GlobalLogic®

Real time processing in microservices

GL Tech Talk #3
2018.10.25

Miłosz Mazurek
Robert Gonciarz

About authors

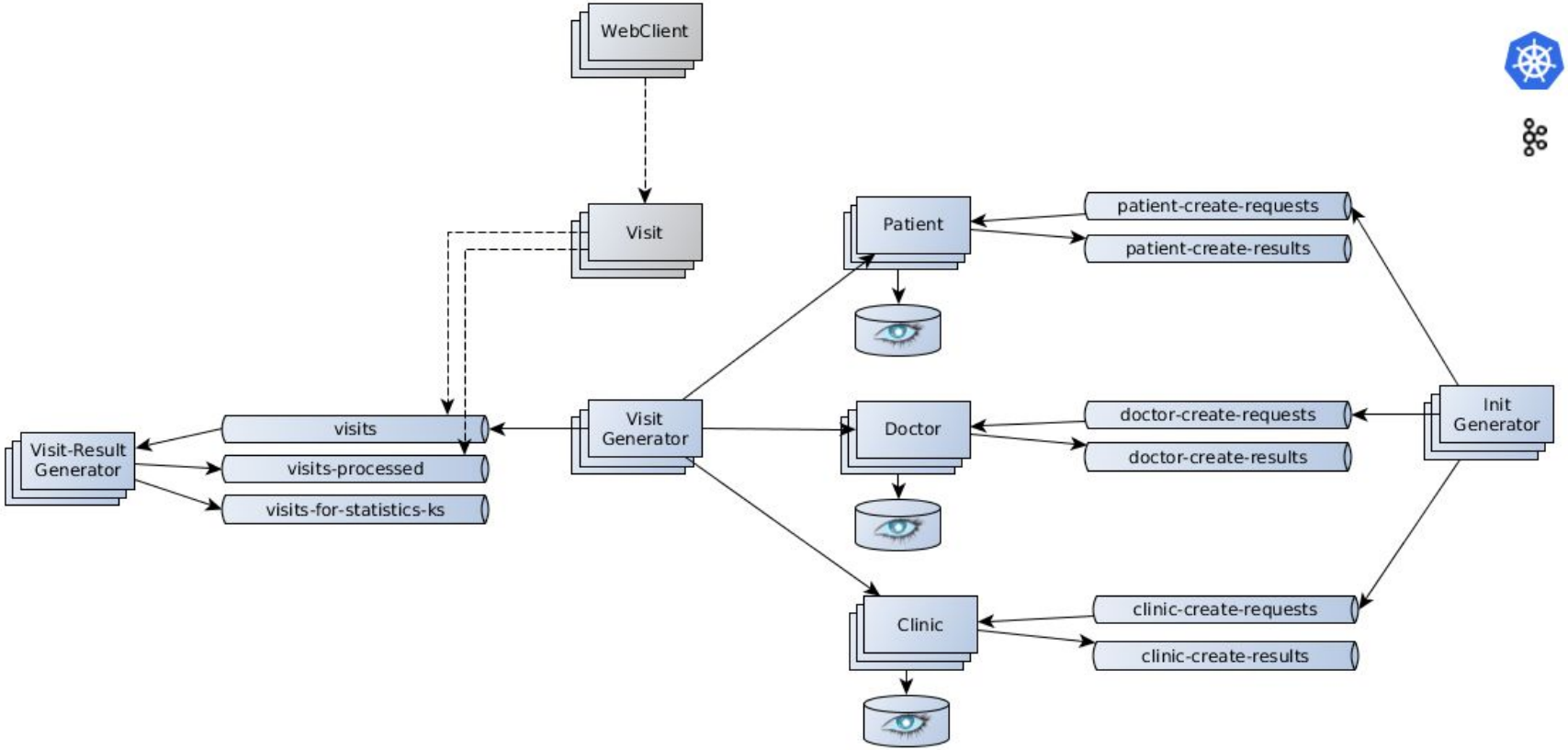
Miłosz Mazurek

Robert Gonciarz

Agenda

1. Introduction
2. Application Architecture
3. Demo
4. Spring Cloud Stream
5. WebFlux
6. Kafka
7. Kafka Streams
8. Kubernetes
9. Resources

Architecture



Demo

Spring Cloud Stream

What is Spring Cloud Stream

Spring framework which enables to build event/message driven architecture in microservices. It defines many types of binders which allow to connect/integrate a lot of communication frameworks such as: Kafka, Kafka Streams, RabbitMQ, WebFlux, Spring MVC.
It works using publisher-subscriber pattern.

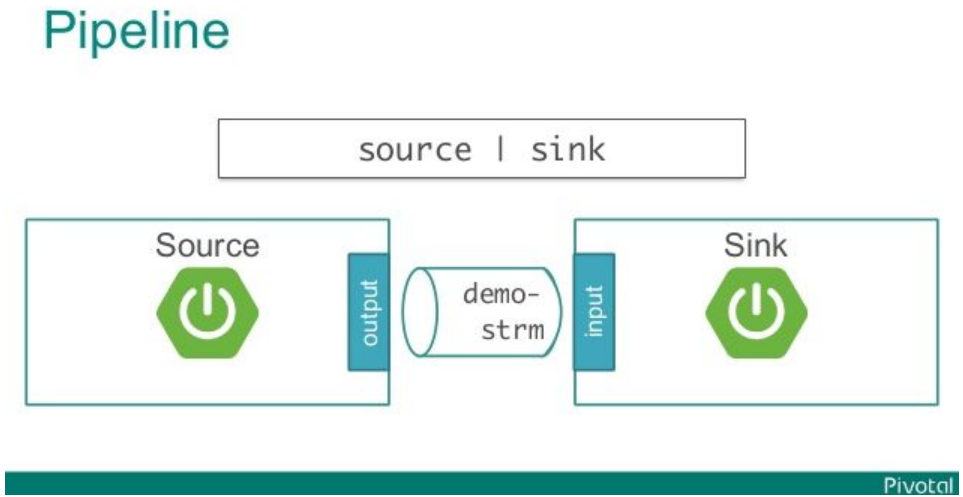
Main Advantages

1. Conditional messages dispatching
2. Very easy way to create message handlers/endpoints

Spring Cloud Stream

Defining endpoints

1. Defining bindings using: `Source`, `Sink`, and `Processor` interfaces. The most common practice is or **custom** interfaces.
2. Enabling binding using @EnableBinding. Spring Cloud creates automatically an implementation of this interfaces.



Spring Cloud Stream

Content negotiation

JAVA

```
@EnableBinding(Sink.class)
@EnableAutoConfiguration
public static class TestPojoWithAnnotatedArguments {

    @StreamListener(target = Sink.INPUT, condition = "headers['type']=='bogey'")
    public void receiveBogey(@Payload BogeyPojo bogeyPojo) {
       // handle the message
    }

    @StreamListener(target = Sink.INPUT, condition = "headers['type']=='bacall'")
    public void receiveBacall(@Payload BacallPojo bacallPojo) {
       // handle the message
    }
}
```

Why Reactive?

Why using Reactive frameworks?

- Not intended to allow you to process your requests or data faster
- Capacity to serve more request concurrently
- handle operations with latency (remote server)
- dealing natively with time and latency
- Non-blocking
- requests only the amount of data it is able to process
- Stream aggregations

Spring WebFlux

('org.springframework.boot:spring-boot-starter-web')
->
('org.springframework.boot:spring-boot-starter-webflux')

```
Flux<String> flux = Flux
        .fromIterable(words)
        .flatMap(word -> Flux.fromArray(word.split("")))
        .zipWith(Flux.range(1, Integer.MAX_VALUE),
            (string, count) -> String.format("%2d. %s",
count, string));
```

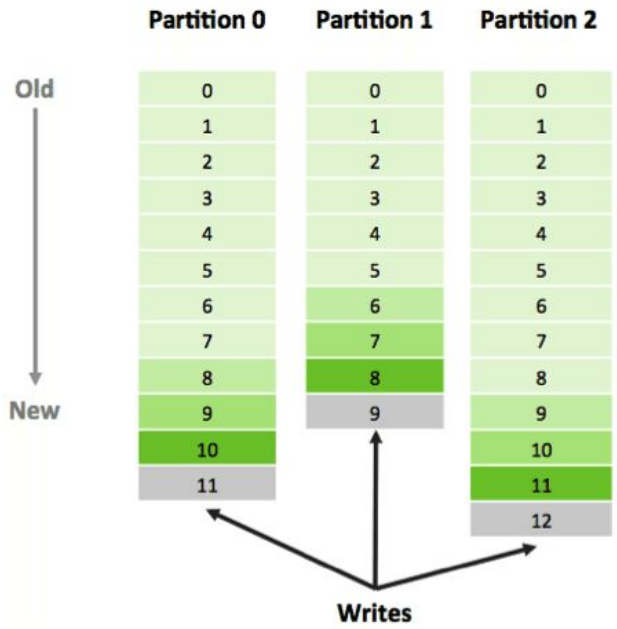
Kafka

Main advantages

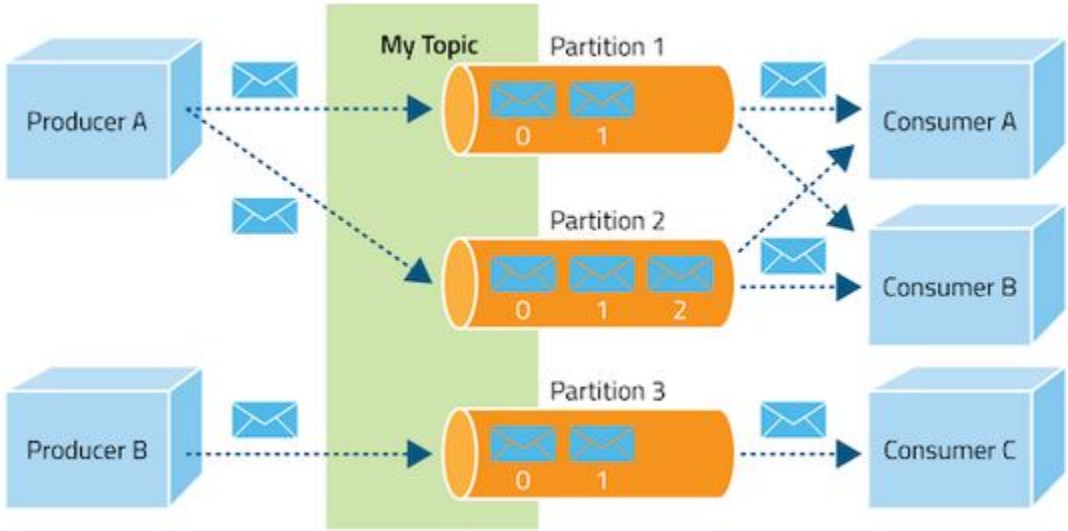
1. Partitions - enables parallel processing, keeps messages order for every partition
2. Consumer groups
3. Very efficient- high-throughput and low-latency
4. Stores messages for a certain period of time. It enables to read them even if microservice isn’t working during sending.
5. Message consumers and producers are independent

Kafka partitions

1. Kafka keeps message order for each partition
2. Every producer writes messages to the end of the partition
3. Every consumer reads messages from the beginning of the partition



Topics, partitions and consumer groups



Kafka Streams

What is Kafka Streams

Library which is dedicated to process real-time data feeds. It has is very efficient because of high-throughput, and low-latency platform for handling real-time data feeds.
Both data input and output are in Kafka.

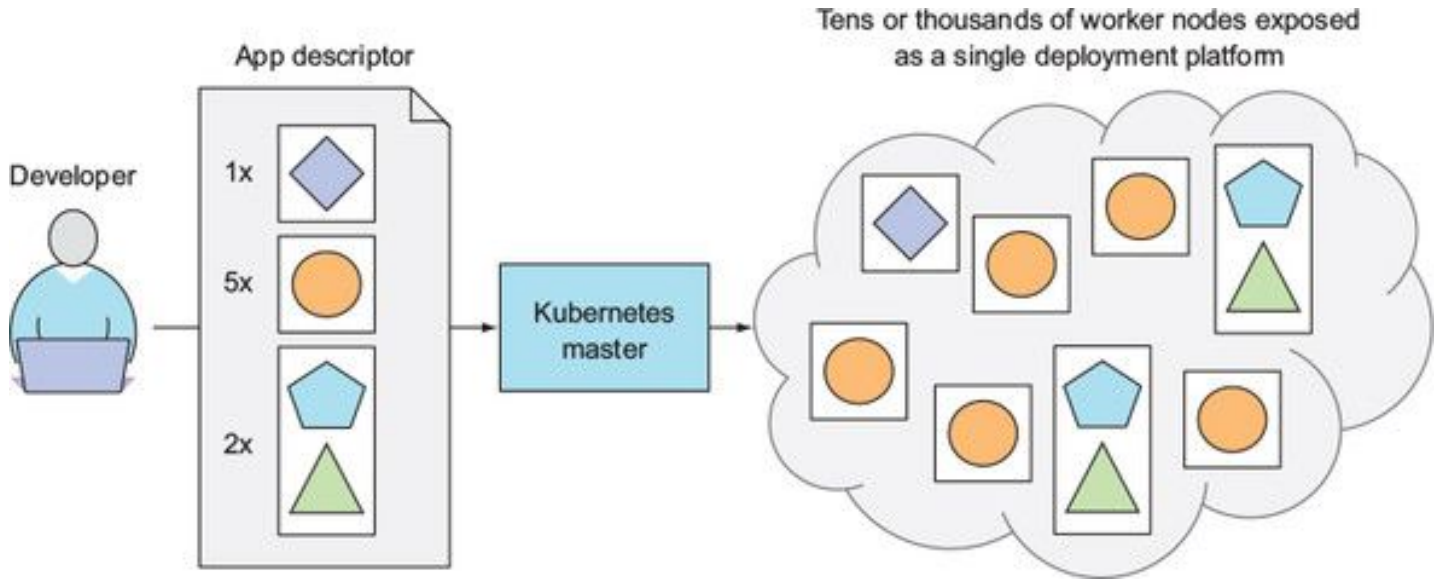
Main advantages

1. Doesn’t require any clusters such as Spark or Storm.
2. Easy to configure - just a library configured using dependency in Maven or Gradle.
3. Very good tool to cooperate with machine learning tools/data mining
4. Allows creating big flows of data processing

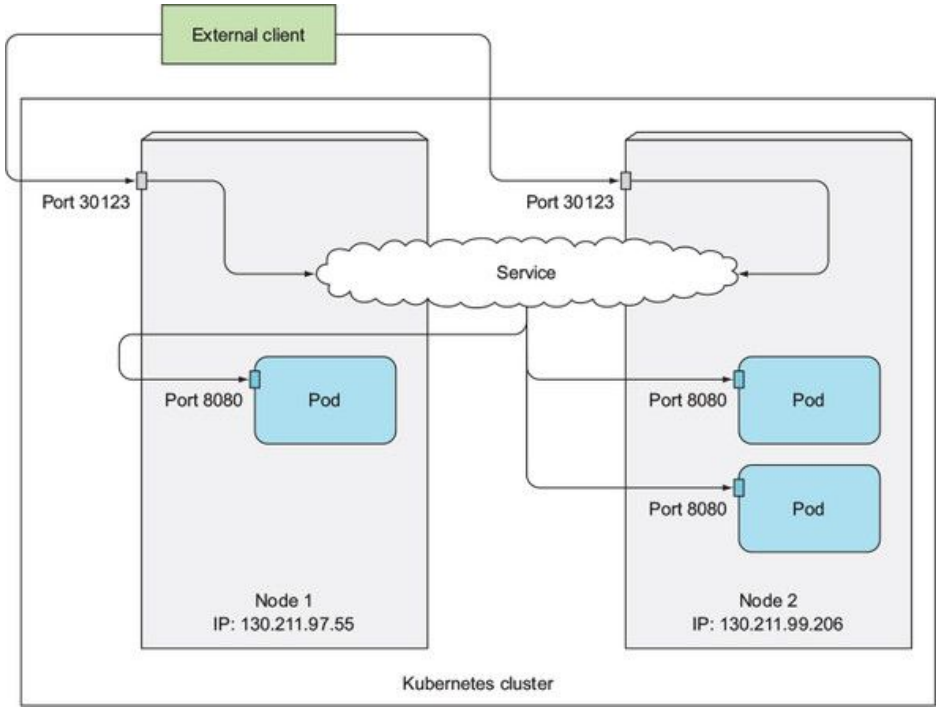
Main features

1. Joining streams (KTables) by keys - similar to joining tables in databases.
2. Many types of agreagations which allow to create different statistics and analisys
3. KTable vs KStream

Kubernetes


Tens or thousands of worker nodes exposed
as a single deployment platform
App descriptor
Developer
1x
5x
2x
Kubernetes
master

Kubernetes cluster



Resources: github.com/globallogic-poland