Integrating Microservices with Mendix

Gonçalo Marcos, Bart Luijten

2018-09-27

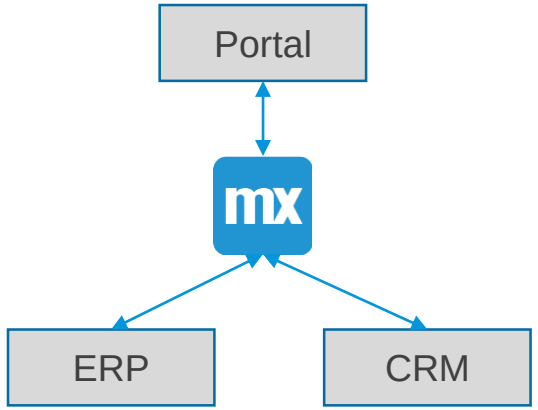
mendix

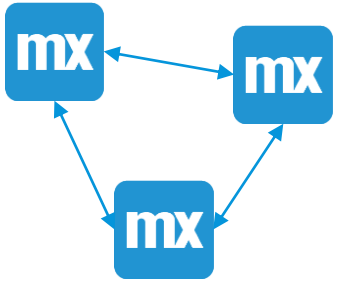
Agenda

- Integration Overview
- Microservices Overview
- Mendix Microservices Scenario 1 – UI Integration
- Demo 1 – UI Integration
- Some Integration Patterns
- Mendix Microservices Scenario 2 – Event State Transfer
- Demo 2 – Event State Transfer with Kafka
- Best Practices

Why Integration?

- Enterprise systems do not exist in isolation
  - Need to connect to CRM, ERP, Inventory, Billing, etc.
- Mendix is often used to augment or link existing systems
- Mendix favors microservices, which rely on integration





mendix

Google Definition of Microservices

*A **microservice architecture** is a method of developing **software applications** as a suite of*

- *independently deployable*
- *small*
- *modular services*
- *in which each service runs a unique business process and*
- *communicates through a well-defined, lightweight mechanism*
- *to serve a business goal*

Google Definition of Micro Services

*A **microservice architecture** is ing **software applications** as a suite of*

- *independently deployable*
- *small*
- *modular services*

ervice runs a unique business process and

through a well-defined lightweight mechanism

- *to serve a business goal*

Microservices should prioritize autonomy

"Small" is bigger than you think: "one pizza" = 3-4 people

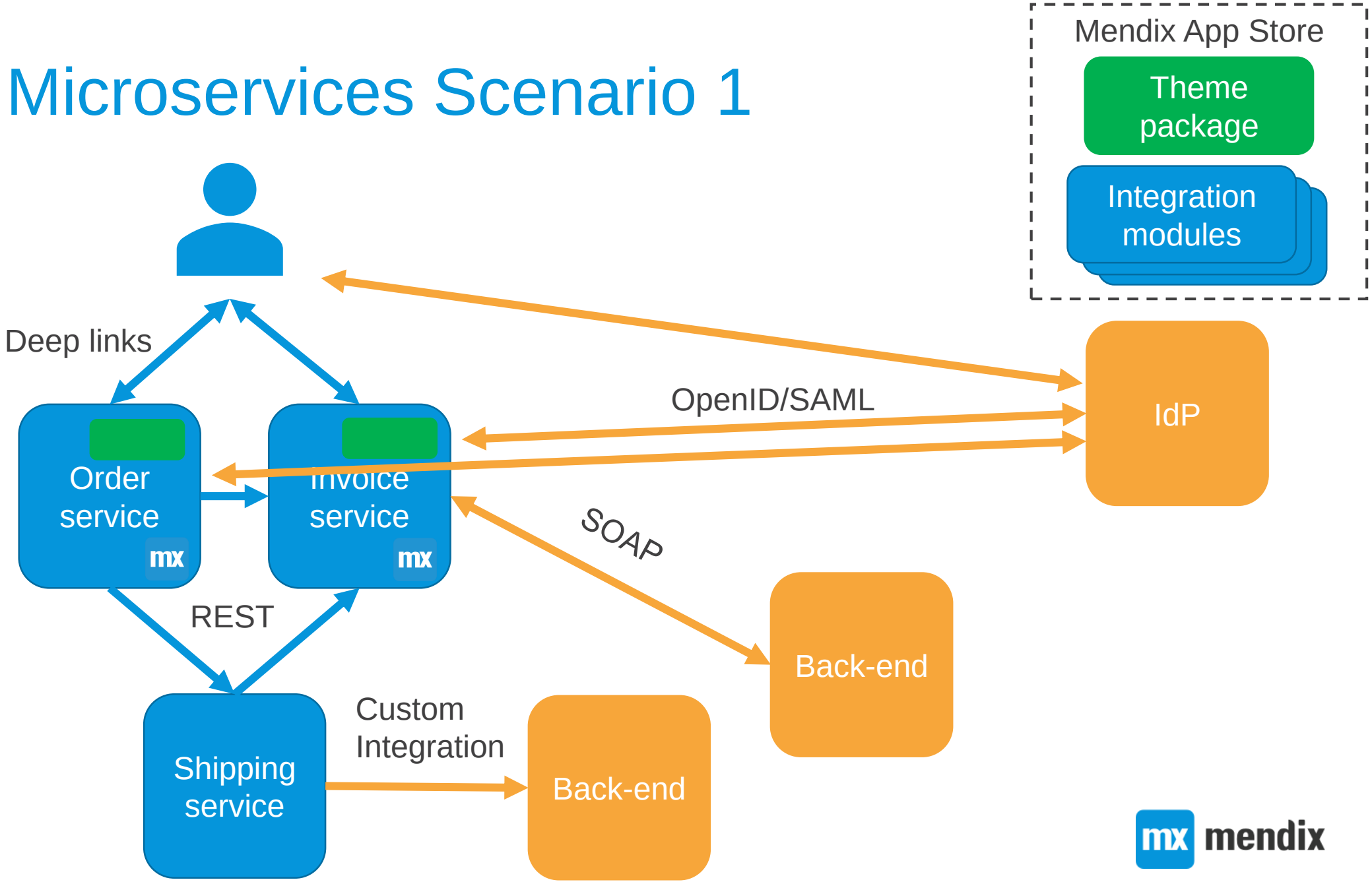
A microservice serves a business goal, over the full stack

mx mendix

Integration Categories

- Data Integration
  - SOAP, REST, OData, App Services
  - Events, Message Queues, Pub-Sub (e.g. Kafka, JMS, RabbitMQ)
- UI Integration
  - Same UI theme package
  - Deep links
  - SSO
- Infrastructure Integration
  - CI/CD(Mendix platform APIs), Monitoring tools

Mendix Microservices Scenario 1



Demo 1 – UI Integration

mendix

Integration Patterns

(A)Synchronous communication

Does the caller of a service get an answer?

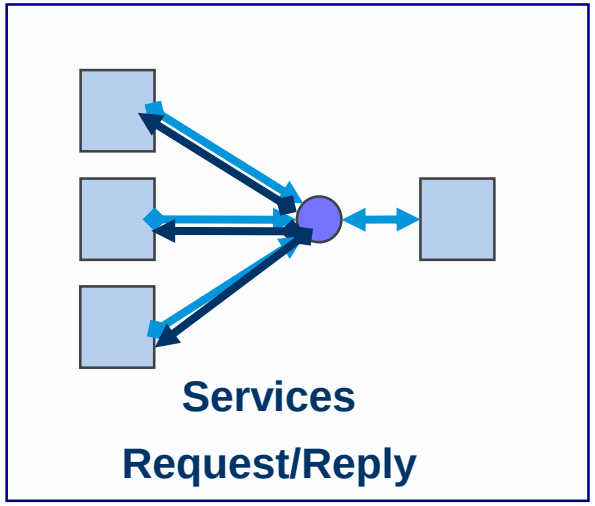
Synchronous

- Send a message and block while waiting for a response
- Straightforward to achieve and understand

Asynchronous

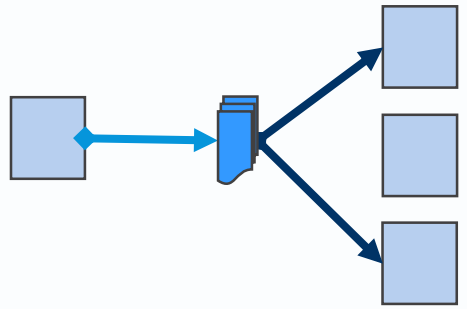
- Fire and (maybe) forget – Can poll for answer
- Great for long-running jobs (as we don’t have to wait)
- More complex to implement and understand

Request-Reply (Sync)



- Pros:
  - Easy to implement
  - Result “immediately” available
  - Guaranteed delivery is easy
- Cons:
  - Decoupling requires intermediate layer
  - Caller has to deal with service downtime
    - For example, through retry mechanism
  - Synchronous, need to wait for completion
- Also requires decision on whether to cache query result

Event-Driven / Publish-Subscribe (Async)

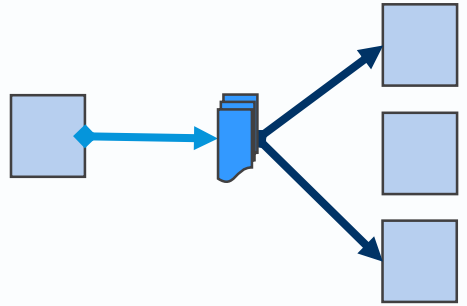


Event Driven

Publish-Subscribe

- One publisher, usually many subscribers
- When there's a change
  - Publisher sends out an event/message to broker
  - Broker notifies consumers
    - Consumers either pull data directly from Publisher
    - Data gets pushed to Consumers from the broker (Event State Transfer)
- Interesting pattern for microservices
  - Decouple services
  - Event-driven state transfer

Event-Driven / Publish-Subscribe (Async)

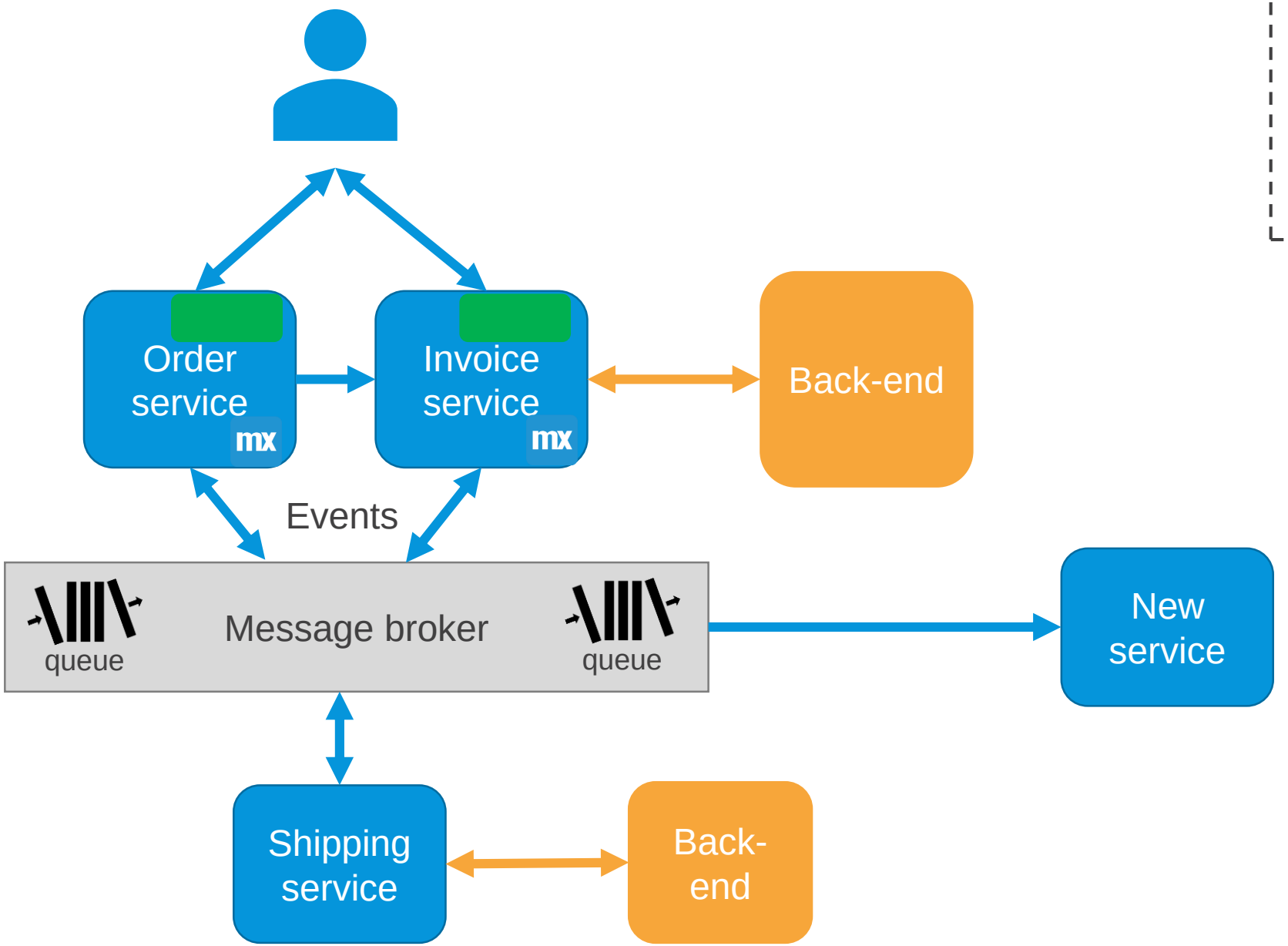


Event Driven

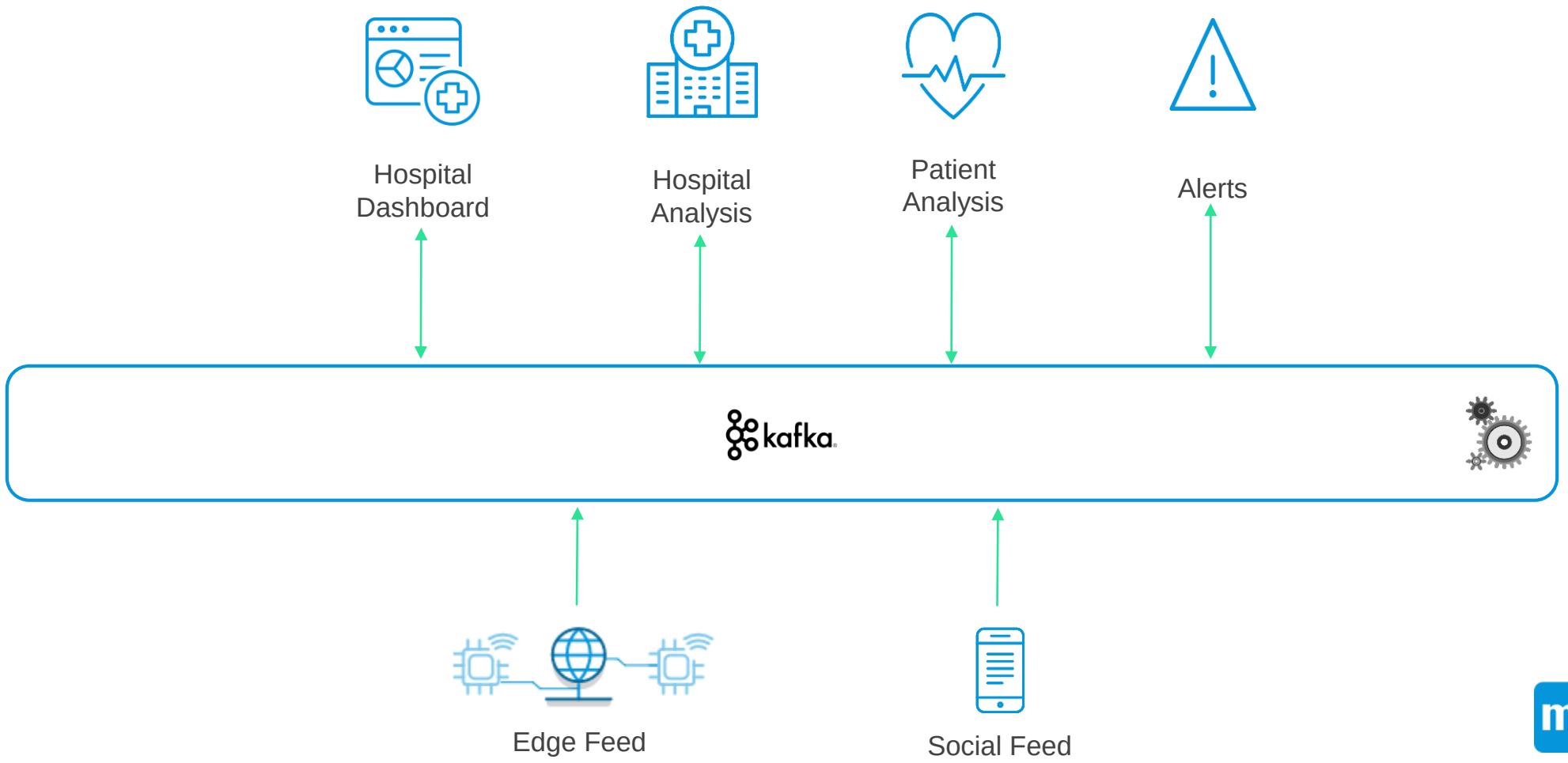
Publish-Subscribe

- Pros:
  - Decouple publisher from subscriber, which is good for back-end processes (no end-user involved)
  - Sender does not have to “wait” so User can continue working
  - Allows message throttling
- Cons:
  - Need to use queue
  - Error feedback (e.g. data validation) is not immediate to a user
  - Implementation is hard
    - Guaranteed delivery is much harder than for request/reply
    - Message ordering can be an issue (Competing messages in multithreaded implementation)

Mendix Microservices Scenario 2



Hospital Management Solution



mx mendix

Demo 2 – Events with Kafka

mendix

Best Practices

Best Practices

- Seamless UI Integration
  - Use the same UX package to provide a unified experience
  - Use deep links to redirect users between services
  - Use SSO to transfer sessions

Best practices

- Microservices integration: Prevent control flow coupling
  - Use event-driven integration to prevent latency stacking

Best practices

- Each shared object has a global GUID to allow easier data sync
- Correlation IDs in messages, included in logging

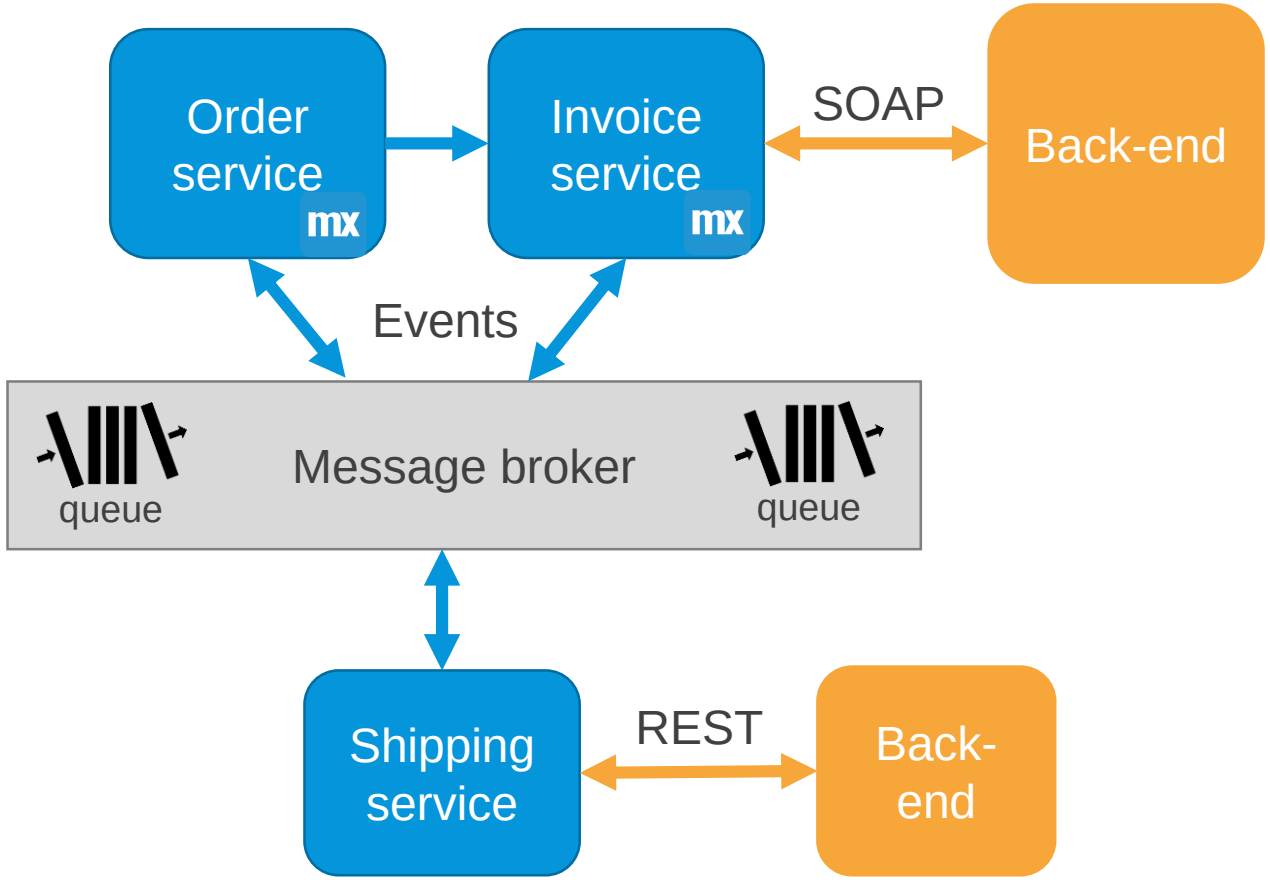
```
07:31:03 [Order service] DEBUG - Integration: [de89fa3] Sending completed order object.

07:31:04 [Invoice service] DEBUG – Integration: [de89fa3] Received new order object.
```

mx mendix

Best practices

- You are not limited to a single pattern between two apps



mx mendix

Best practices

- Pick the messaging type that best fits the situation

- Events: less urgency, less need for guaranteed delivery
  - Best decoupling
  - Best served via queues
  - But has only eventual consistency (not real-time)
- Poll for Data / Push Data: immediate need for update
  - Best for real-time updates
  - Calling system can directly manage failures
  - But calling system needs to wait on updates to complete (synchronous)

Flexible Integration Options to Support Next Generation Low-Code Applications

Andrej Koelewijn / July 19, 2018

Mendix Blog

Mendix.com/blogs

Platform Evaluation Guide

Search

Overview
- Welcome to Mendix

App Lifecycle
- Requirements
- Developing in Mendix
- DevOps

App Capabilities
- UX & Multi-Channel Apps
- Data Management
- Integration
  - **Integration Overview**
  - Data Mappings
  - Service Consumption
  - Service Exposure
  - External Data

Integration Overview

Table of contents

1 How Can I Integrate with Mendix?

Integration has always been an important part of application development, but the topic has not received much attention in the marketplace for developing apps until now. The maturation in cloud-native and as-a-service software options has increased the need for APIs that connect disparate applications together to exchange data, trigger events, and orchestrate workflow processes.

At Mendix, we have made investments in the platform to make it easy to connect applications and services with each other in a variety of ways, most recently with REST. And while REST has become a standard for building APIs and web-services for applications to talk, it's important to recognize that other web service protocols, standardized frameworks, and APIs remain valid options, depending on the problem you are trying to solve.

Mendix Platform Evaluation Guide

mendix.com/evaluation-guide

mx mendix

Questions?