

# Reactive Programming Reactive System





Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1

View Activity Log 10+

...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata

3



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc

@somkiat.cc

Home Posts Videos Photos

Liked Following Share ...

+ Add a Button



**[https://github.com/up1/  
workshop-reactive-programming](https://github.com/up1/workshop-reactive-programming)**



# Topics

Why and What Reactive ?

Reactive architecture vs Programming

Spring Reactive

Demo

Testing

Q/A

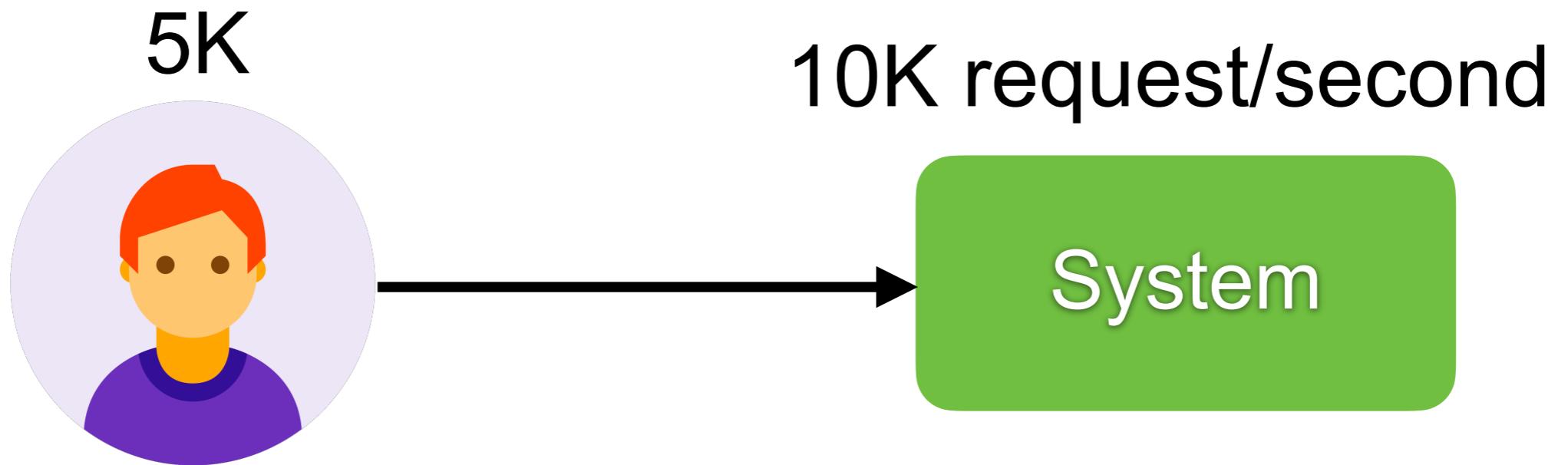


# Why and What

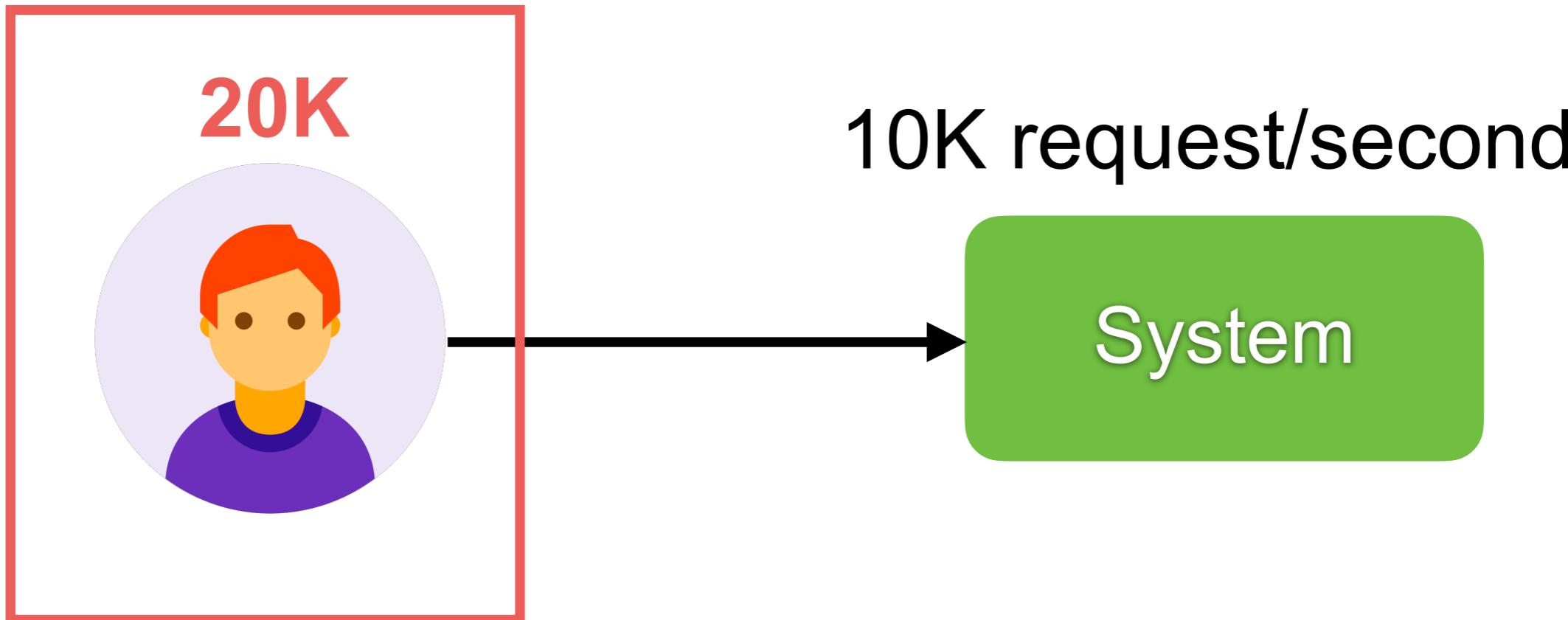




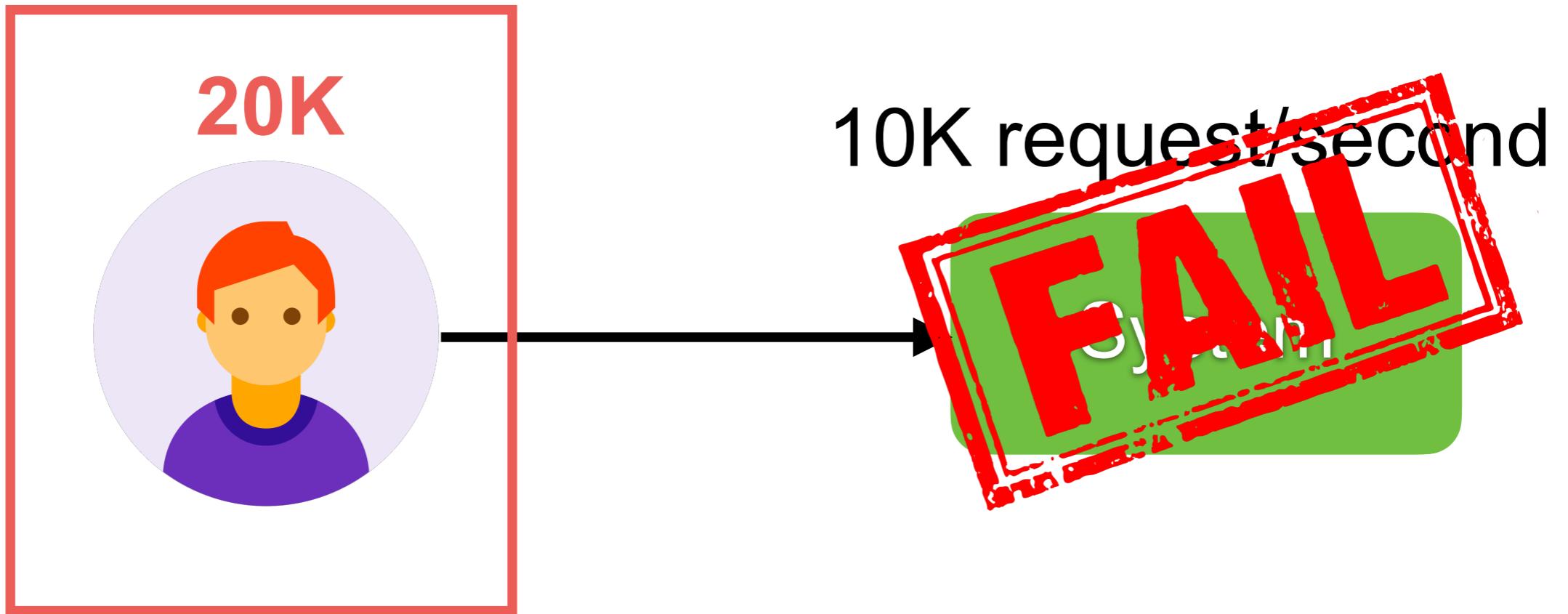
# Why Reactive ?



# Why Reactive ?



# Why Reactive ?



# Problems

Slow response

Outage system

User/business dissatisfied

Potential customer and money were lost



# Problems

I/O tasks

Working with Database  
Read and Write file



# Thread per request model

Thread blocked until the task is completed

More threads == More memory

Scalable problem !!

Bad for utilize resources



**Need better system**  
**Need better principles**



Scalable

Robust

Responsive

Elasticity



# Reactive System

Responsive

Resilient

Elastic

Message  
driven

Async messaging

<https://www.reactivemanifesto.org/>



# Reactive ...

React to events => **Event/message-driven**

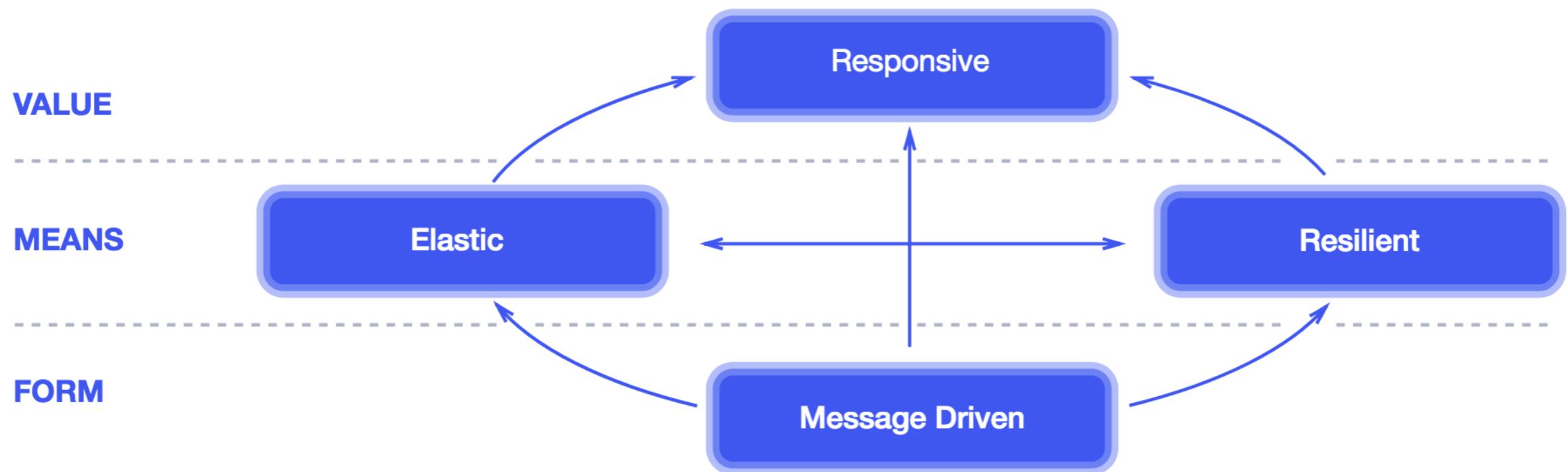
React to load => **Elastic**

React to failure => **Resilient**

React to users => **Responsive**



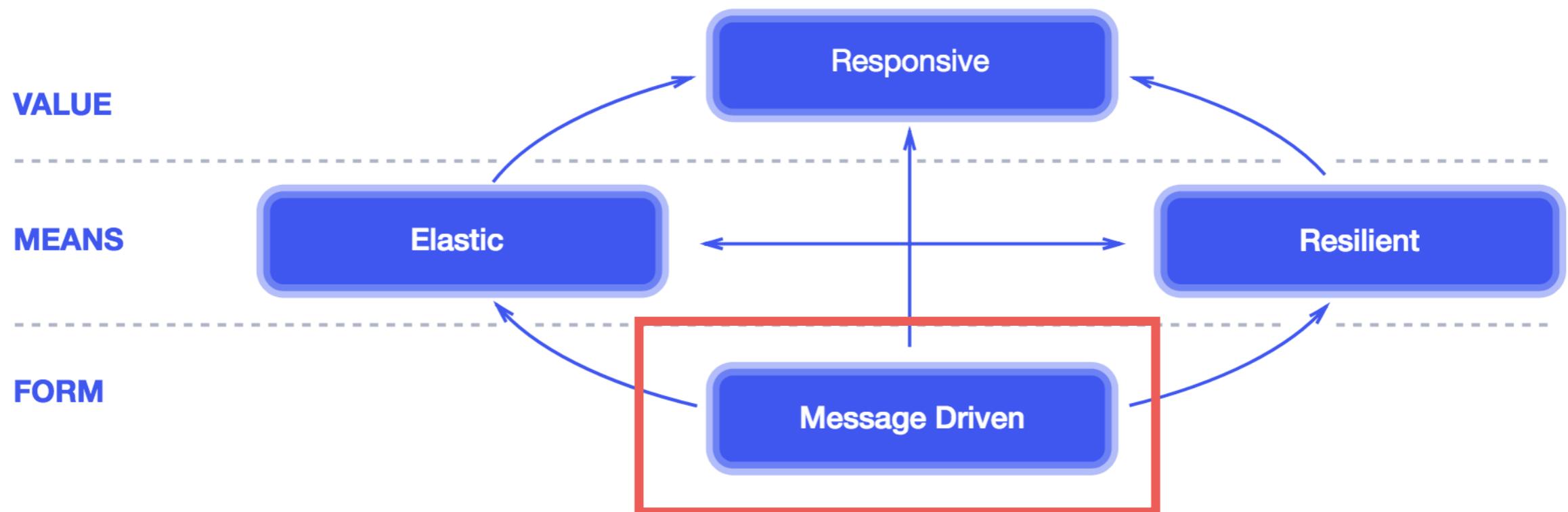
# Reactive Manifesto



<https://www.reactivemanifesto.org/>



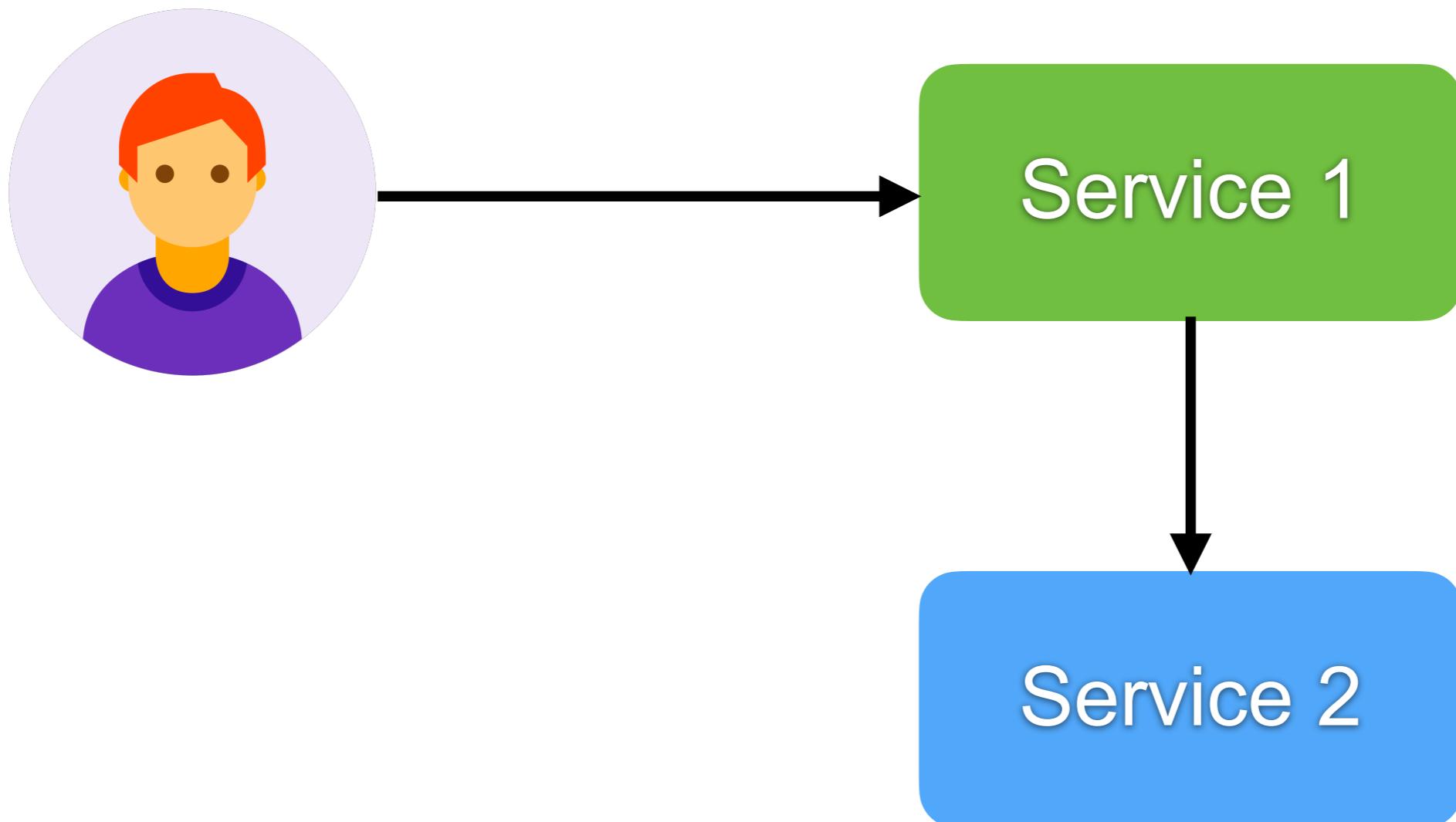
# Reactive Manifesto



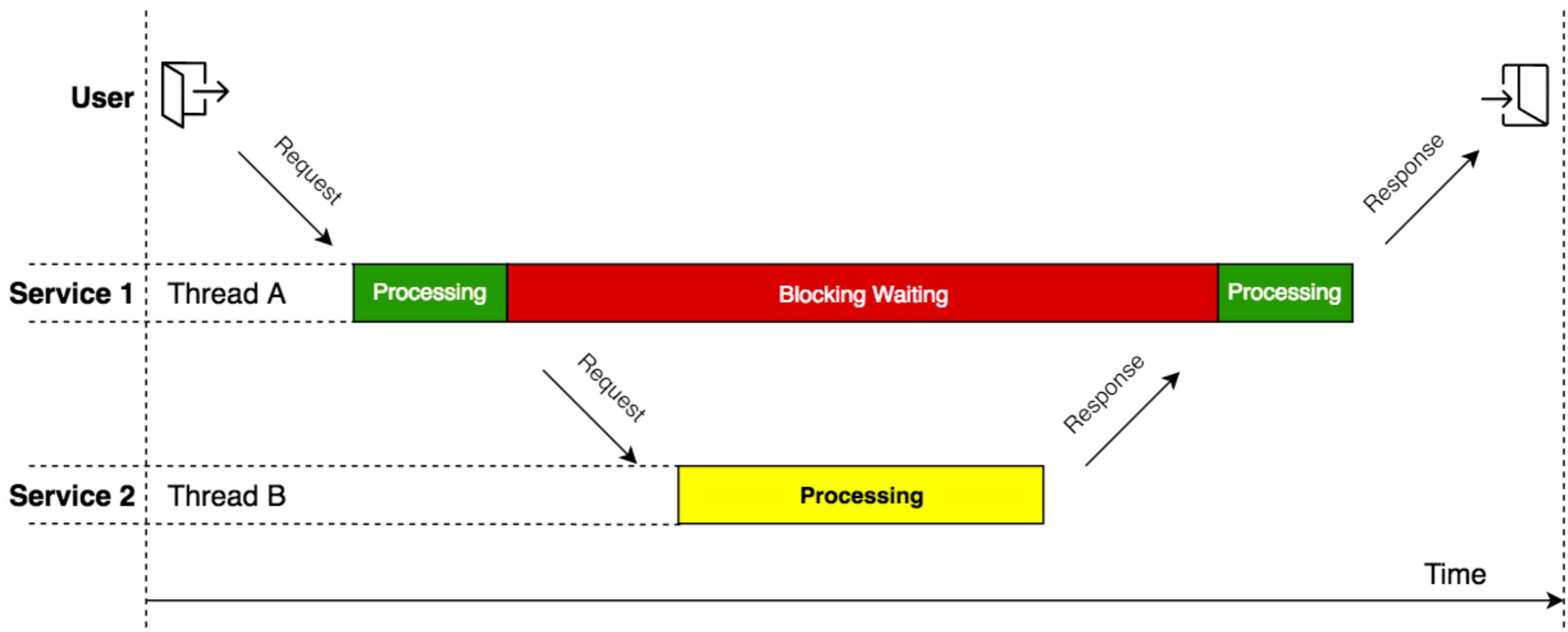
<https://www.reactivemanifesto.org/>



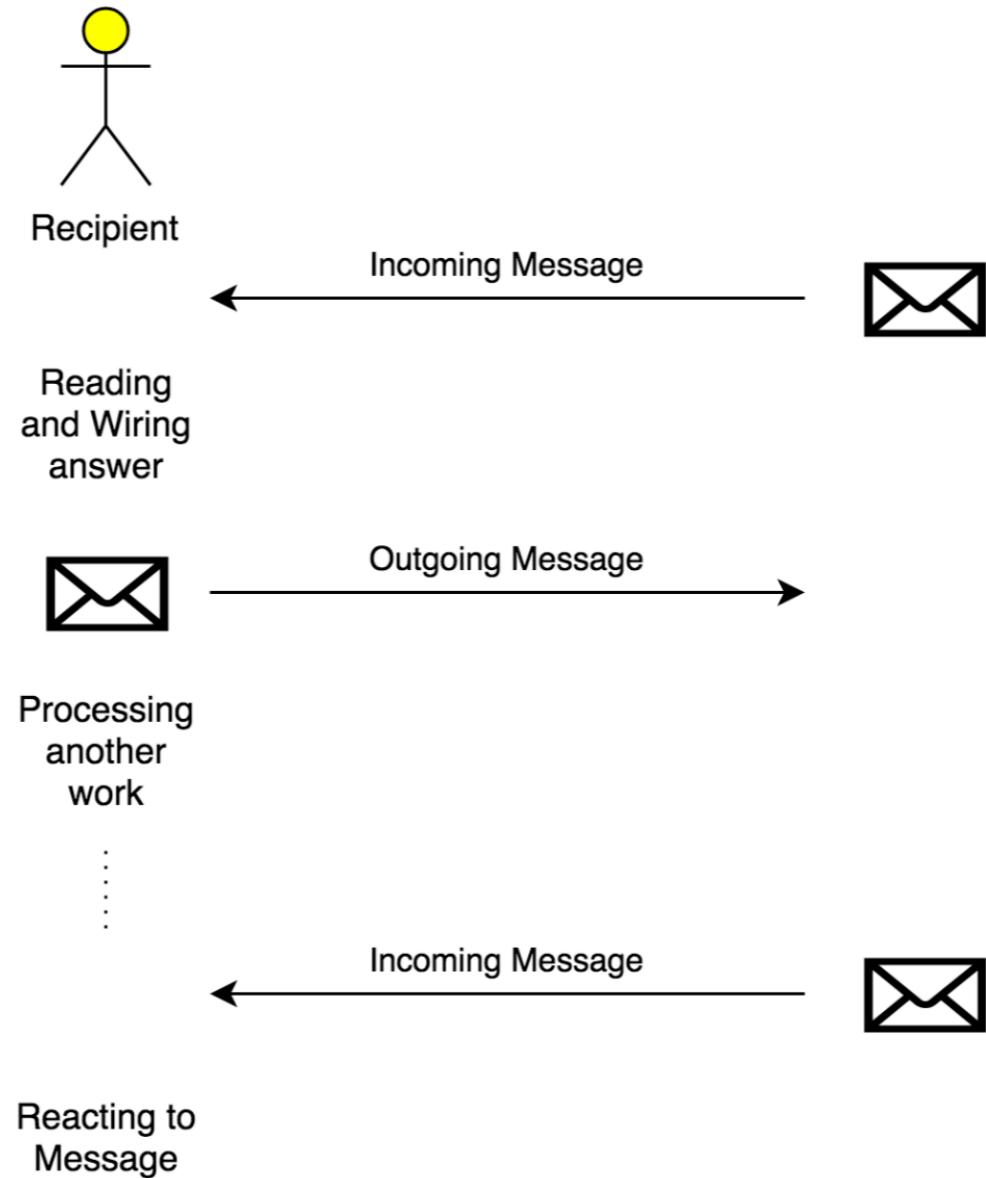
# Message-driven communication



# Message-driven communication



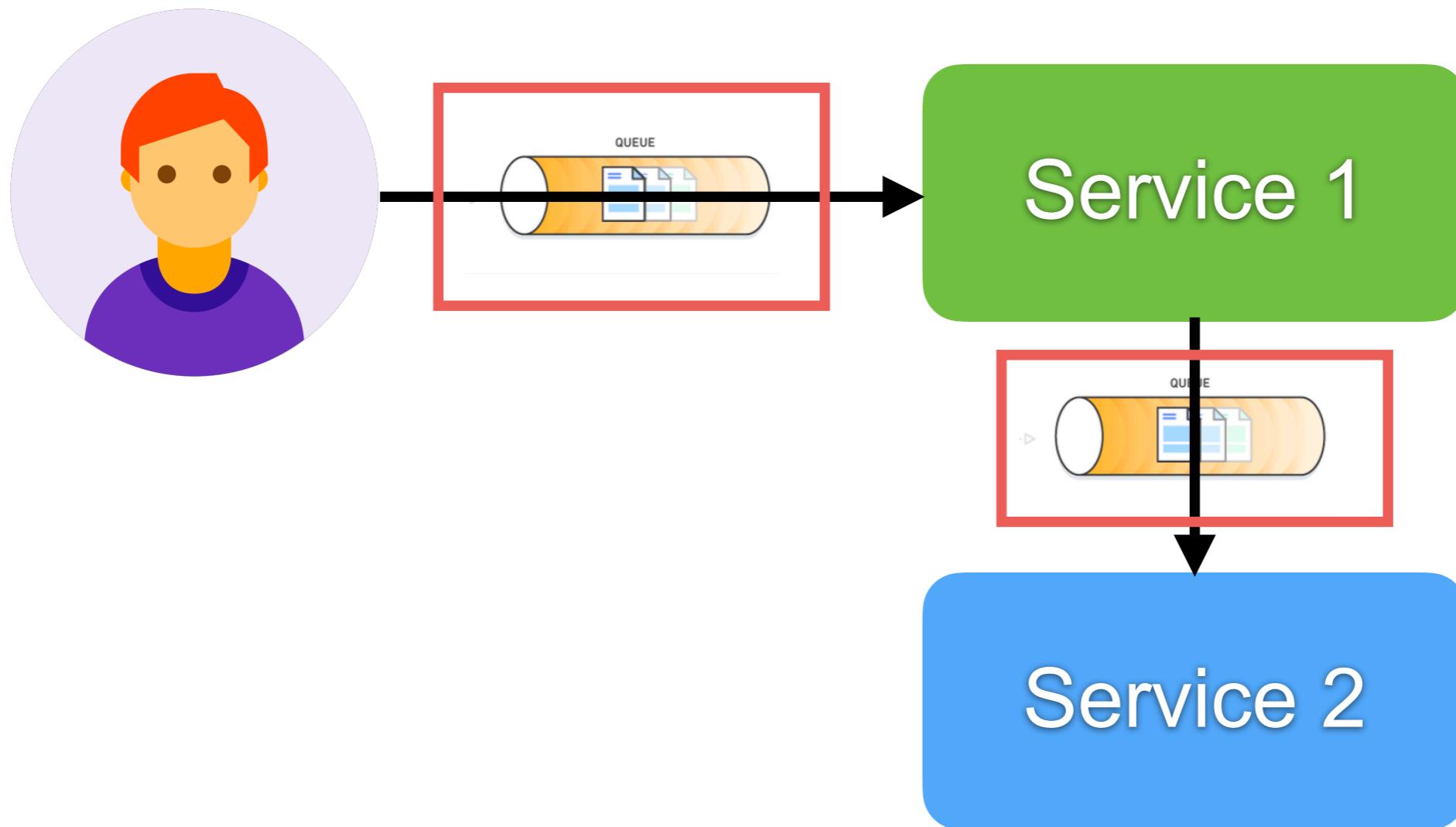
# Non-blocking message communication



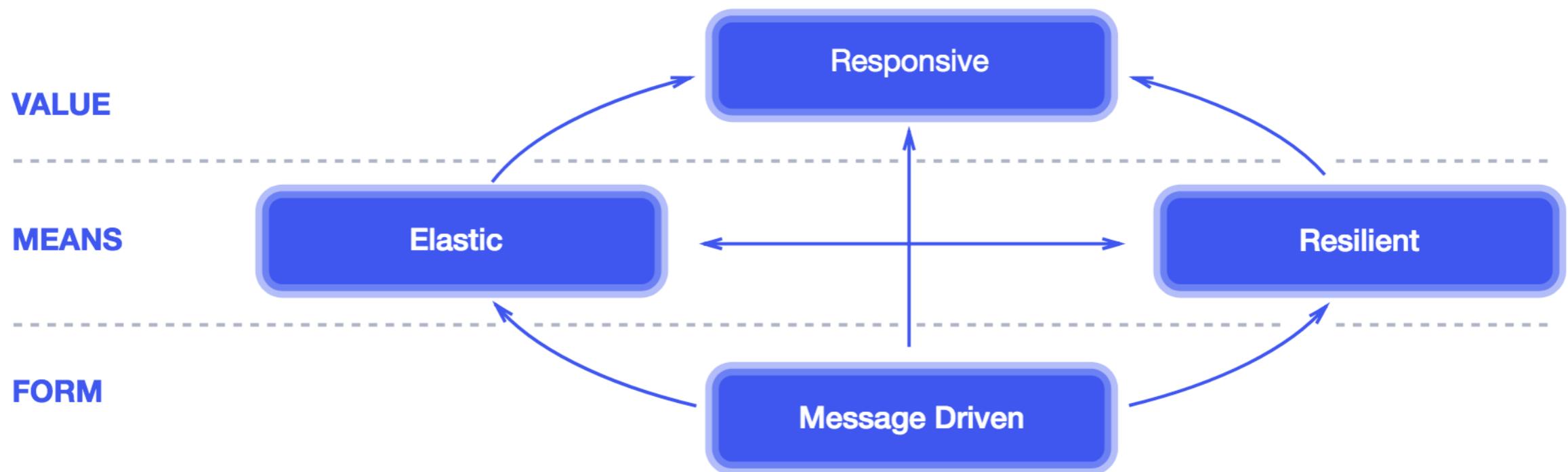
# **Efficient resource utilization when communicating between services in distributed system**



# Using message broker



# Reactive System



<https://www.reactivemanifesto.org/>



# Blocking

Sync

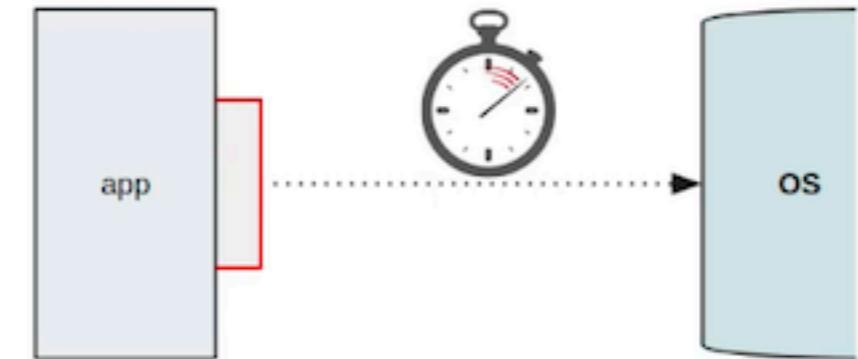
Async

Non-Blocking

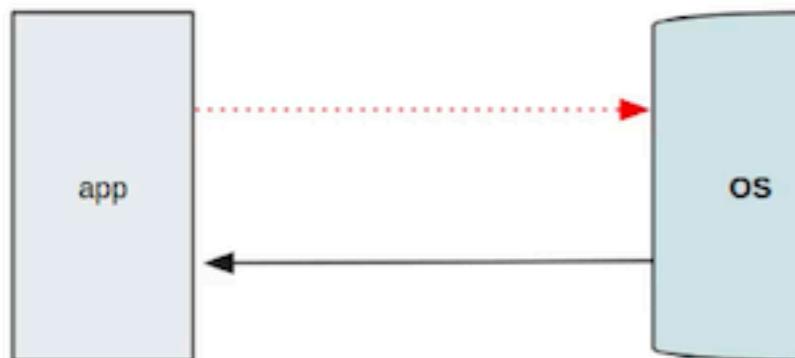




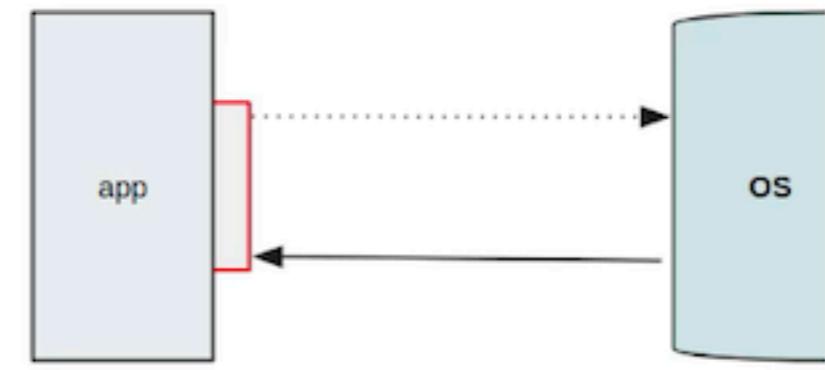
sync + blocking



async



non-blocking



non-blocking + async

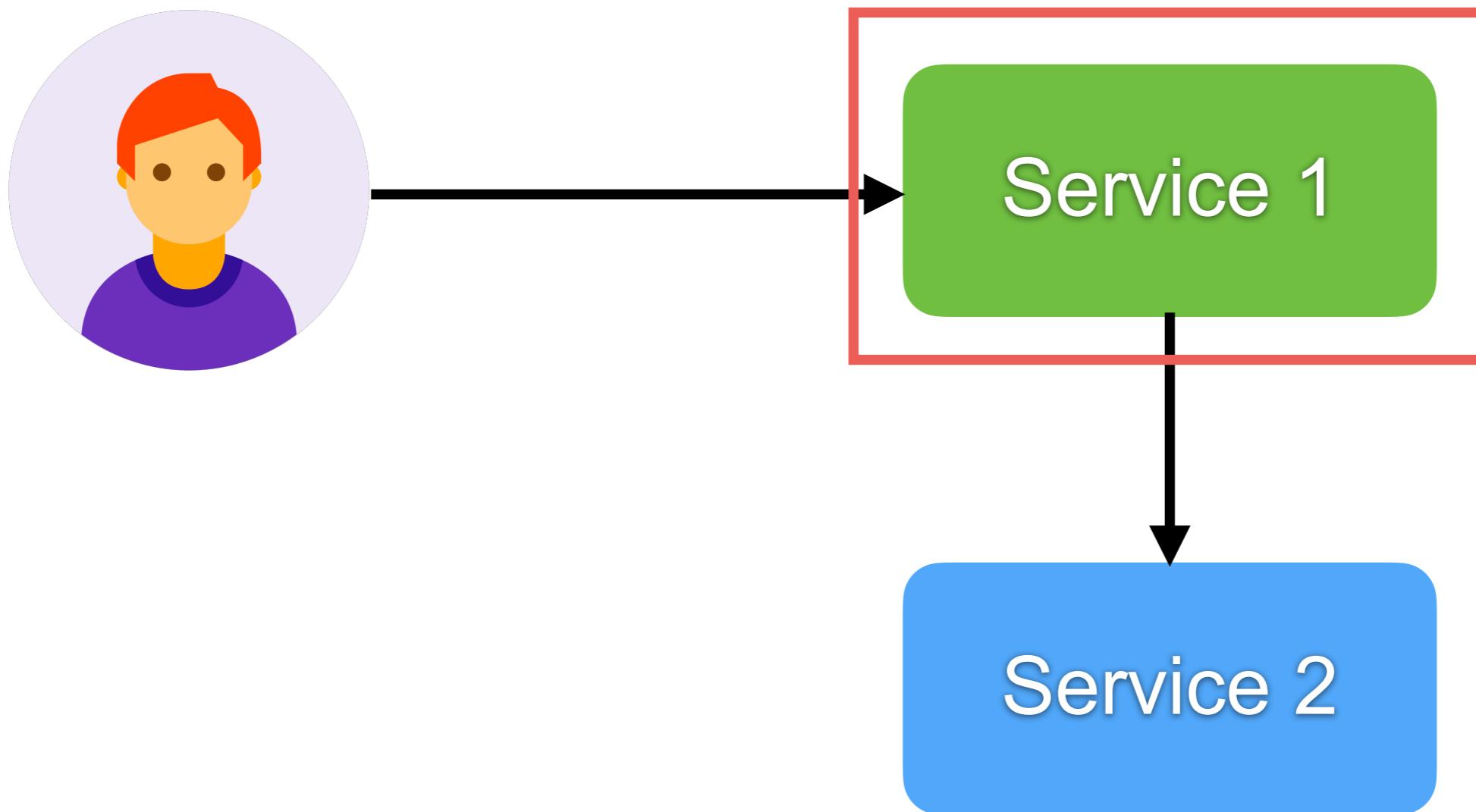


# Reactive programming

[https://en.wikipedia.org/wiki/Reactive\\_programming](https://en.wikipedia.org/wiki/Reactive_programming)



# Reactive programming



# Reactive programming

Implementation technique

Subset of Asynchronous programming

Decompose the problem into multiple discrete step

Asynchronous and non-blocking

**Dataflow / Stream programming**



# Reactive programming

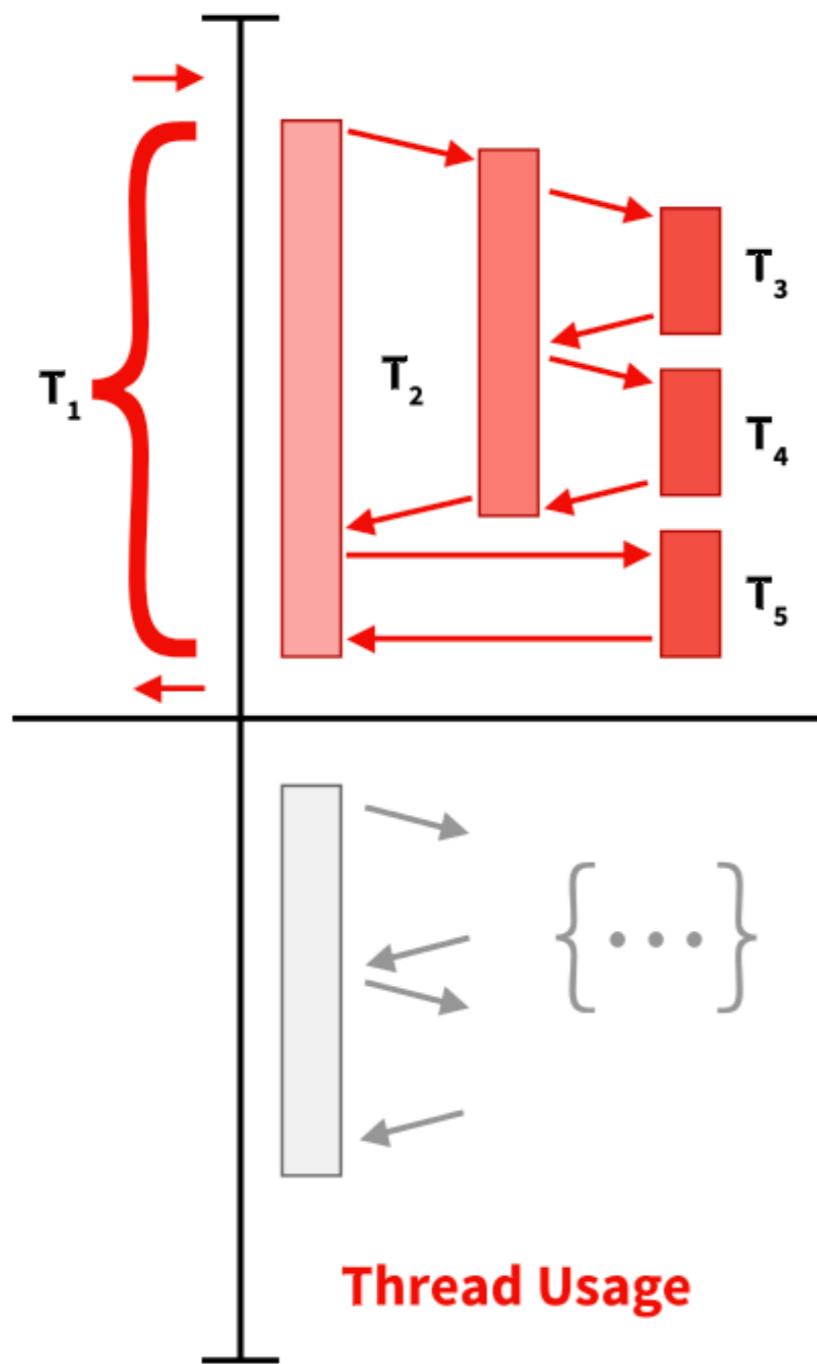
Async data  
processing

Non-blocking

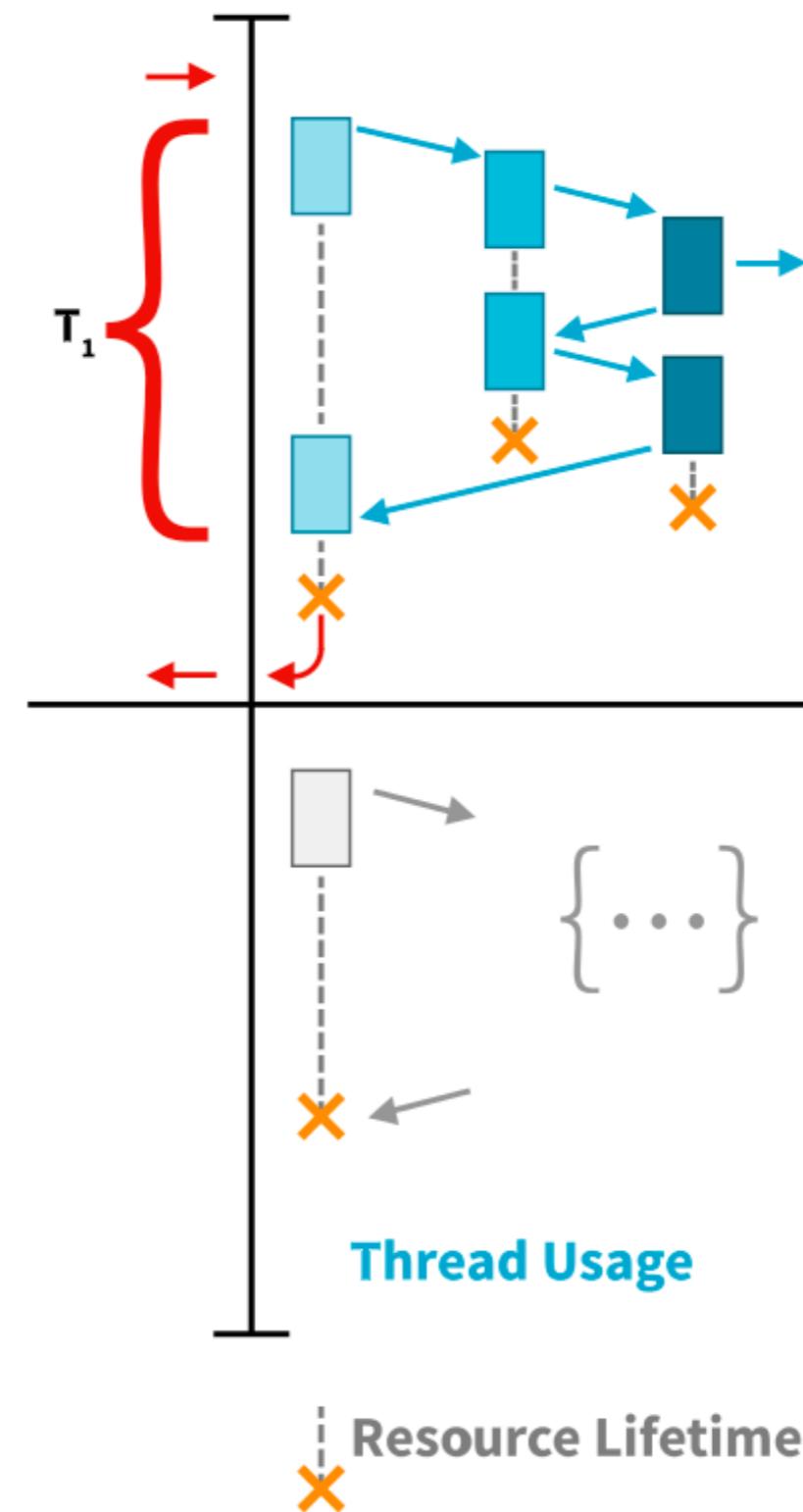
Functional / Declarative  
Programming



## Blocking APIs



## Messaging



Synchronous, blocking communication (left) is resource inefficient and easily bottlenecked. The Reactive approach (right)

reduces risk, conserves valuable resources, and requires less hardware/infrastructure

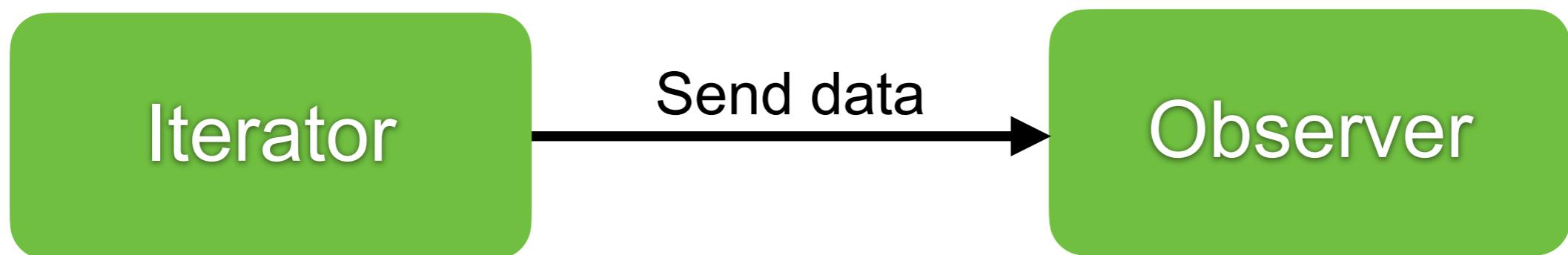
<https://mostafa-asg.github.io/post/reactive-systems-vs-reactive-programming/>



# API for Reactive programming

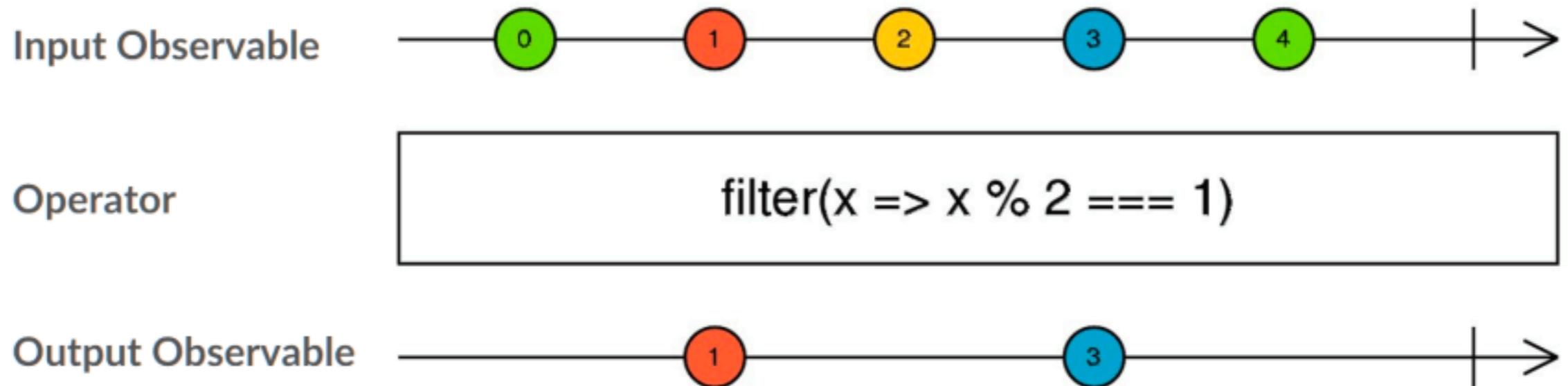
Callback

Declarative with functional composition



# Stream everywhere

Many processes can be modeled as a stream

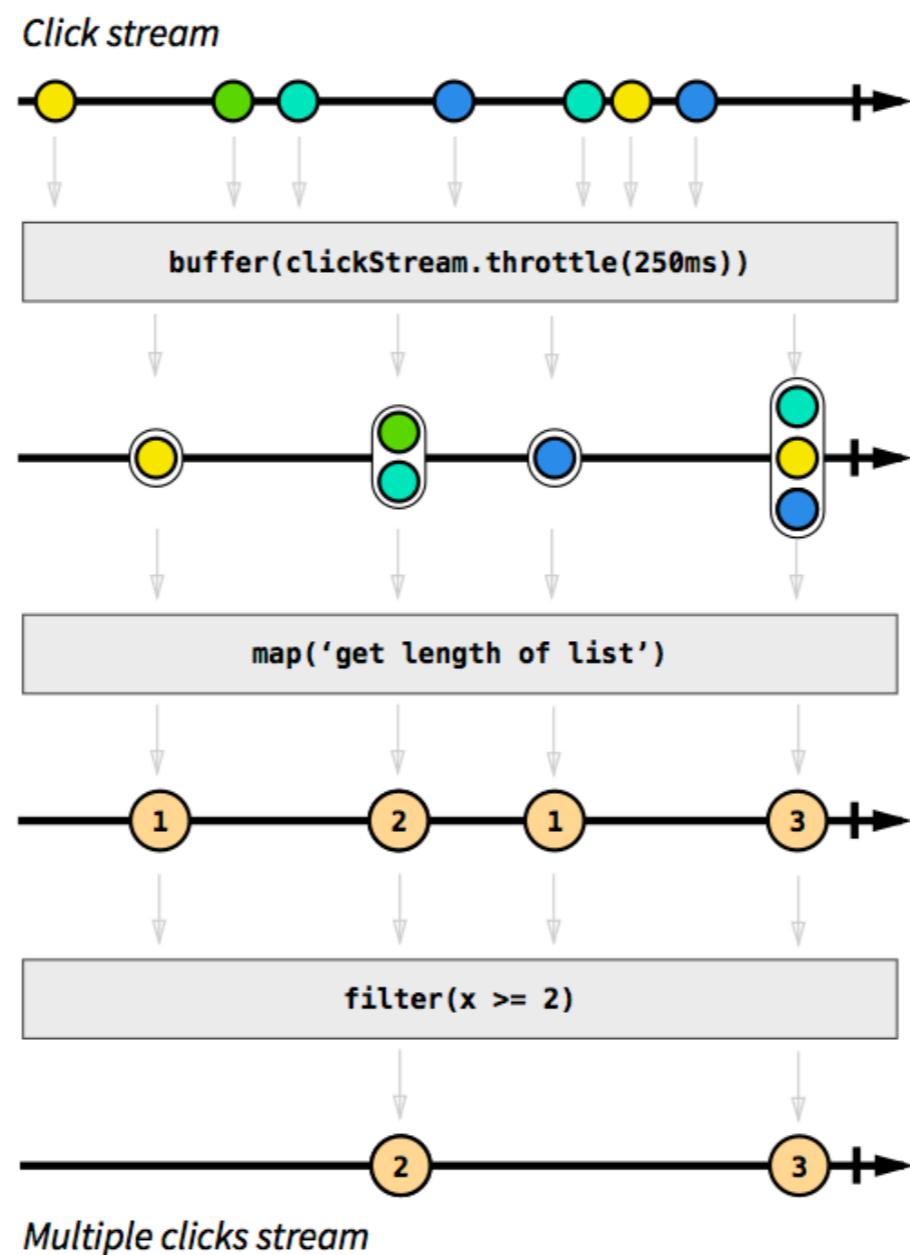


<https://reactive.how/>



# Stream everywhere

Many processes can be modeled as a stream



# JVM Reactive Libraries

RxJava

Vert.X

Akka

Ratpack

Reactor

<http://reactivex.io/languages.html>



# Benefits Reactive programming

Increase utilization of computing resources

Increase performance by reduce serialization point

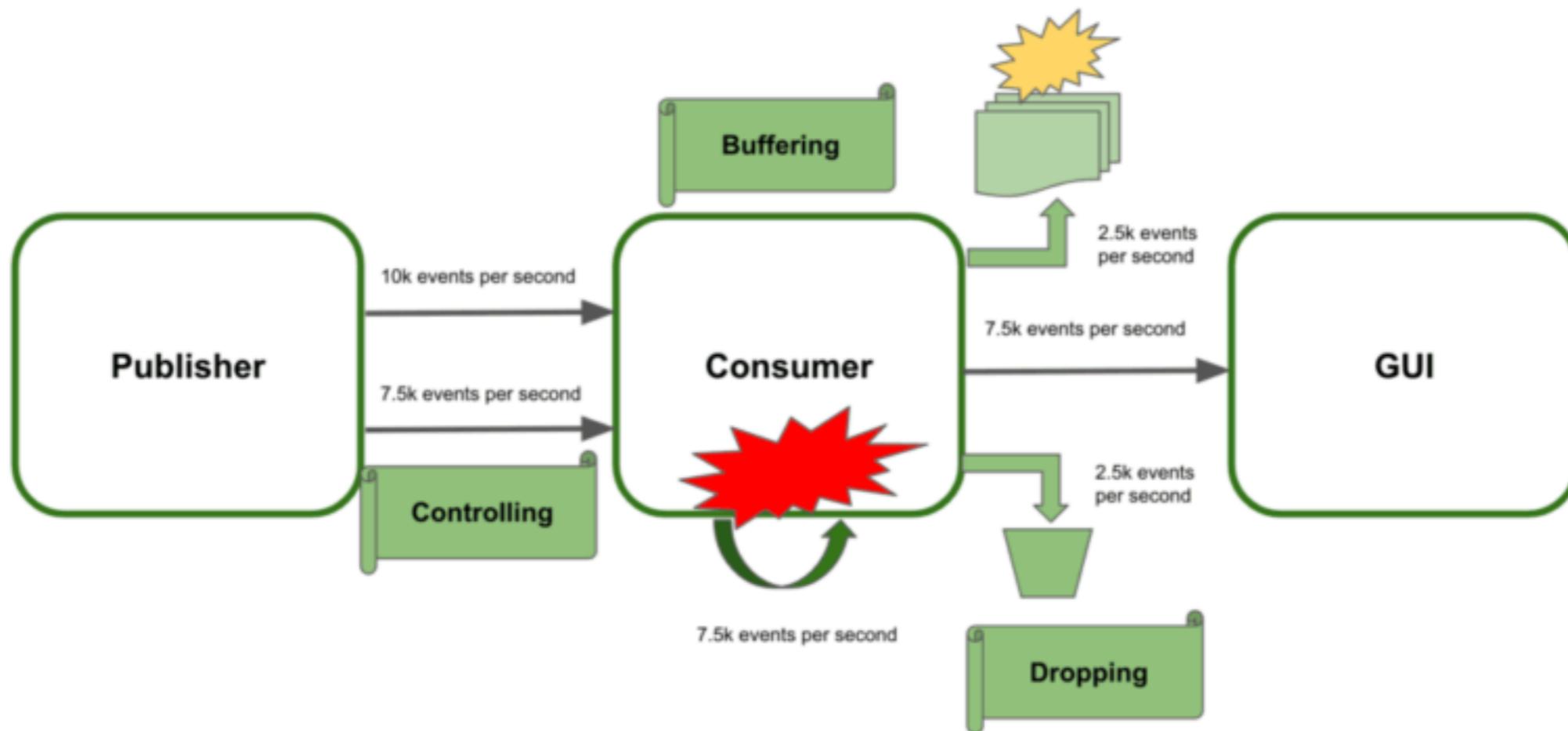
Developer productivity

**Avoid over-utilization (back-pressure)**



# Back-pressure

To prevent systemic failures



<https://www.baeldung.com/spring-webflux-backpressure>



# Reactive programming

**!=**

# Reactive system



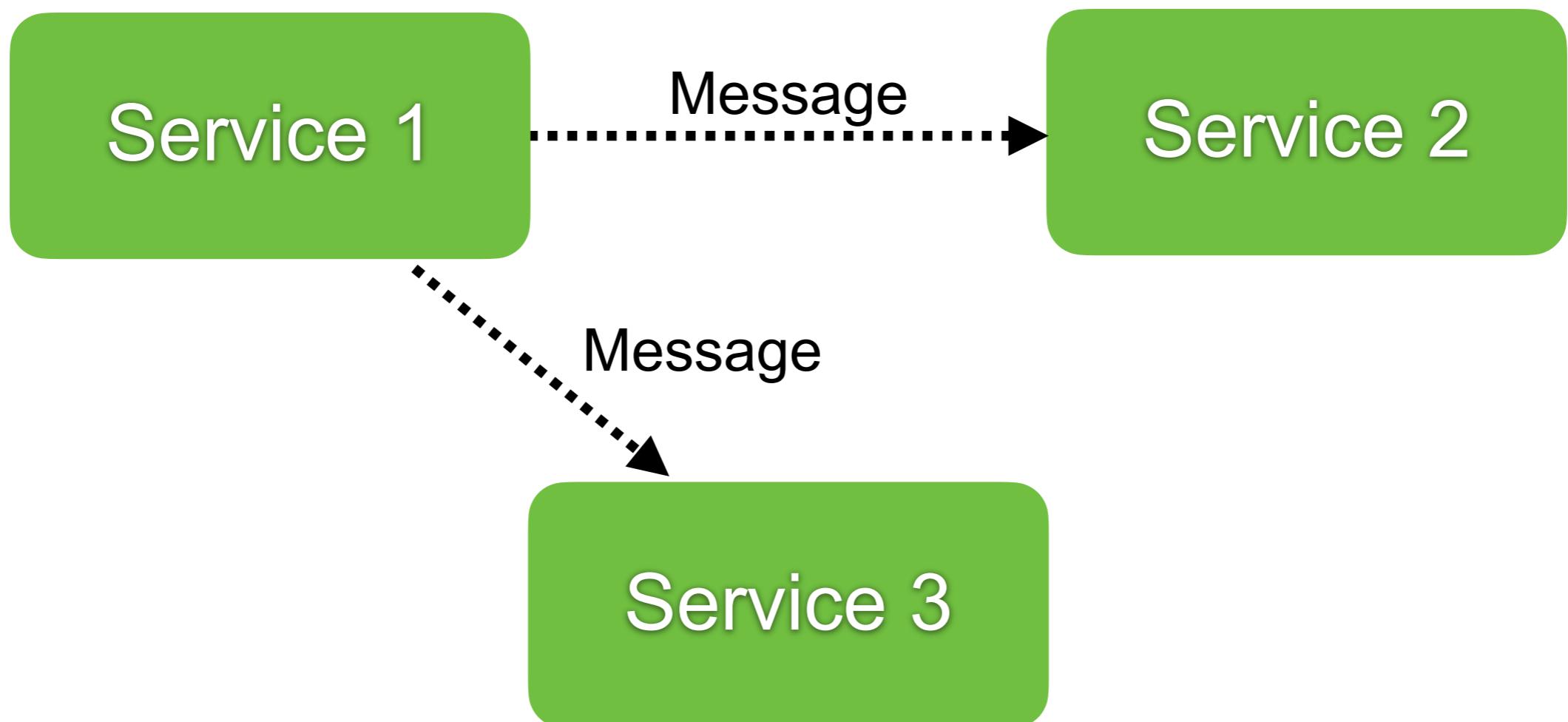
# Reactive programming

Provide a development model with async concept  
Readable and understandable

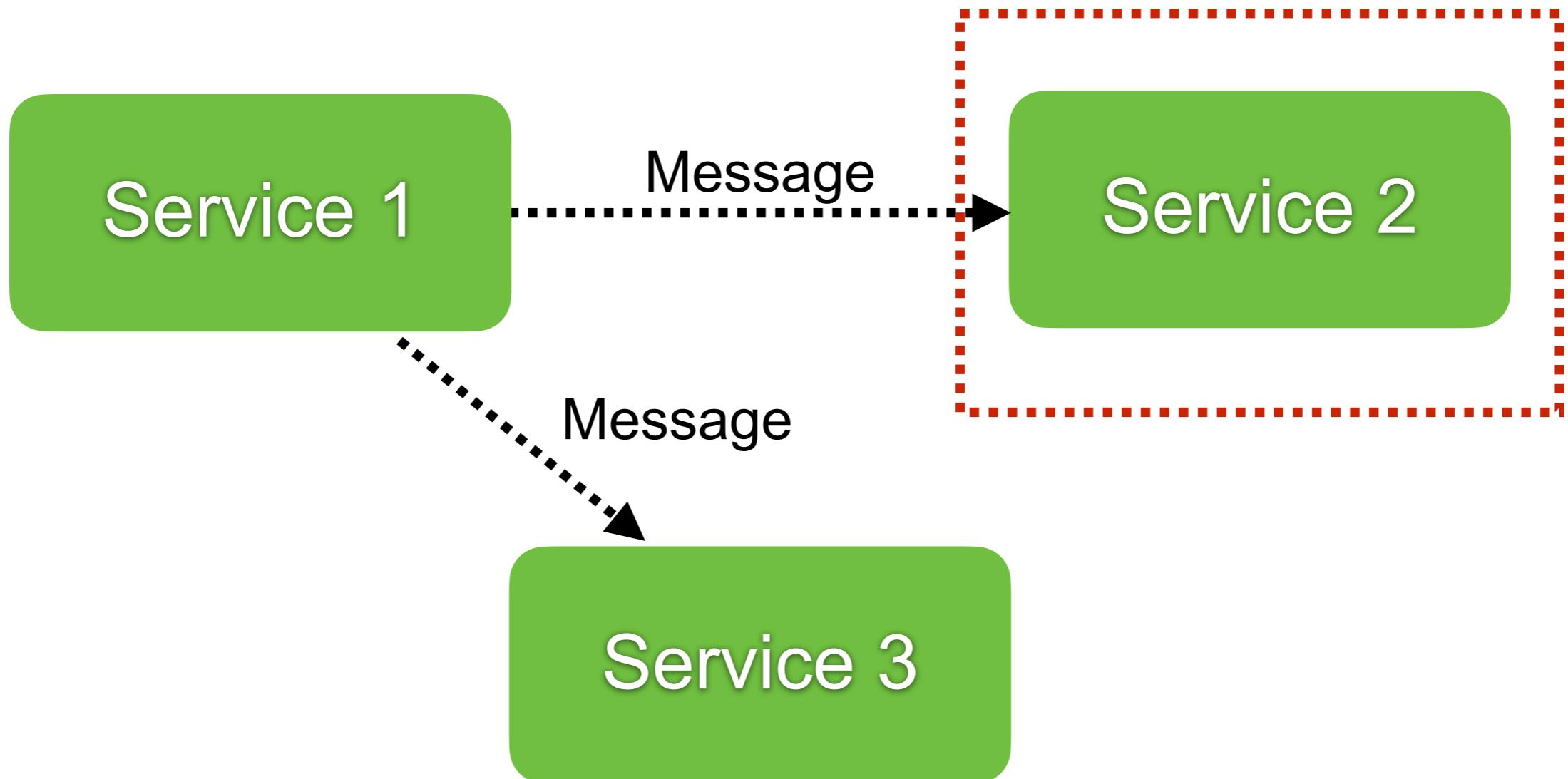
**Not transform your system into a Reactive system**



# Reactive system

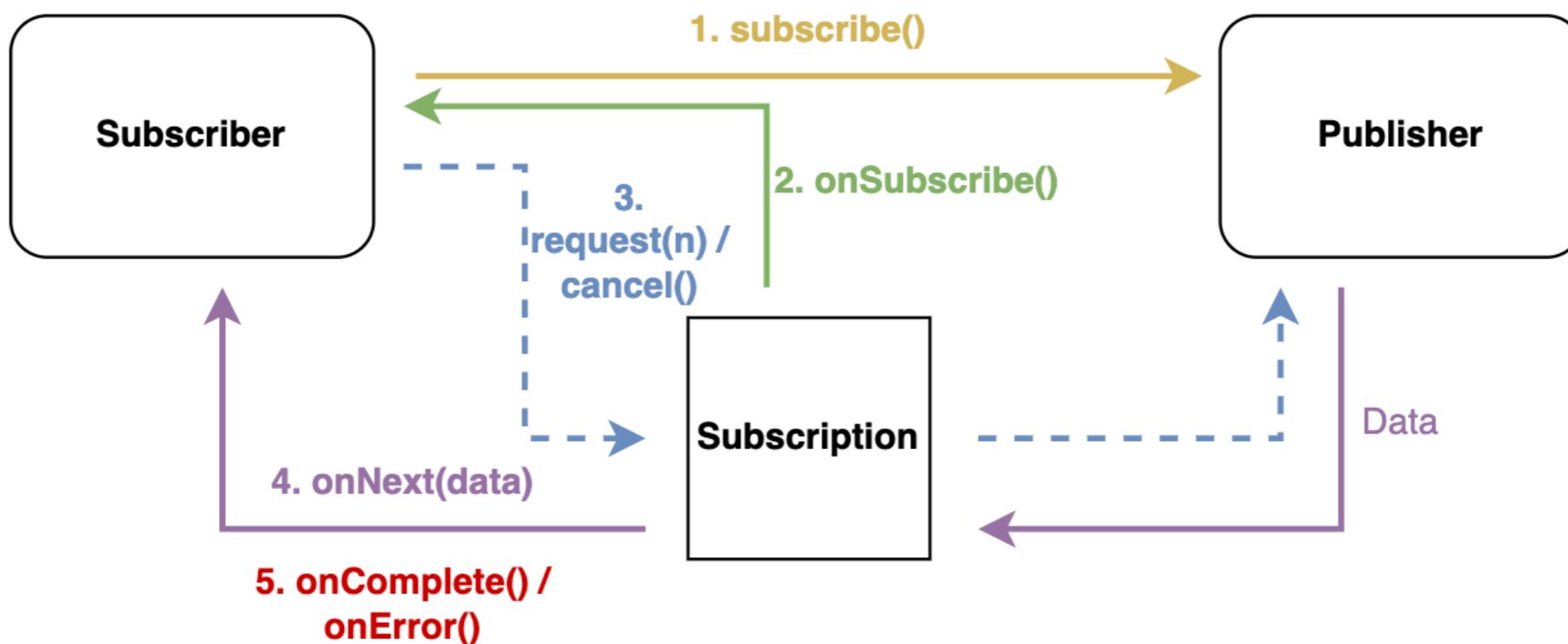


# Reactive programming



# Publisher/Subscriber

Communication in same process



# Publisher

Source  
Observable  
Upstream  
Producer

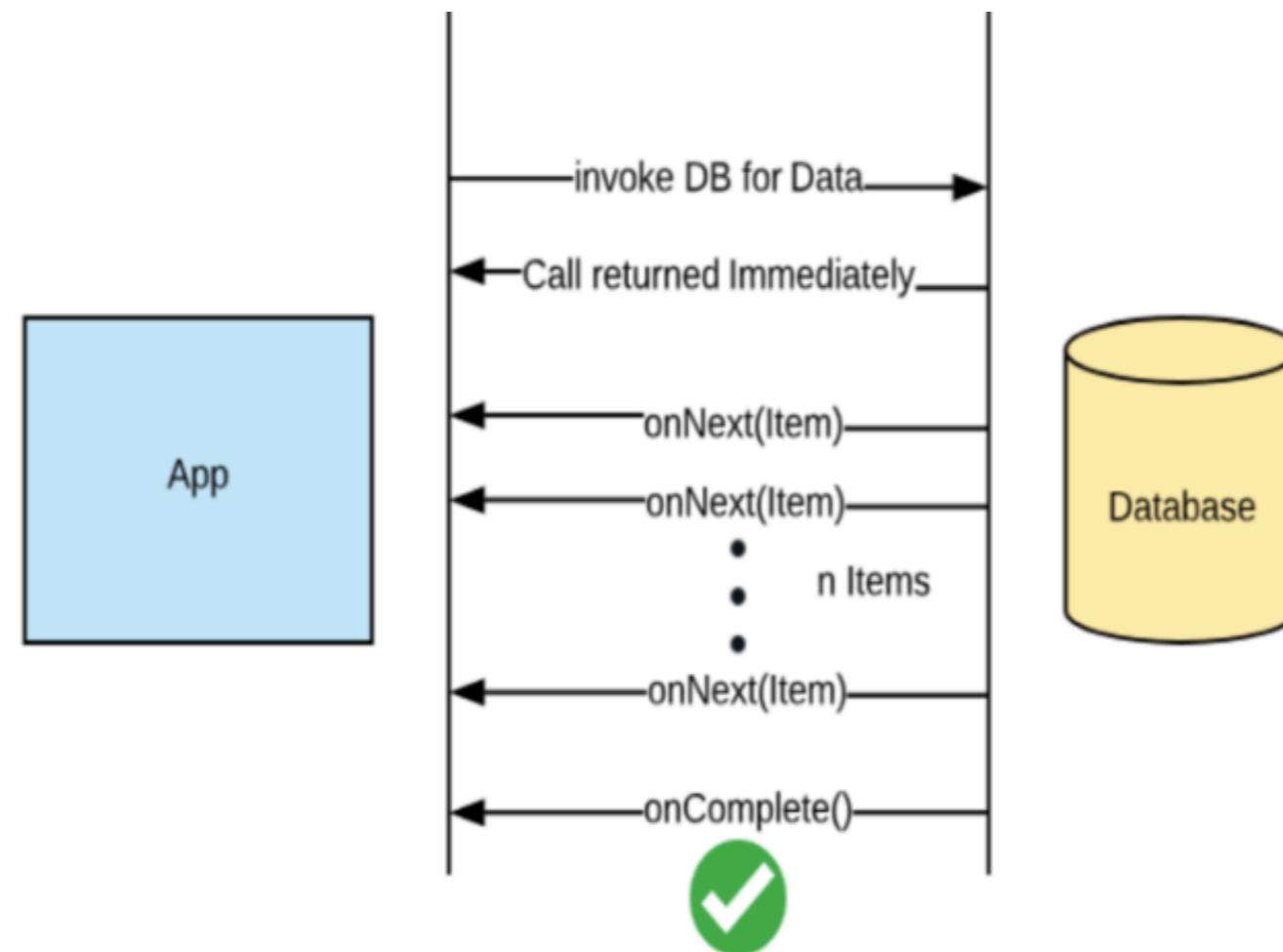


# Subscriber

Sink  
Observer  
Downstream  
Consumer

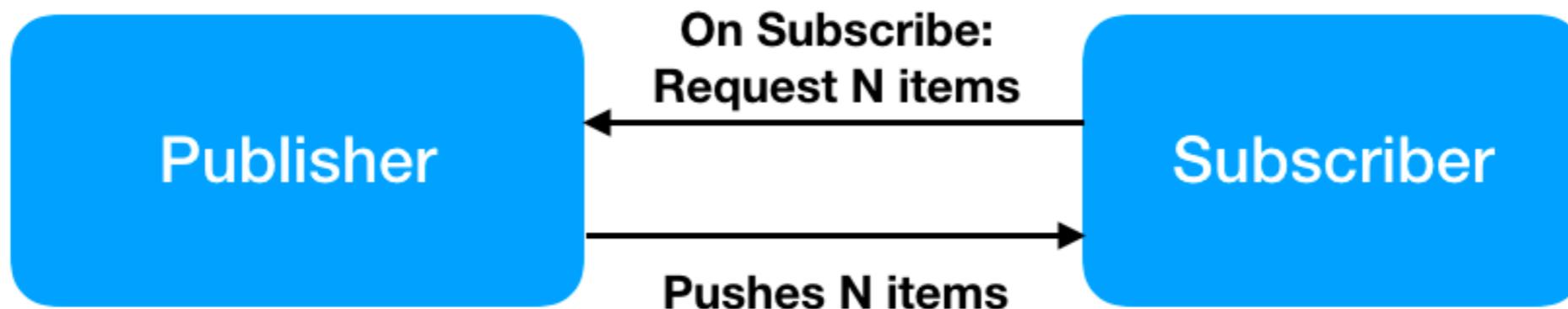


# Demo with Java 9



# Demo with Java 9

Working with `java.util.concurrent.Flow`



<https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/Flow.html>

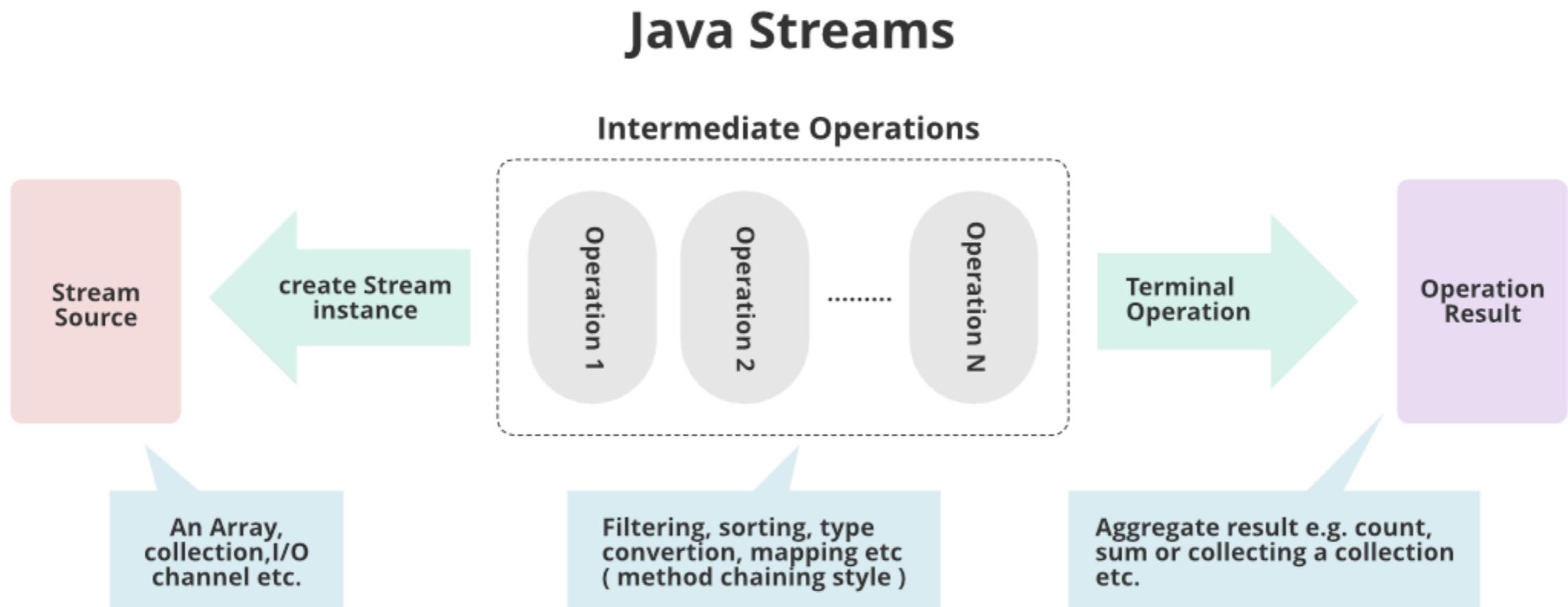


# Demo with Java 9

<https://github.com/up1/workshop-reactive-programming/tree/main/workshop/example01>



# Working with Stream



<https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html>



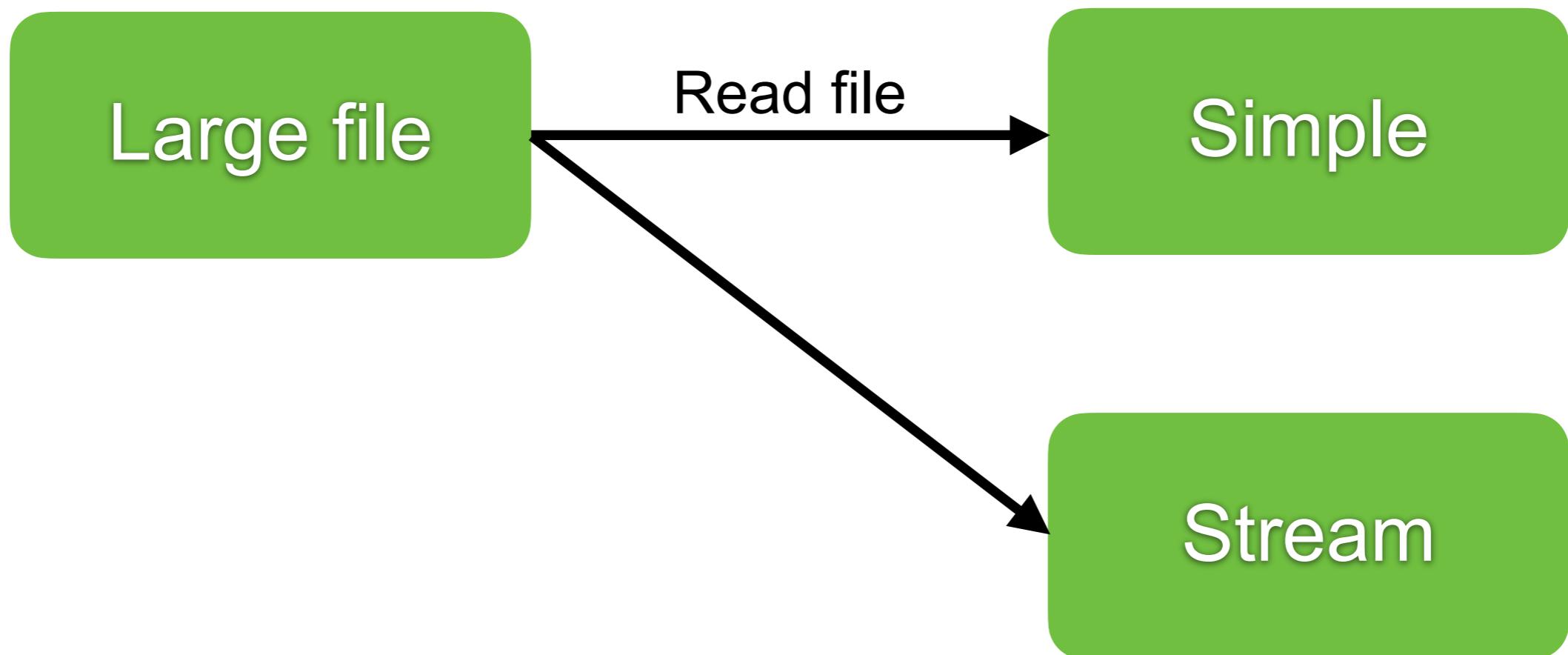
# Working with Stream

```
List<String> datas = Arrays.asList("A1", "A2", "B1", "B2", "C1");
// Imperative
for (String data : datas) {
    if(data.startsWith("A")) {
        out.println(data);
    }
}
// Declarative with Stream API
datas.stream()
    .filter( data -> data.startsWith("A") )
    .forEach(out::println);
```

<https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html>



# How to read a large file ?



# Spring Reactive

<https://spring.io/reactive>



# Spring Reactive



Spring **Boot 2**

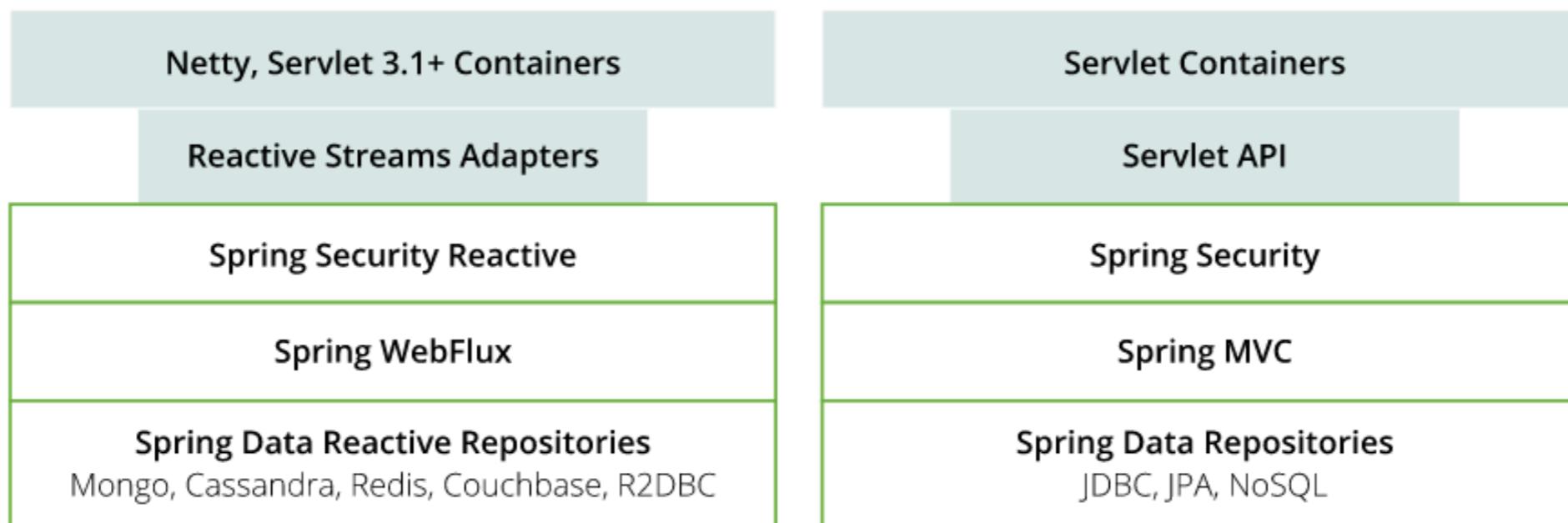


**Reactor**

Optional Dependency

## Reactive Stack

Spring WebFlux is a non-blocking web framework built from the ground up to take advantage of multi-core, next-generation processors and handle massive numbers of concurrent connections.



## Servlet Stack

Spring MVC is built on the Servlet API and uses a synchronous blocking I/O architecture with a one-request-per-thread model.

<https://spring.io/reactive>



Reactive Programming

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

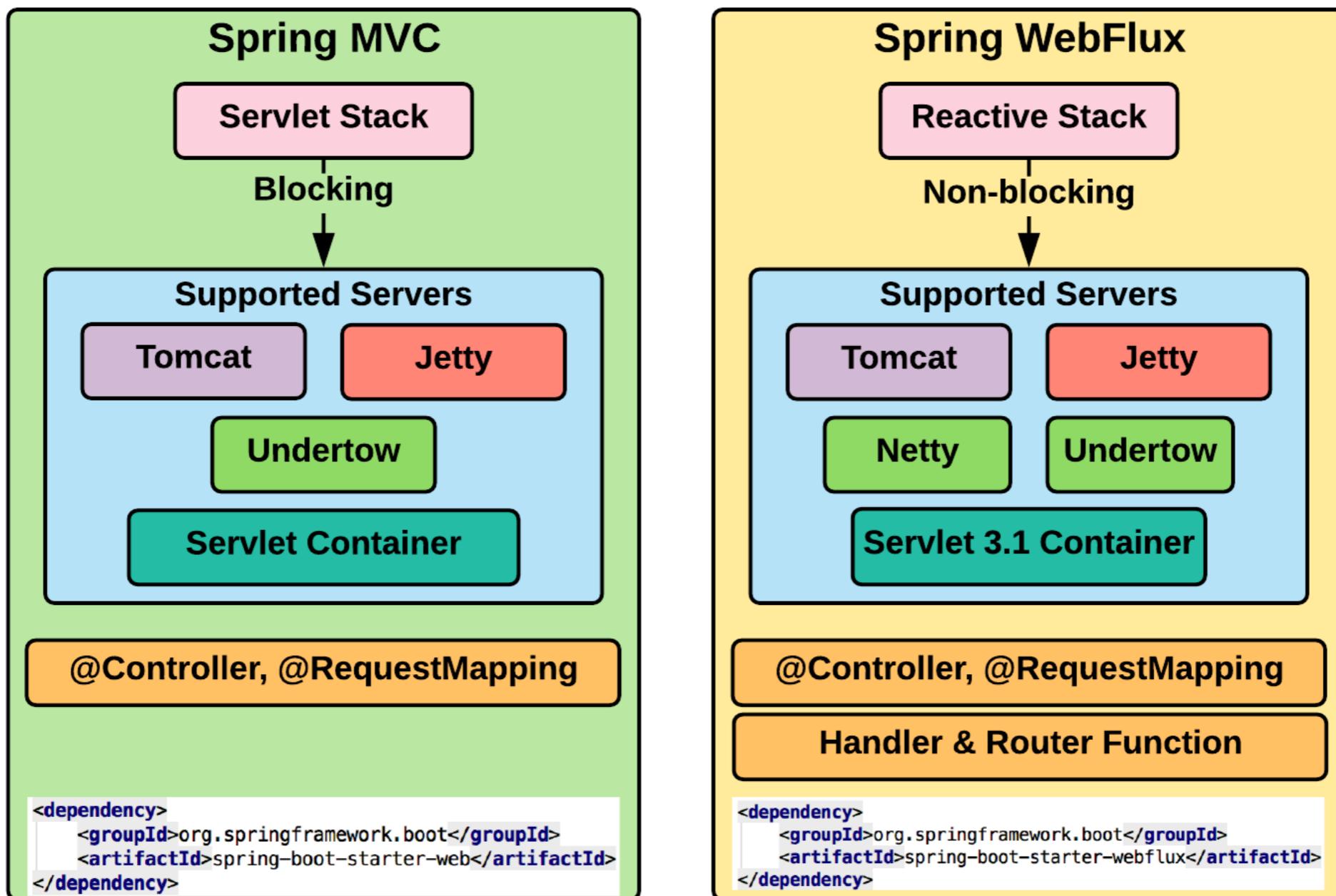
# **Spring MVC Async**

# **Spring WebFlux**

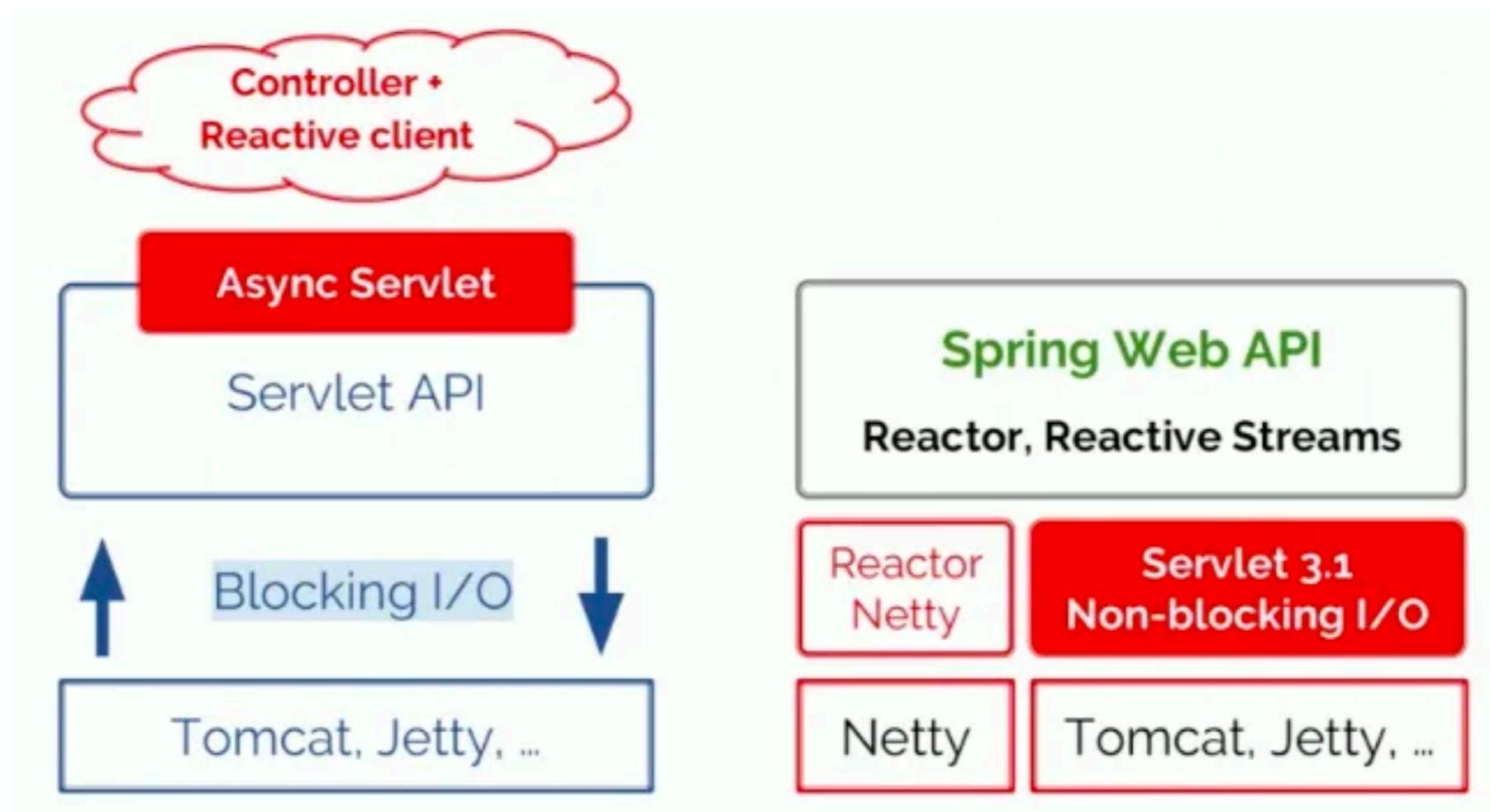
<https://spring.io/reactive>



# Compare ...



# Compare ...



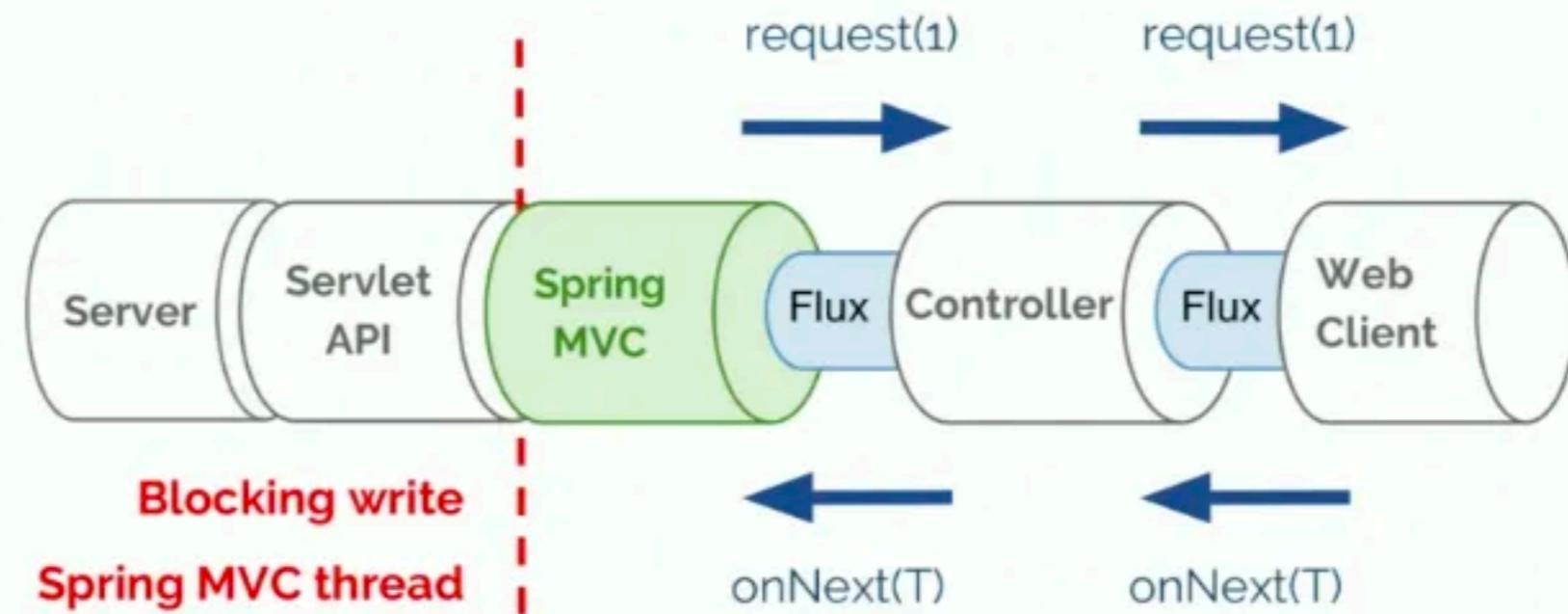
# Compare ...

Spring MVC Async	Spring WebFlux
Servlet 3.0	Servlet 3.1+
Communication with the client is blocking	Non-blocking
Read the request body is blocking	Non-blocking
Filters and Servlets are working sync	Async
-	Support back-pressure
Imperative style	Declarative and functional style

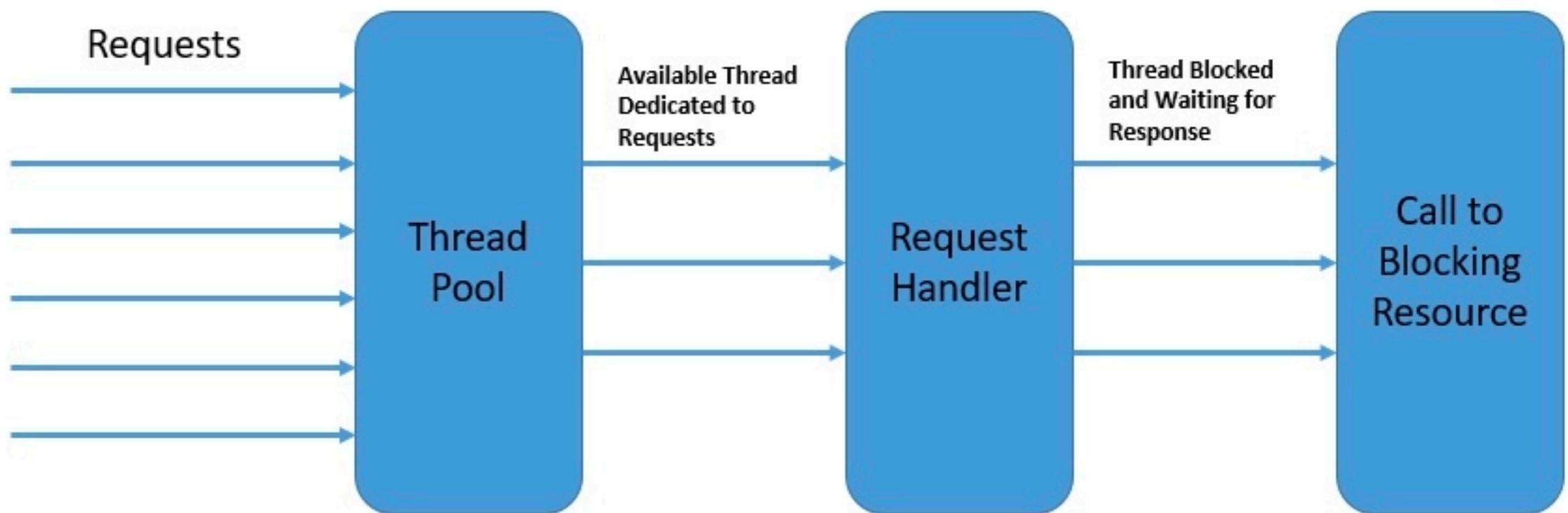


# Spring MVC

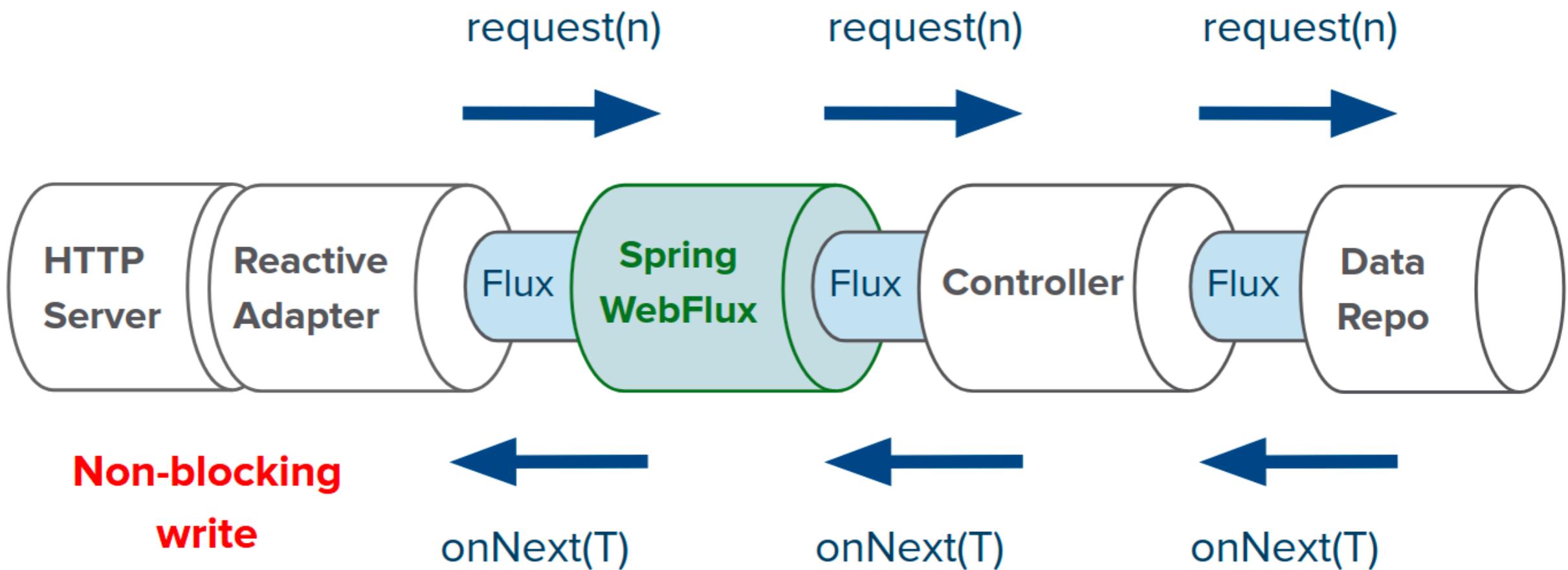
## Spring MVC Response



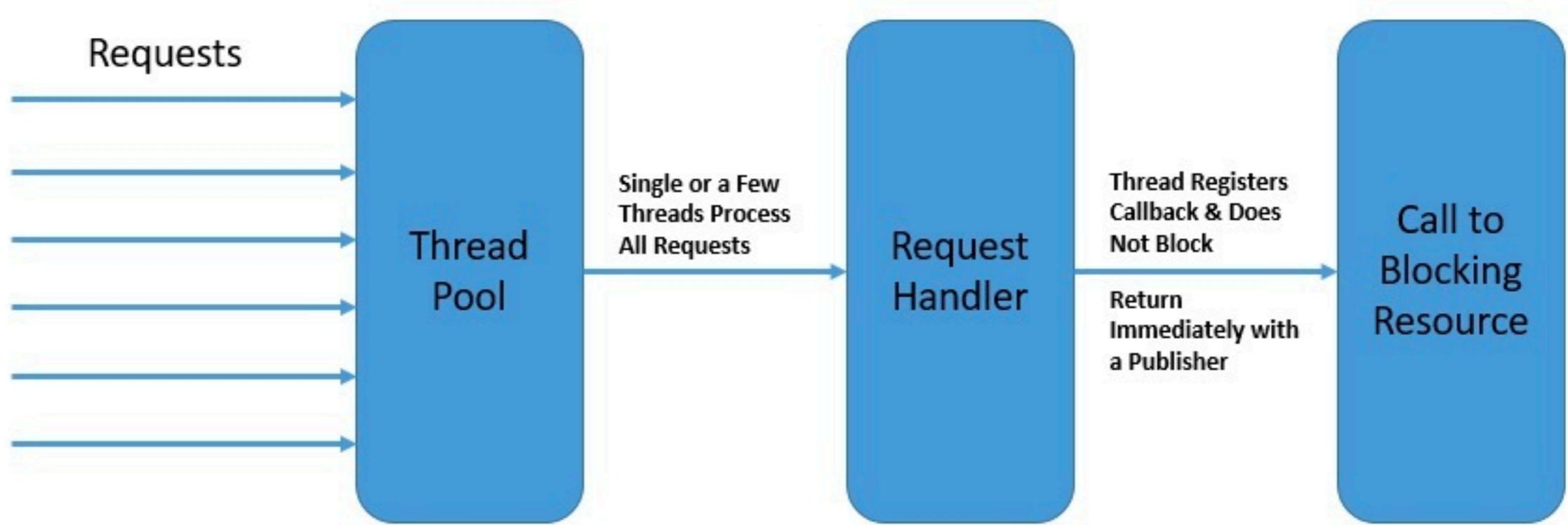
# Spring MVC



# Spring WebFlux

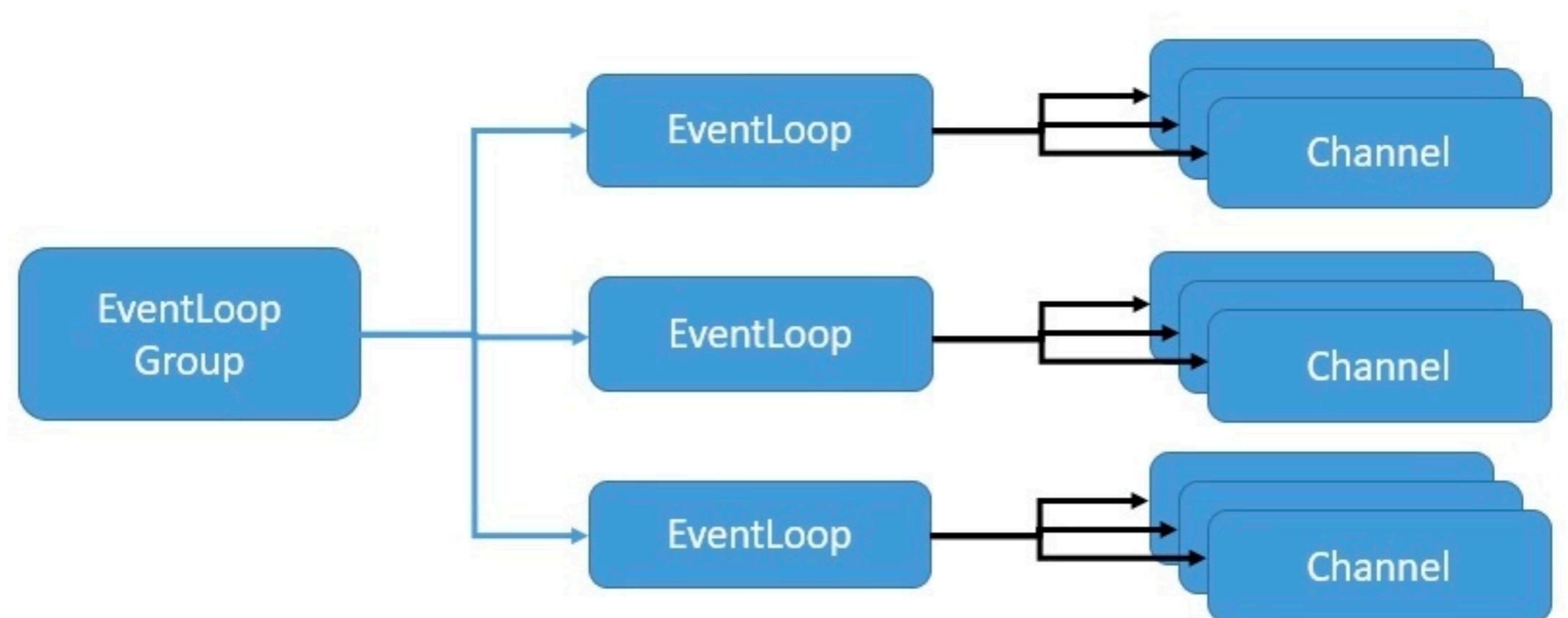


# Spring WebFlux



# Using Netty

Use event loop model to provide highly scalable concurrency



<https://netty.io/>



# **Choose right answer to your question/problem**



# Spring WebFlux

Reactive programming support for web app  
Use project **Reactor**  
Publisher implementations

**Mono** = 0 or 1 item

**Flux** = 0 or N items (infinity)

<https://projectreactor.io/>



# Flux vs Mono

## Mono

Handling zero(optional) or one result

## Flux

Handling zero(optional) or many results (infinite)

<https://projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html>  
<https://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html>



Lirrary	Types
RxJS	Observable
Spring WebFlux	Mono or Flux
Java 9+	Flow

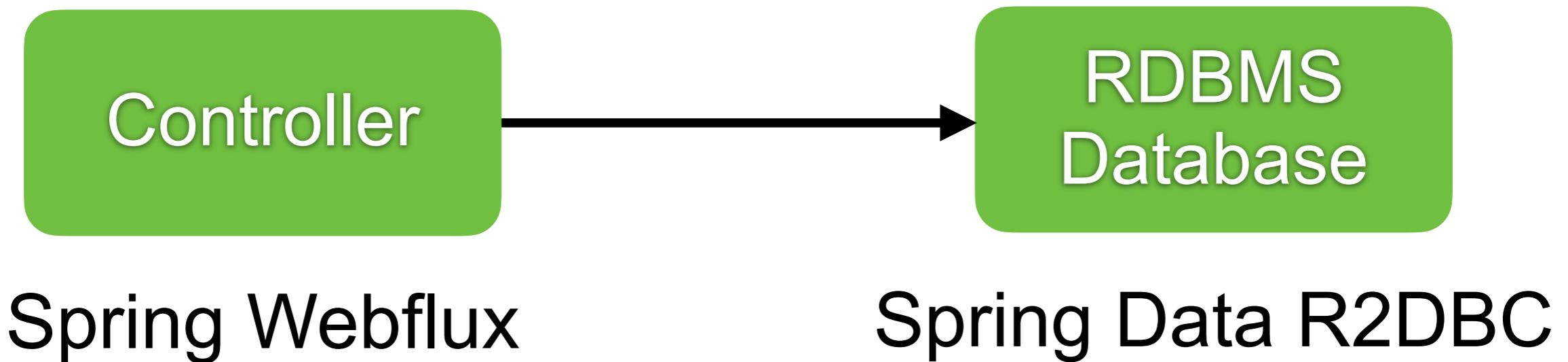
<https://www.reactive-streams.org/>



# Demo



# Spring WebFlux



<https://spring.io/projects/spring-data-r2dbc>



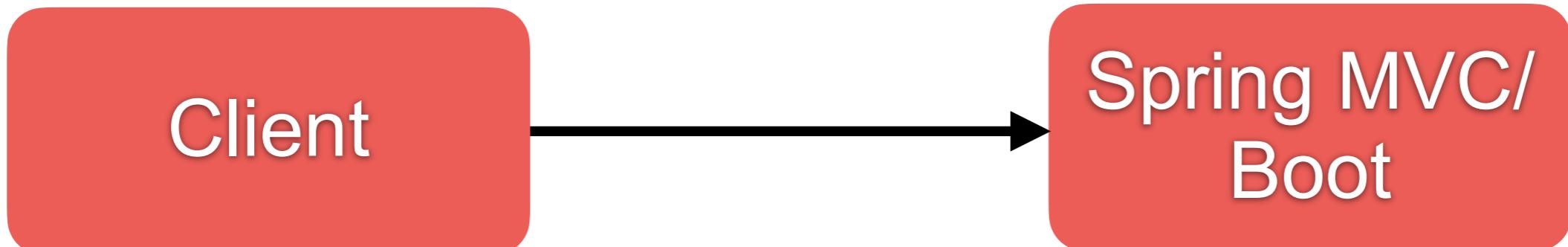
# Reactive Relational Database Connectivity

H2  
MySQL, MariaDB  
PostgreSQL  
SQL Server  
Oracle

<https://spring.io/projects/spring-data-r2dbc>



# Spring WebFlux



RestTemplate



WebClient

<https://spring.io/projects/spring-data-r2dbc>



# Demo

# Reactive REST application



# Create Project



**Project**  
 Maven Project  Gradle Project

**Language**  
 Java  Kotlin  Groovy

**Spring Boot**  
 2.6.2 (SNAPSHOT)  2.6.1  2.5.8 (SNAPSHOT)  2.5.7

**Project Metadata**

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	<input checked="" type="radio"/> Jar <input type="radio"/> War
Java	<input type="radio"/> 17 <input type="radio"/> 11 <input checked="" type="radio"/> 8

## Dependencies

[ADD DEPENDENCIES... ⌘ + B](#)

### Spring Reactive Web WEB

Build reactive web applications with Spring WebFlux and Netty.

### Spring Data R2DBC SQL

Provides Reactive Relational Database Connectivity to persist data in SQL stores using Spring Data in reactive applications.

### H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

### Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

<https://start.spring.io/>



# Flux and Mono

## Reactive REST

```
@GetMapping("/{id}")
public Mono<User> getUserById(@PathVariable String id) {
    return this.userRepository.findUserById(id);
}

@GetMapping
private Flux<User> getAllEmployees() {
    return this.userRepository.findAllUsers();
}
```



# Testing



# Testing

SpringBootTest + WebTestClient  
WebFluxTest for Controller

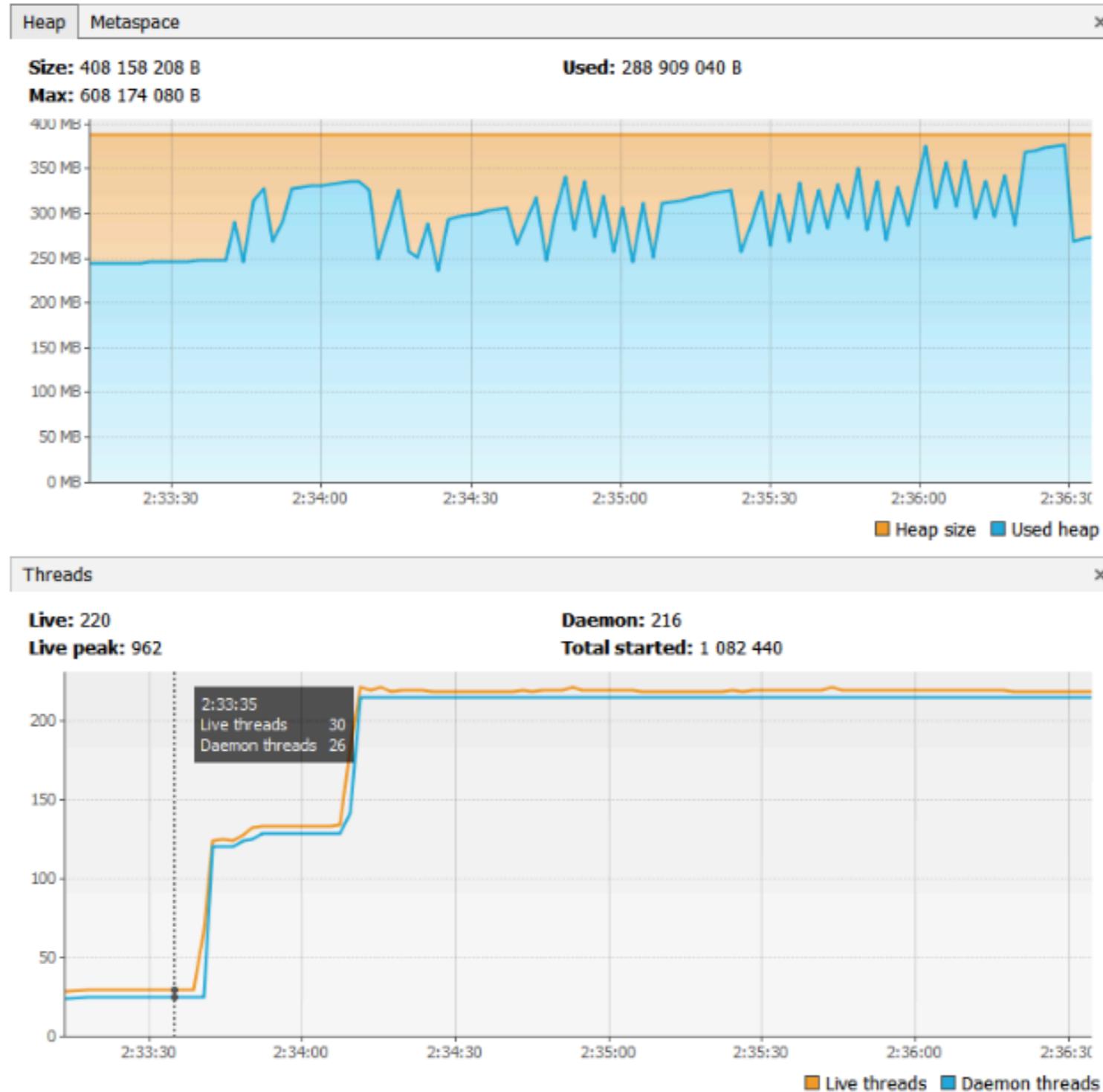


# **Performance testing**

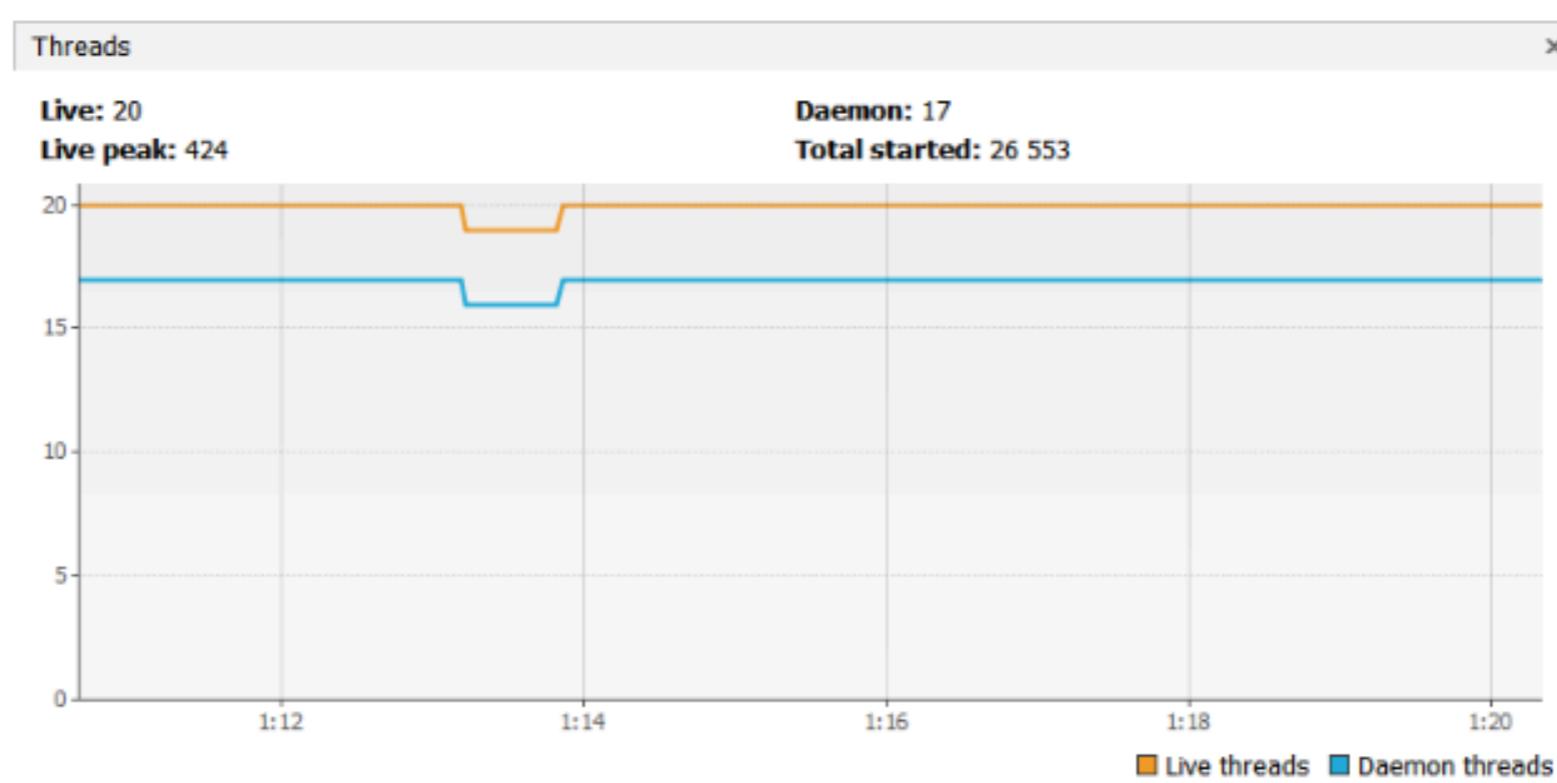
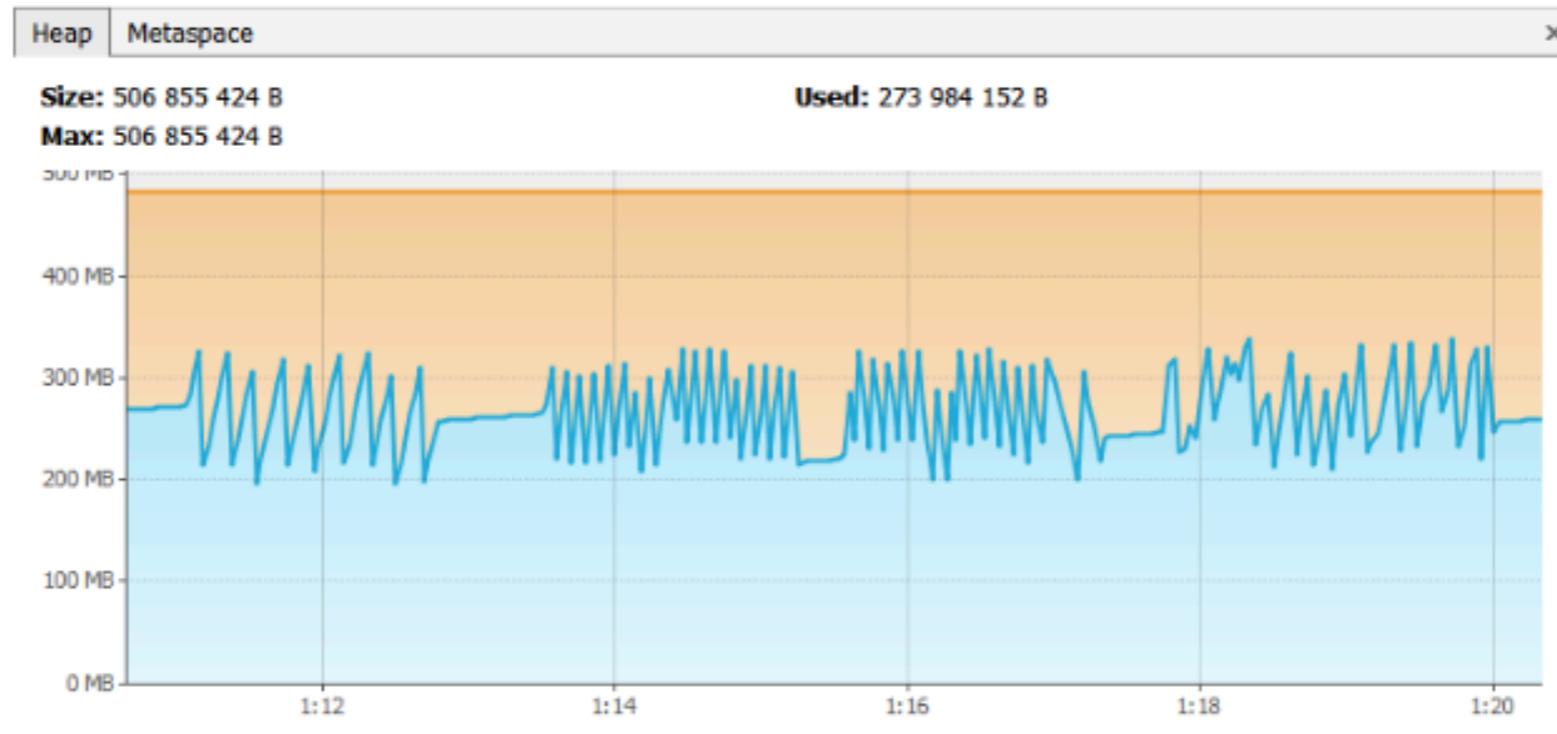
## **Profiling**



# Simple REST (default = 200)



# WebFlux



# REST vs WebFlux

Use delay = 3 seconds

▶ STATISTICS														<a href="#">Expand all groups</a>   <a href="#">Collapse all groups</a>	
Requests ▲	🕒 Executions					⌚ Response Time (ms)									
	Total ▲	OK ▲	KO ▲	% KO ▲	Cnt/s ▲	Min ▲	50th pct ▲	75th pct ▲	95th pct ▲	99th pct ▲	Max ▲	Mean ▲	Std Dev ▲		
Global Information	600	451	149	25%	4.348	5	20111	44147	60001	60003	60005	28864	15541		
rest	300	152	148	49%	2.174	3088	20113	45139	60001	60004	60005	30158	17380		
webflux	300	299	1	0%	2.174	5	19102	44147	44159	54132	54156	27570	13327		

<https://gatling.io/>

