

# Architecture of a Serverless Cloud Application

IEEE 23rd International Symposium on Computational Intelligence and Informatics  
(CINTI 2023)

Ondrej Habala, **Martin Bobák**, Martin Šeleng, Viet Tran, Ladislav Hluchý, Lukáš  
Ivica

Institute of Informatics of the Slovak Academy of Sciences

November 20, 2023



INSTITUTE OF INFORMATICS  
SLOVAK ACADEMY OF SCIENCES



# Outline

Introduction

Function as a Service

Pilot Application - Airport Visibility

OpenWhisk

Airflow

Architecture

Summary and Future Work



# Introduction

- ▶ We present part of the construction of an airport visibility meteorological application based on the Function-as-a-Service paradigm
- ▶ monolithic version already in use at airports
- ▶ our work: transformation into a server-less application
- ▶ goal of the paper: to provide other developers with a methodology on how to use FaaS systems



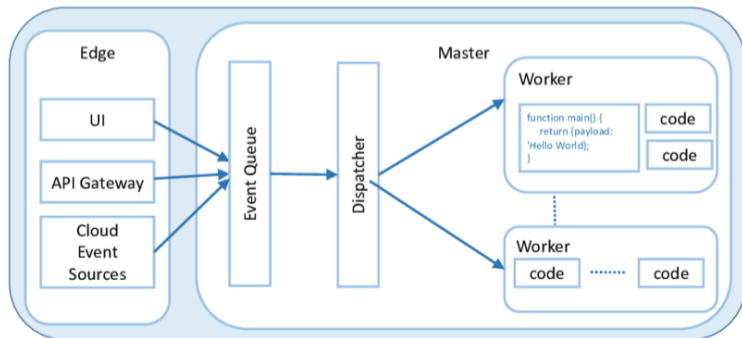
# Motivation

- ▶ For the research partner:
  - ▶ part of long-standing research in cloud computing
  - ▶ acquisition of new know-how
  - ▶ opportunity to verify research results in real life use
- ▶ For the application developer:
  - ▶ modularization of application, allowing several different deployments (different requirements, functionality, pricing)
  - ▶ unburdens customers from hardware acquisition and maintenance
  - ▶ easier development of new functionality, even spanning new domains
  - ▶ opportunity to gain know-how in modern computing paradigms



# What is Function as a Service?

- ▶ Subset of serverless computing that provides a platform allowing developers to write and deploy applications without building and maintaining the underlying infrastructure
- ▶ infrastructure management (resource provisioning, maintenance and regular update of base operating systems) are the responsibility of the Cloud provider
- ▶ developer focuses only on application code and logic



# Advantages of FaaS

- ▶ Automatic scaling: functions are scaled automatically, independently, and instantaneously according to the actual demands by the cloud provider. That relieves developers from concerns of high traffic or heavy use.
- ▶ Cost efficiency: Users have to pay only for the computing resources they really use, not for idle resources that are often reserved for handling possible high demands in the typical IaaS (scaling by cloud provider).
- ▶ Quick development: developers don't have to manage infrastructure, they can focus only on the code, reducing the cost of development and the time to market.



## Disadvantages of FaaS

- ▶ Potential vendor lock-in: The application codes are built on the top of a concrete FaaS platform and difficult to port to another vendor.
- ▶ Difficulties for testing: The codes are running on the top of a FaaS platform, it may make difficulties for creating local test environments for applications



# Current FaaS Frameworks

- ▶ The first commercial provider offering FaaS is Amazon AWS with **AWS Lambda** platform,
- ▶ followed by Google with **Google Cloud Functions**.
- ▶ We will focus on two open-source platforms: **Apache OpenWhisk**, originally by IBM,
- ▶ and **OpenFaaS** (by a company of the same name)



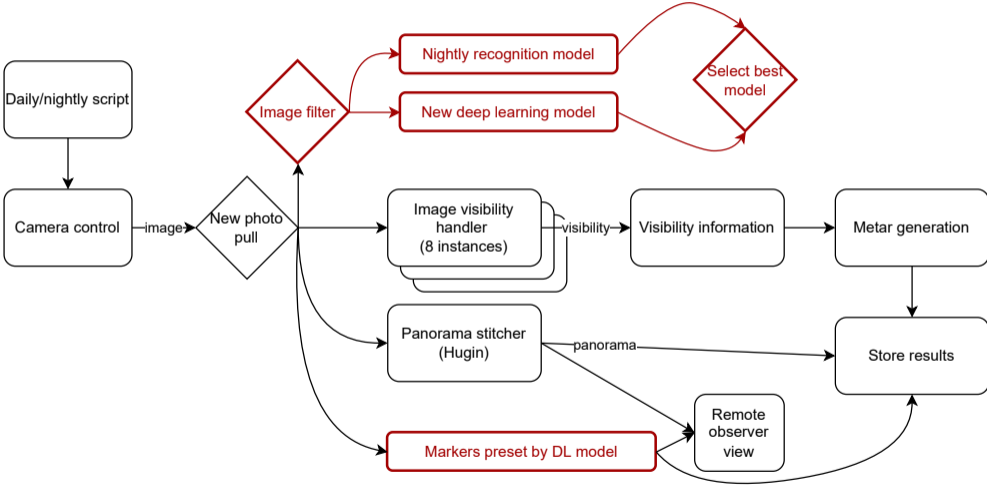


## Application - Motivation

- ▶ Visibility is a crucial element in the safety of all kinds of transport
- ▶ Almost 50% of all aircraft accidents is due to weather conditions
- ▶ main cause of weather-related aviation accidents is reduced visibility
- ▶ better visibility information also leads to better traffic management
  - ▶ reduced fuel consumption
  - ▶ reduced flight delays



# Application - Architecture



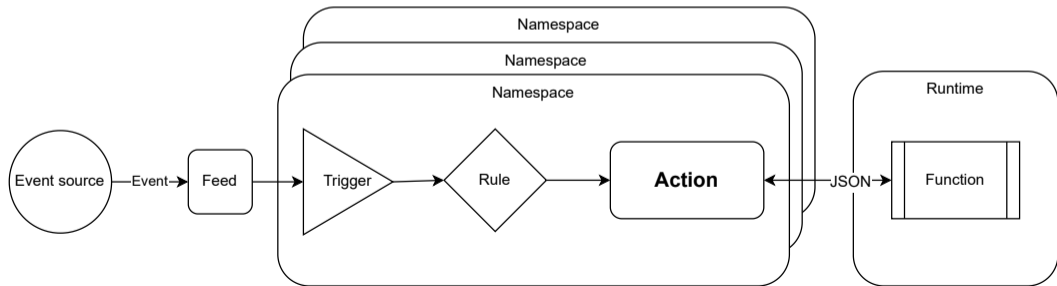
# OpenWhisk - Origin and Use

- ▶ free and open implementation of the Function-as-a-Service paradigm
- ▶ tied to AWS Lambda service, first presented in November 2014
- ▶ started by Rodric Rabbah of IBM Research
- ▶ initially developed at IBM Research as Whisk
- ▶ later renamed to OpenWhisk, made OSS and transferred to the Apache Software Foundation Incubator
- ▶ can be installed in several ways, we have used Kubernetes
  - ▶ Helm chart for OpenWhisk is available
  - ▶ local installation of wsk command-line tool also necessary on the developer's machine



# OpenWhisk - Programming Model

- ▶ an event-driven system
- ▶ event from an even source feeds into a trigger
- ▶ a trigger uses rules to execute an action
- ▶ uses REST API to accept new events
- ▶ accepts functions in several languages (Java, Python, PHP, Go, Ruby...) as well as black-box code



# OpenWhisk - Application

- ▶ application is being implemented according to the architecture shown above
- ▶ so far implemented:
  - ▶ ImageVisibilityHander, as a Java action
  - ▶ Visibility info from 8 xmls, also as a Java action
  - ▶ Panorama stitch, as a black-box docker action, since we use a 3rd party software (Hugin<sup>1</sup>)
- ▶ In the case of the Panorama stitch action, we have created a specific docker image, based on OpenWhisk's Docker Runtime image, adding the Hugin software for panorama-stitching
- ▶ Panorama stitch could also be done as a Python action with a custom docker image for OpenWhisk's Python actions

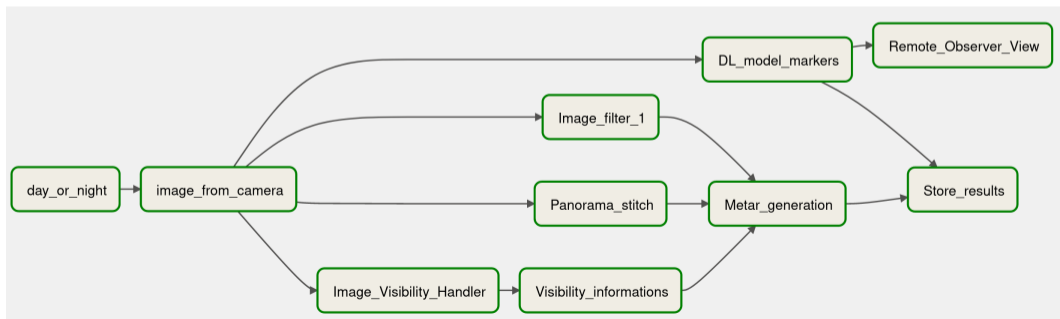
---

<sup>1</sup>Hugin - Panorama Photo Stitcher, <https://hugin.sourceforge.io/>



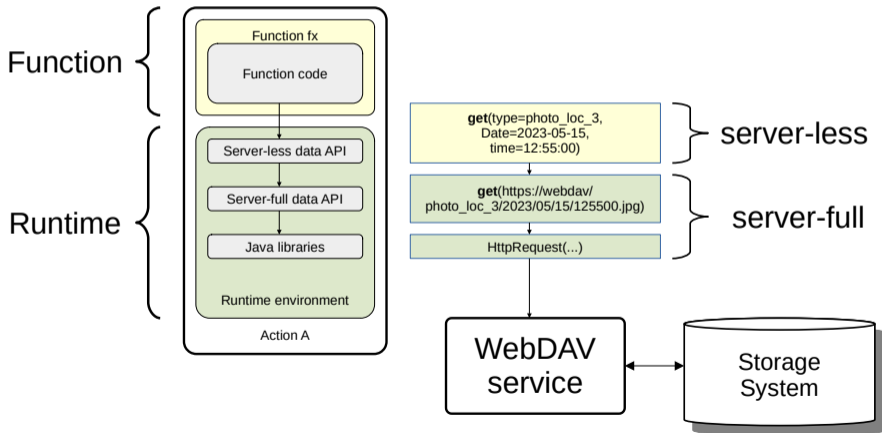
# Airflow

- ▶ open source framework for lightweight serverless functions management
- ▶ amalgamates individual tasks into a workflow which is expressed as a directed acyclic graph
- ▶ in our case the tasks are serverless functions
- ▶ the resulting workflow characterizes the relations between its tasks which also defines their execution order.



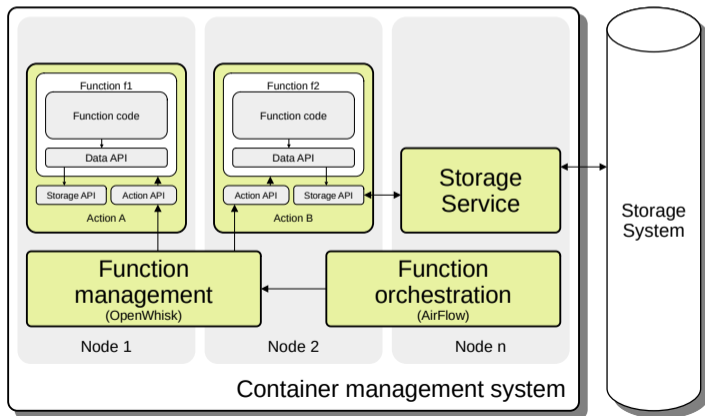
# Storage architecture

- ▶ a server-less layer API
- ▶ a server-full layer API
- ▶ storage service



# Serverless Application Architecture

- ▶ server-less application - function code, server-less data API layer
- ▶ server-full layer - data API, action API
- ▶ storage service connected to a storage system
- ▶ function management (OpenWhisk) and function orchestration (AirFlow)





# Summary

- ▶ presented the transformation of a monolithic application into the serverless cloud domain (FaaS)
- ▶ we have chosen the open source OpenWhisk platform, it allows the event-driven execution of actions
- ▶ so far transformed parts of the application
- ▶ we have defined a generalized architecture of a server-less application



## Future Work

- ▶ transformation of additional parts of the application into FaaS actions
- ▶ continue work of defining a detailed methodology of the transformation of a legacy application to a server-less one
- ▶ a generalized server-less application architecture with additional components



*Thank you!*

For questions please write to [martin.bobak@savba.sk](mailto:martin.bobak@savba.sk) or [ondrej.habala@savba.sk](mailto:ondrej.habala@savba.sk)

