

CAMUNDA COMMUNITY SUMMIT 2021

Session Notes:

<Session 3>

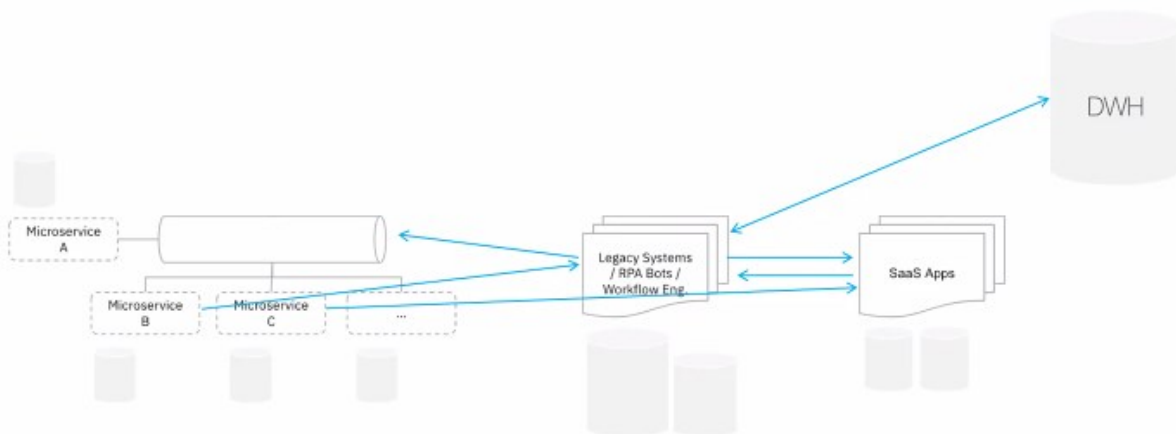
Topic: **Microservices Orchestration Using Camunda Platform**

Session Leader: Omid Tansaz

Link to Slides: <https://drive.google.com/file/d/15YxEcwJjz4k4aeqyKA0hyCLNP-j7f-PN/view?usp=sharing>

Interesting Notes:

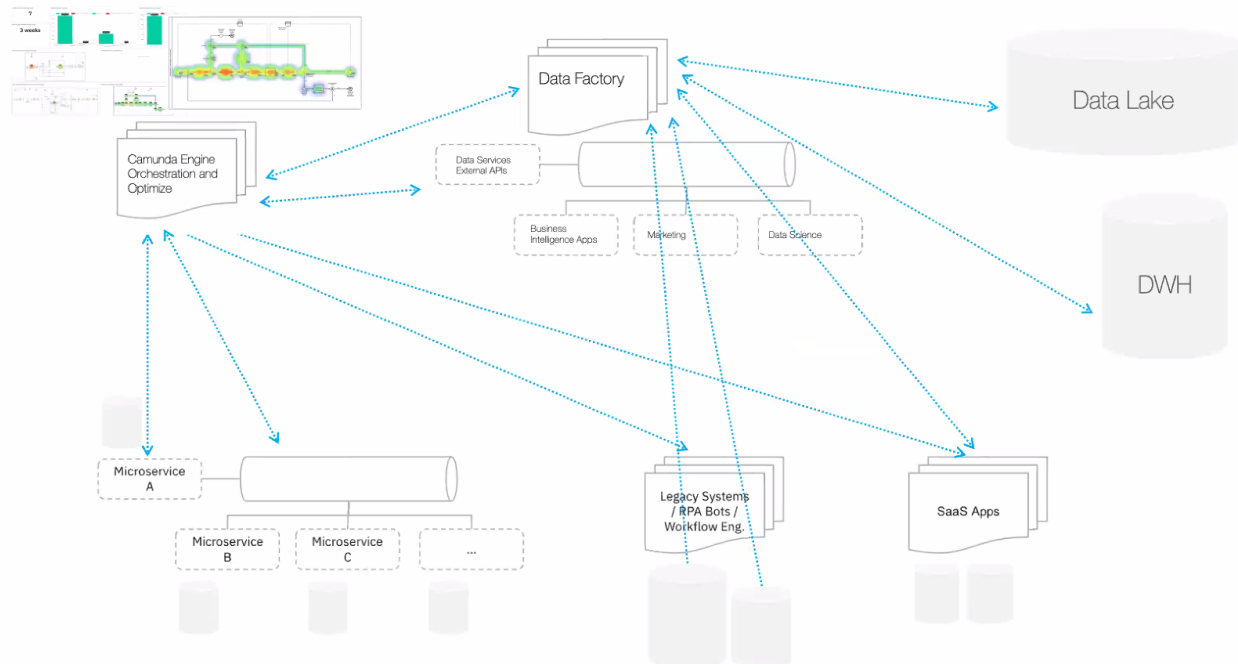
Omid started with a relative “standard” microservices architecture - some participants could relate to



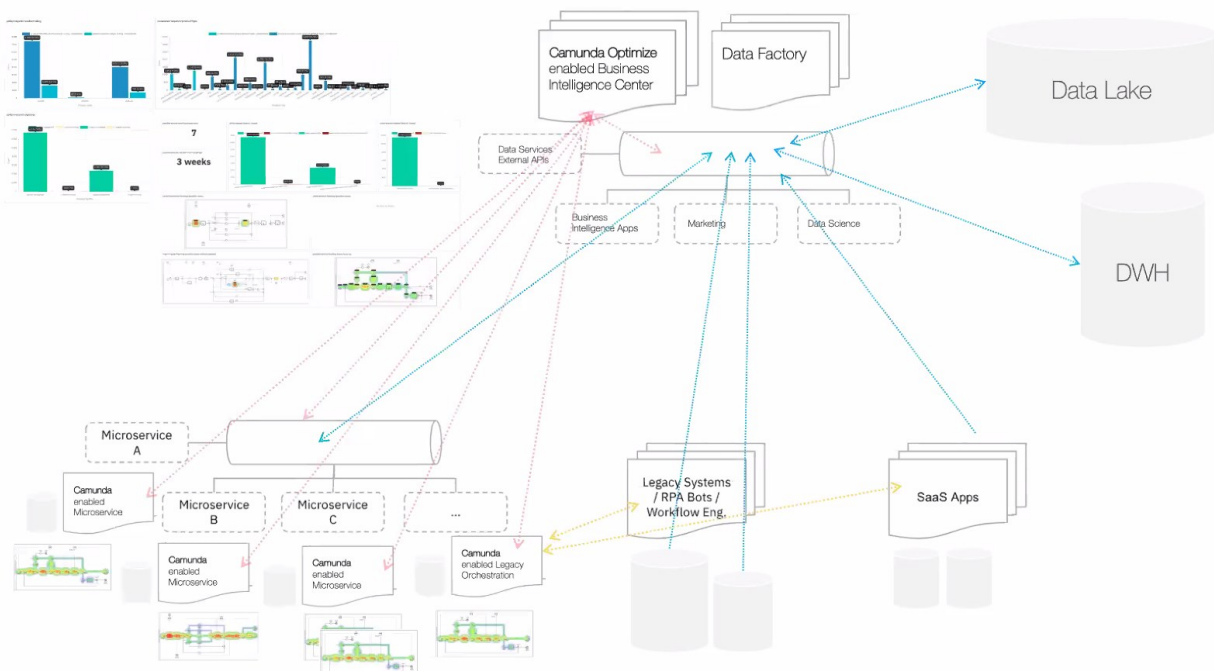
But this got unhandy and they moved towards a more orchestrated approach, some attendees reported that they are doing the same as kind of a natural journey

CAMUNDA COMMUNITY SUMMIT

2021



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



CAMUNDA COMMUNITY SUMMIT 2021

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

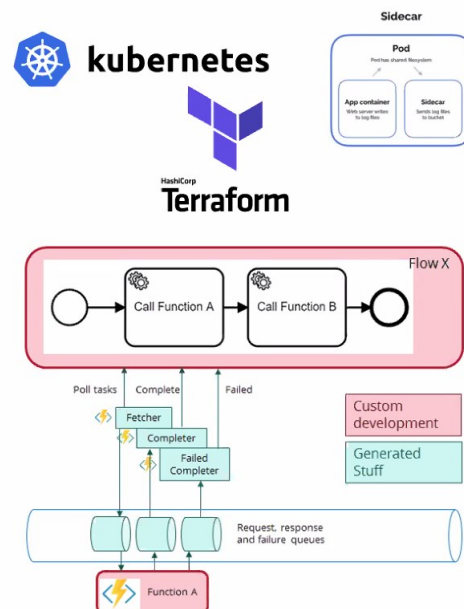
- ✓ Reuse generic event workers that know nothing about Camunda
- ✓ Include orchestration in each microservice
- ✓ Do not depend on Java (allow cross platform components)

Gains

- ✓ Separating logic of dispatching Camunda workload from workers
- ✓ Single deployment of Camunda orchestration components Kubernetes (sidecar)
- ✓ 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

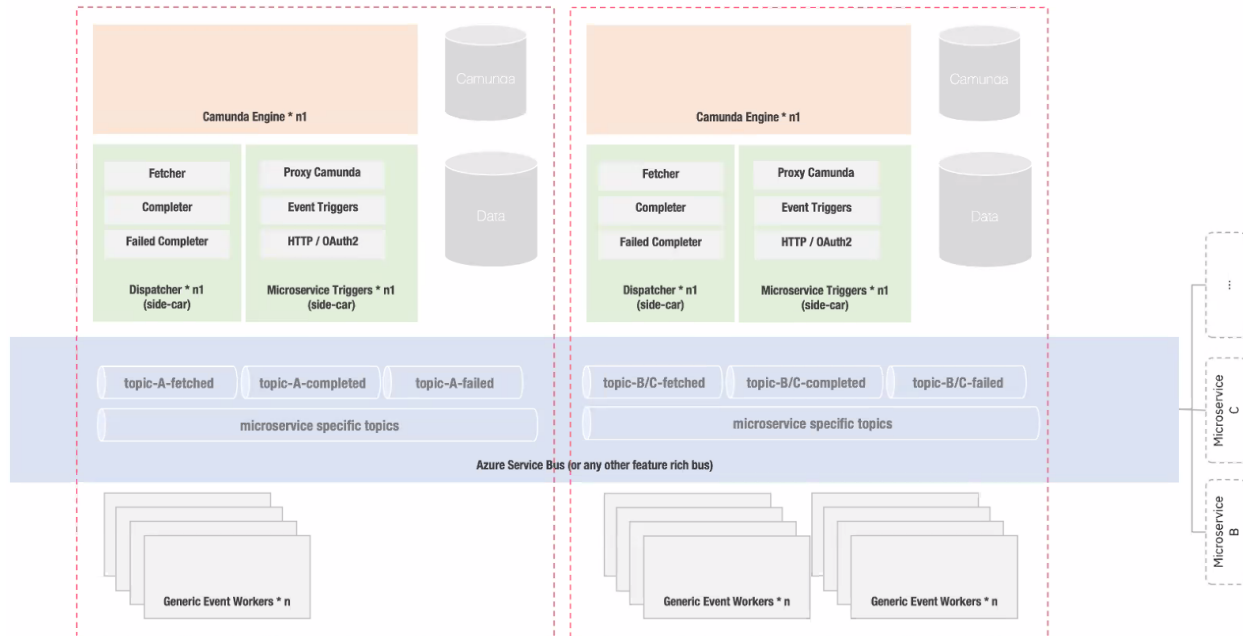
- ✓ Complex infrastructure setup
- ✓ Lock synchronization (message TTL and Camunda task TTL)
- ✓ Message duplication (consider intelligent service bus features to deduplicate)
- ✓ With External Service Architecture process execution is by default asynchronous



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
- ✓ Tight 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



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)

ExternalService_FrontOffice

General | Listeners | **Input/Output** | Field Injections | Extensions

Input Parameters

- > nxxbz_activity_unique_identifier ← 1be32feafe9a43c38b302ae189aa2b0b
- > nxxbz_activity_version ← 1.*

Output Parameters

No variables defined.

ExternalService_BackOffice

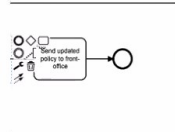
General | Listeners | **Input/Output** | Field Injections | Extensions

Input Parameters

- > nxxbz_activity_unique_identifier ← 56f63fa1a28742e88de47006fcb23c61
- > nxxbz_activity_version ← 2.*

Output Parameters

No variables defined.



ExternalService_FirstCobre

General | Listeners | **Input/Output** | Field Injections | Extensions

General

ID: ExternalService_FirstCobre

Name: Send policy change to back-office

Details

Implementation: External

Topic: nxxbz_activity_unique_identifier

External Task Configuration

Task Priority:

Asynchronous Continuations

- ☒ Asynchronous Before
- ☒ Asynchronous After
- ☒ Exclusive

Job Configuration

Job Priority:

Retry Time Cycle:

Documentation

Element Documentation:

CAMUNDA COMMUNITY SUMMIT 2021

topics	A JSON array of topic objects for which external tasks should be fetched. The returned tasks may be arbitrarily distributed among these topics. Each topic object has the following properties:
Name	Description
topicName	Mandatory. The topic's name.
lockDuration	Mandatory. The duration to lock the external tasks for in milliseconds.
variables	A JSON array of String values that represent variable names. For each result task belonging to this topic, the given variables are returned as well if they are
localVariables	
businessKey	
processDefinitionId	
processDefinitionIdIn	
processDefinitionKey	
processDefinitionKeyIn	Filter tasks based on process definition keys.
processDefinitionVersionTag	Filter tasks based on process definition version tag.
withoutTenantId	Filter tasks without tenant id.
tenantIdIn	Filter tasks based on tenant ids.
processVariables	A JSON object used for filtering tasks based on process instance variable values. A property name of the object represents a process variable name, while the property value represents the process variable value to filter tasks by.
deserializeValues	Determines whether serializable variable values (typically variables that store custom Java objects) should be deserialized on server side (default: false). If set to true, a serializable variable will be deserialized on server side and transformed to JSON using Jackson's POJOStream property introspection feature. Note that this requires the Java classes of the variable value to be on the REST API's classpath. If set to false, a serializable variable will be returned in its serialized format. For example, a variable that is serialized as XML will be returned as a JSON string containing XML.
includeExtensionProperties	Determines whether custom extension properties defined in the BPMN activity of the external task (e.g. via the Extensions tab in the Camunda modeler) should be included in the response. Default: false

Camunda 7.15.0

includeExtensionProperties

Determines whether custom extension properties defined in the BPMN activity of the external task (e.g. via the Extensions tab in the Camunda modeler) should be included in the response. Default: false

ExternalService_FrontOffice
General
Listeners
Input/Output
Field Injections
Extensions

Properties

Add Property +

Name	Value	
nxxbz_activity_unique_identifier	1be32feafe9a43c38b302ae189aa2b0b	x
nxxbz_activity_version	1.**	x

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.