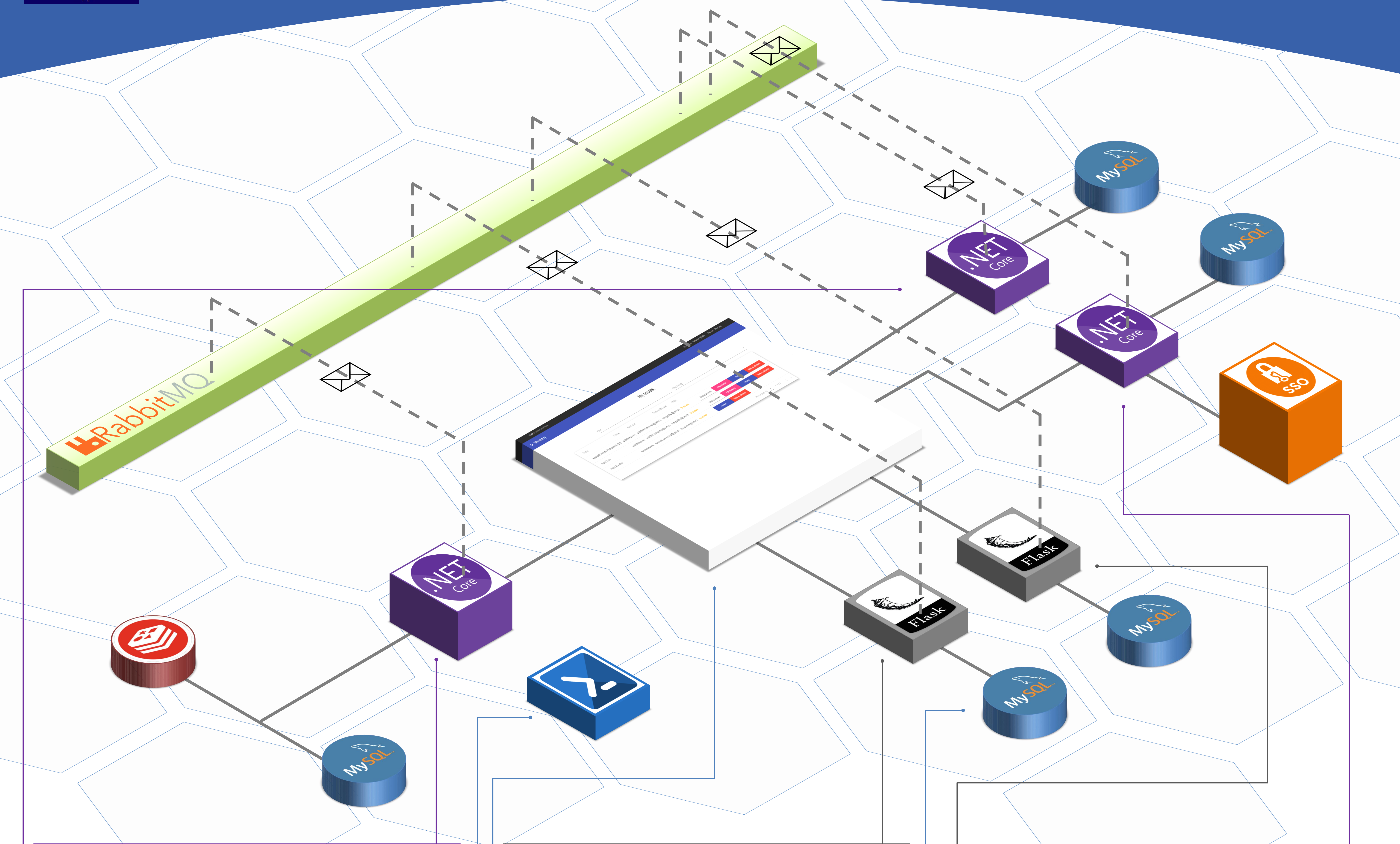
Winventory: microservices architecture case study

Sebastian Bukowiec, Pawel Tadeusz Gomulak (CERN, IT-CDA)



Applications microservice
Provides functionality about applications. Besides MySQL the service uses Redis in-memory database for caching

Pools microservice
Manages pools sent to the user and gathers the answers

Event Bus (RabbitMQ)
Publish/Subscribe channel, asynchronous communication across microservices

Notifications microservice
Manages communication with the users including templates and application grouping

Frontend
SPA based on Angular 7 with Angular Material providing features like: lazy-loading, XSS protection, http interceptors, dependency injection and routing

PowerShell worker
Feeds the systems with data from Computer Management Framework (CMF)

CERN SSO (OAuth2)
Authentication for the application

Identity microservice
Authenticates against CERN SSO, provides role based authorization and generates JWT token

Users microservice
Manages users and groups

Databases
Each microservice has its own database

- Facilitates loose coupling
- No blocking by other service
- Independent development

Overview

Software inventory and communication tool, microservices architecture case study.

The Winventory system gathers user inputs to build a comprehensive inventory of software assets across CERN. The system is built on a microservices architecture pattern, which separates the application into multiple and independently deployable units that can be individually developed, tested and deployed.

Purpose

- Identify systems approaching their end-of-life e.g. Windows Server 2008 R2. Inform the users, gather information about the use cases and help to take appropriate actions.
- Gather statistics about licensed software and facilitate communication with users to understand various use cases and gives the overall license cost.

Data producer

Winventory currently has one data producer - Computer Management Framework (CMF), a custom software installed on every Windows machine that is a member of the CERN domain.

Data is collected once a day. Only data less than three months old is considered. Missing data such as user and responsible are fetched from the network database.

Technology stack

Containerized deployment using Docker and OpenShift Container Platform.

Technology-agnostic system. Two different languages and frameworks: ASP .NET Core services written in C# communicating easily with Flask and Celery services written in Python.

Resilience and fault-tolerance in synchronous communication has been improved by using an open source Polly circuit-breaker library.

The server-side code pushes content to connected web application clients (frontend) as it happens, in real-time using open source library SignalR.

The object-relational mapping (ORM) is implemented using open source frameworks Entity Framework for C# and SQLAlchemy for Python.

DevOps

The GitLab CI/CD pipeline is used to automatically build new Docker images and deploy them to the OpenShift Container Platform after new code is pushed to the Git repository.

To facilitate local development, a Docker Compose is used to build and run the Winventory multi-container application locally on the developer's computer.