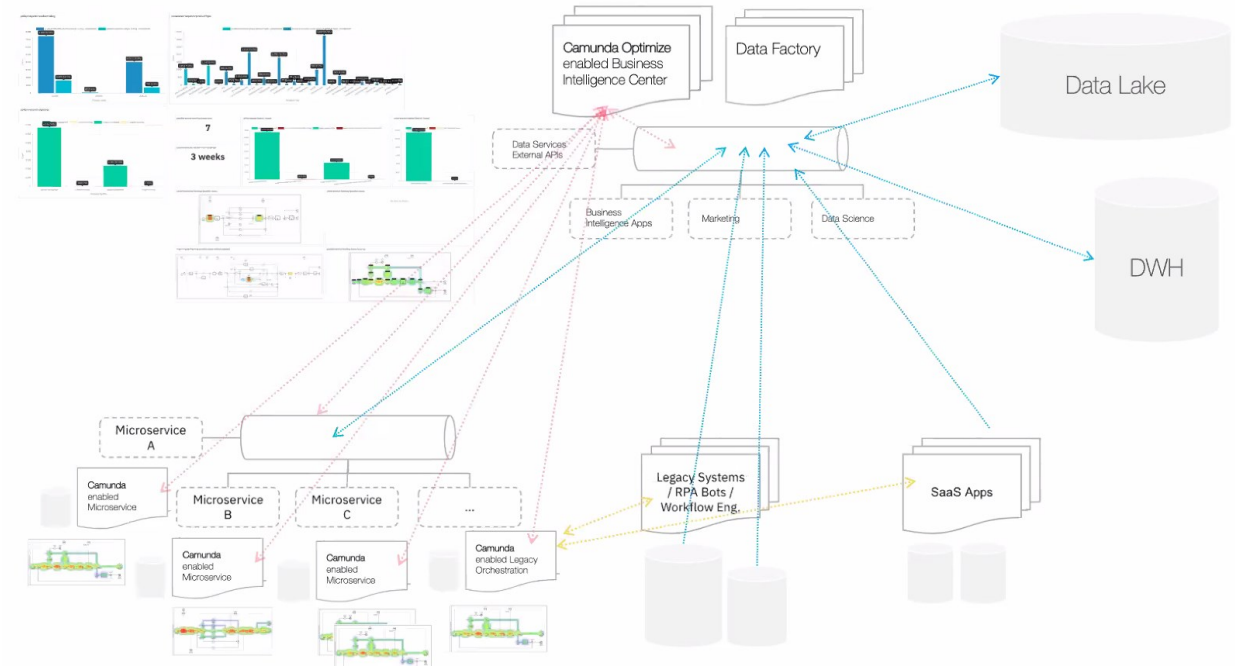
Interesting Notes:
Omid started with a relative “standard” microservices architecture - some participants could relate to



But this got unhandy and they moved towards a more orchestrated approach, some attendees reported that they are doing the same as kind of a natural journey



But this resulted in probably too monolithic orchestration capabilities. So they decomposed the orchestration



But this could lead to losing end-to-end visibility - but Optimize could help to regain the end-to-end visibility

Microservice-local engines seems to be favorable, but come with challenges:

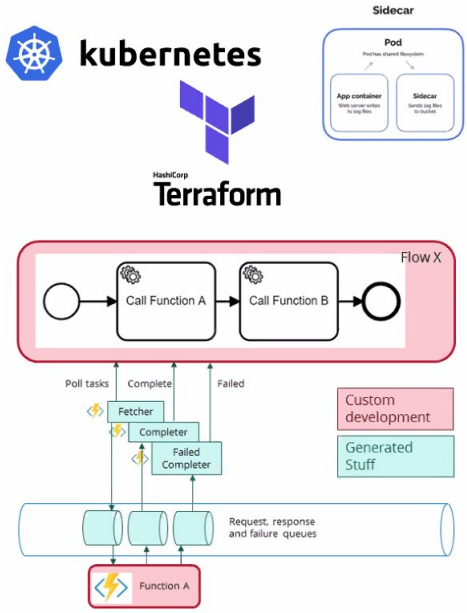
1. Microservice teams need to operate (and probably configure) their own engine
2. End-to-end visibility

Nexxbiz tried the following approach to dispatch External tasks to .NET core workers - where nobody has to write Java code:

Problems
- ✓ Reuse generic event workers that know nothing about Camunda
- ✓ Include orchestration in each microservice
- ✓ Do not depend on Java (allow cross platform components)

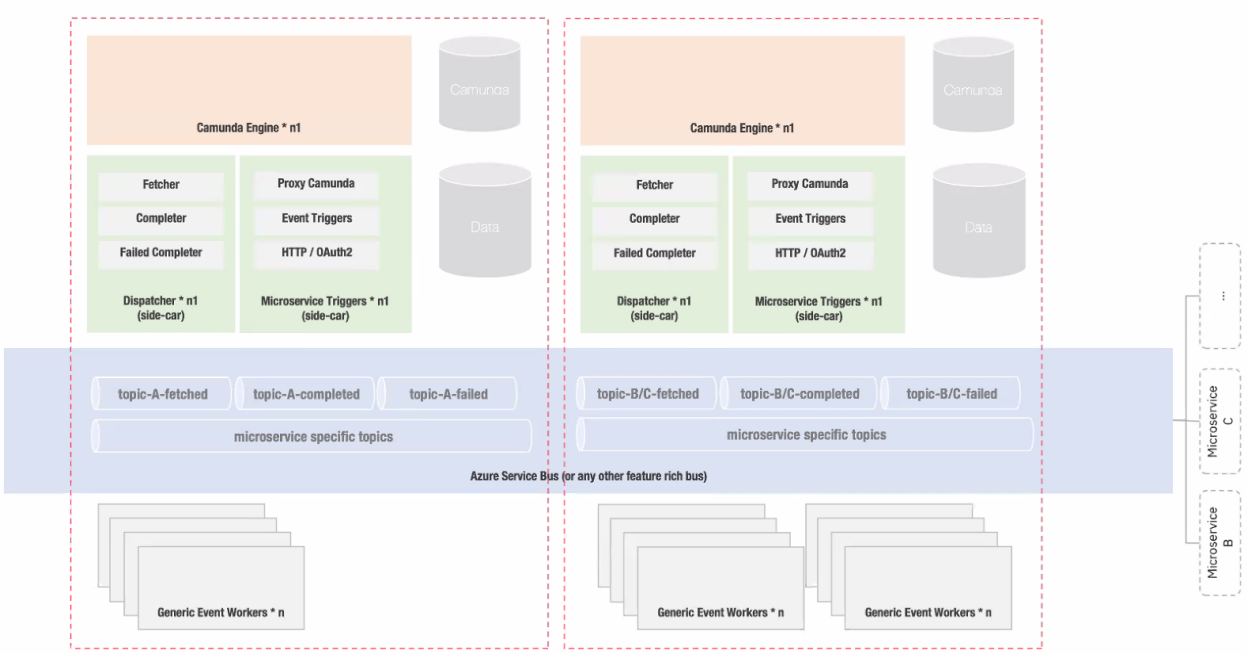
Gains
- ✓ Separating logic of dispatching Camunda workload from workers
- ✓ Single deployment of Camunda orchestration components Kubernetes (side-car)
- ✓ Allows Integration of enhanced security and RBAC without additional changes in Camunda (Java)
- ✓ Allows hyper scalability of workload
- ✓ Allows better control of workload (hence an intelligent service bus is used)

Trade-offs / Considerations
- ✓ Complex infrastructure setup
- ✓ Lock synchronization (message TTL and Camunda task TTL)
- ✓ Message duplication (consider intelligent service bus features to deduplicate)
- ✓ With External Service Architecture process execution is by default asynchronous



Write External Tasks into Azure service bus and subscribe to it by “normal” Azure functions

CAMUNDA COMMUNITY SUMMIT 2021



The big challenge was the infrastructure overhead (deploy a lot of resources for every task)
So they switched to a different approach and prepared some kind of task types and clarified the deployment via Kubernetes Operator

Problems

- ✓ Create dynamically scalable microservices with integrated orchestration
- ✓ Include orchestration in each microservice
- ✓ Enable business to control the logic of microservice
- ✓ Do not depend on Java (allow cross platform components)

Gains

- ✓ Allows scalability of each component separately within the bounded context of the Microservice
- ✓ Plugin system to define executable workflows and business logic by business
- ✓ Allows Integration of enhanced security and RBAC without additional changes in Camunda (Java)
- ✓ Less complex infrastructure is needed
- ✓ Camunda acts as internal bus and is responsible for managing workload (locking, etc.)
- ✓ More integrated deployment = Deployment using Kubernetes Operators

Trade-offs and considerations

- ✓ Complex infrastructure setup
- ✓ Tide coupling of dependencies (worker / orchestration)
- ✓ Additional logic needed to control the workload (Back-pressure and Back-off policies)
- ✓ Consider a plugin system to keep the microservice component generic and ensure dynamic and flexible work executors.
- ✓ With External Service Architecture process execution is by default asynchronous

KUBERNETES OPERATORS

KUDO

RabbitMQ

LOW-CODE PLUGINS

Business Rule Engines

Data Transformers

Code Runners

Http Connectors

Micro Workflows

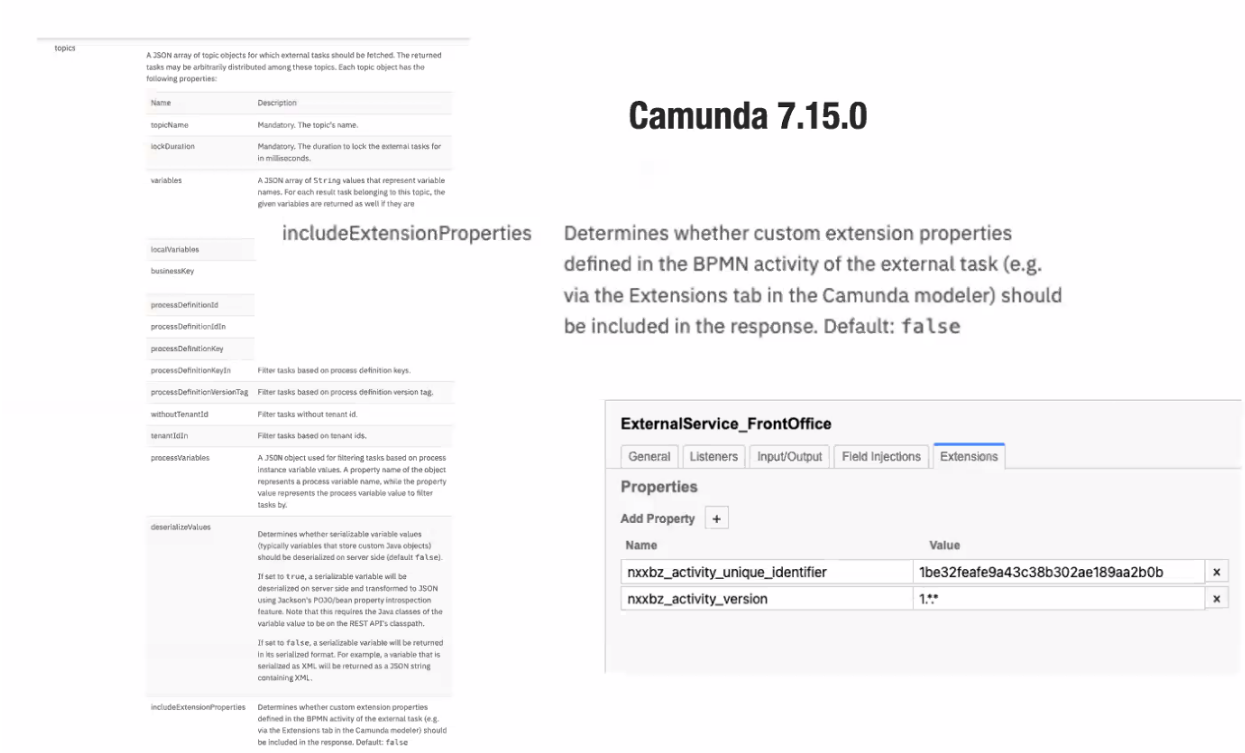
DMN Executors

CAMUNDA
COMMUNITY
SUMMIT
2021

Generic or specific workers?

Specific workers can be used and configured (especially from Camunda 7.15 on)

CAMUNDA
COMMUNITY
SUMMIT
2021



High-Level Takeaways:

1) What is the exact problem you are trying to solve, find the right pattern, consider trade-offs
2) Camunda External Service task the way to go if you don't want to use Java, check the documentation and some of the design patterns you can consider
3) Lousy coupled microservices require a level of visibility (an outpost) like Camunda Optimize, to ensure business stays in control of THEIR processes.