Game of Streams 

How to tame & get the most from your messaging platforms

Mark Heckler
Professional Problem Solver, Spring Developer & Advocate
www.thehecklers.com
mark@thehecklers.com
mheckler@vmware.com
@mkheck

vmware®

“Please do *LESS* with *MORE*!” 💰💰💰

Who am I?

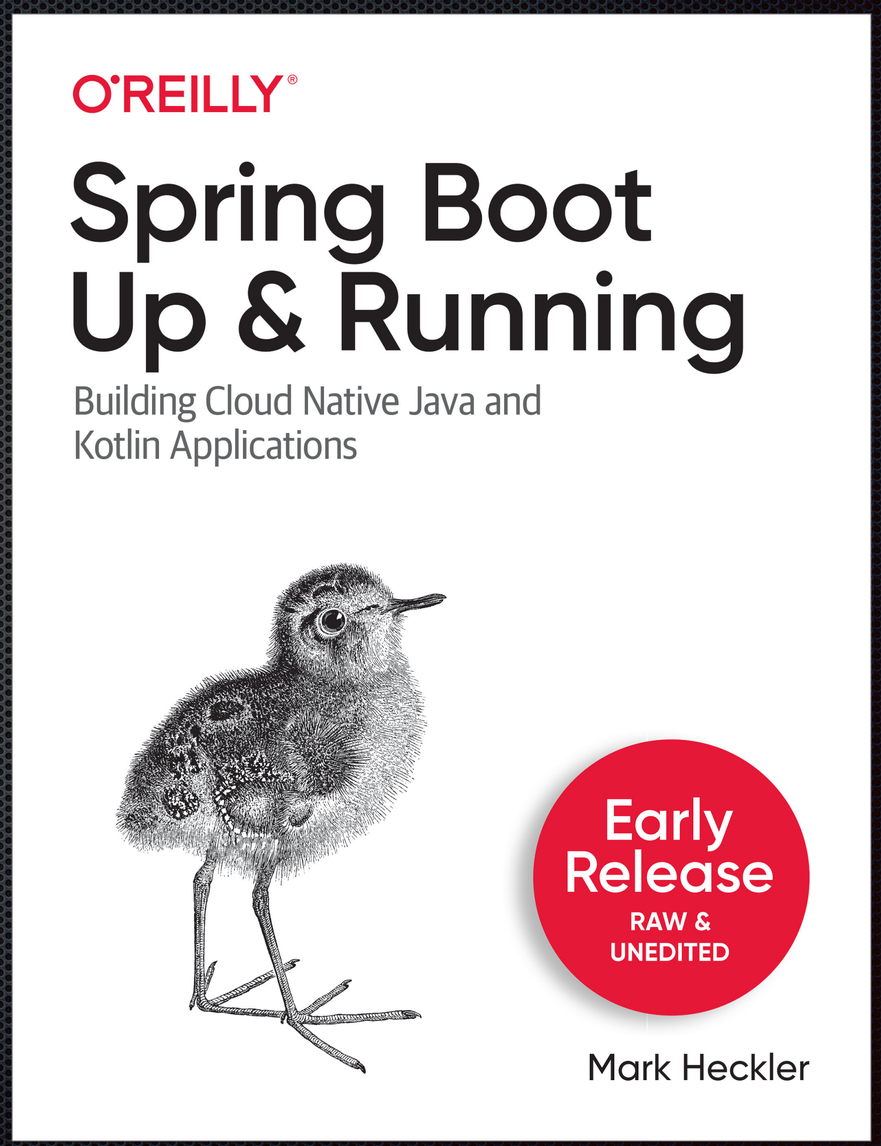
- Author
- Architect & Developer
- Java Champion, Rockstar
- Professional Problem Solver
- Spring Developer & Advocate
- Creador y curador de

SPRING NOTICIAS EN ESPAÑOL

New book!

Now available for early access

https://bit.ly/springbootbook



Takeaways

- **Why use messaging platforms/where do they fit in a distributed architecture?**
- Examples of leading messaging platforms
- What is Spring Cloud Stream?
- Why use it?

Takeaways

- Why use messaging platforms/where do they fit in a distributed architecture?
- **Examples of leading messaging platforms**
- What is Spring Cloud Stream?
- Why use it?

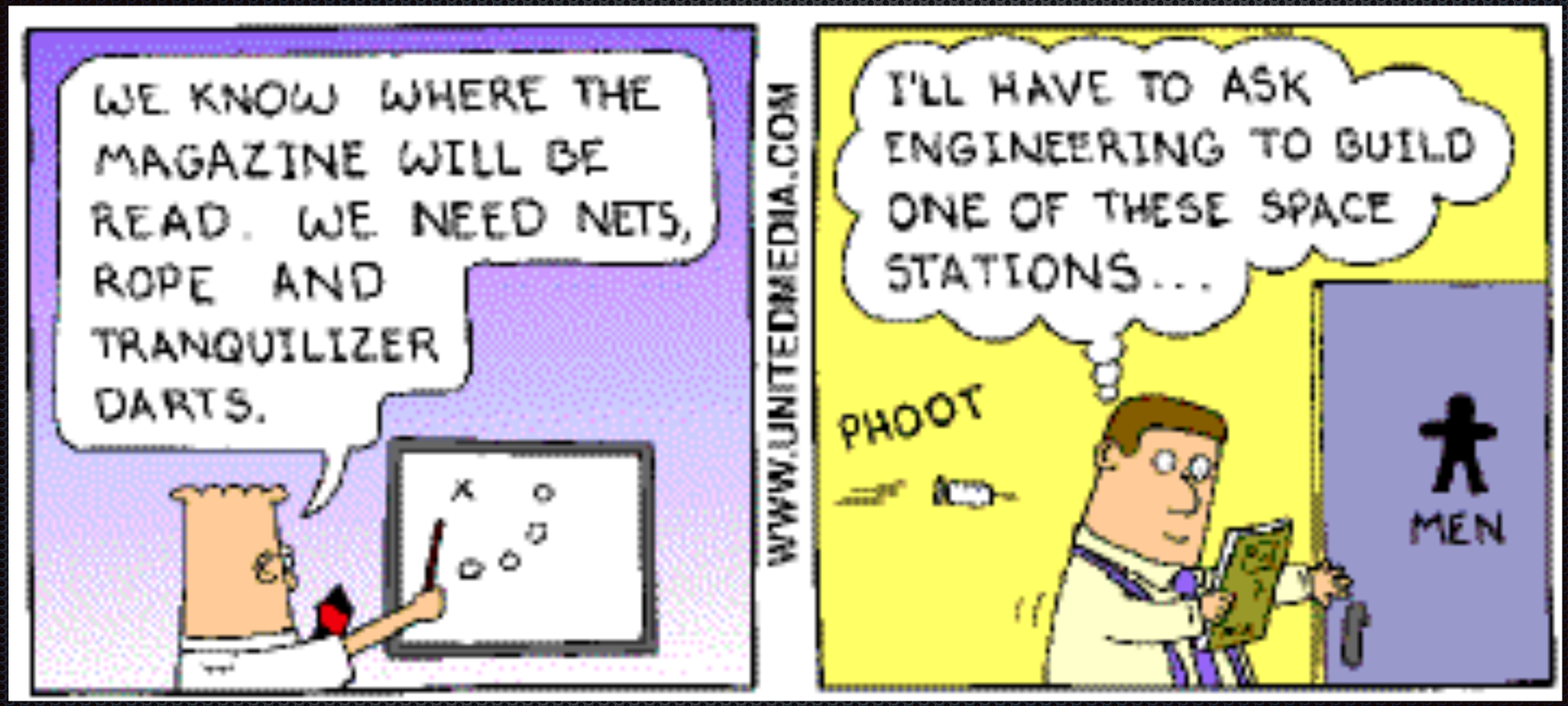
Takeaways

- Why use messaging platforms/where do they fit in a distributed architecture?
- Examples of leading messaging platforms
- **What is Spring Cloud Stream?**
- Why use it?

Takeaways

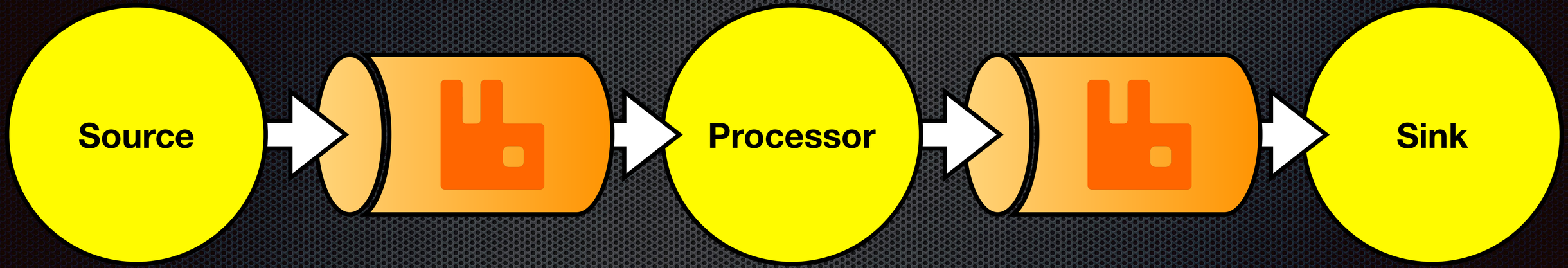
- Why use messaging platforms/where do they fit in a distributed architecture?
- Examples of leading messaging platforms
- What is Spring Cloud Stream?
- **Why use it?**

Why use it?

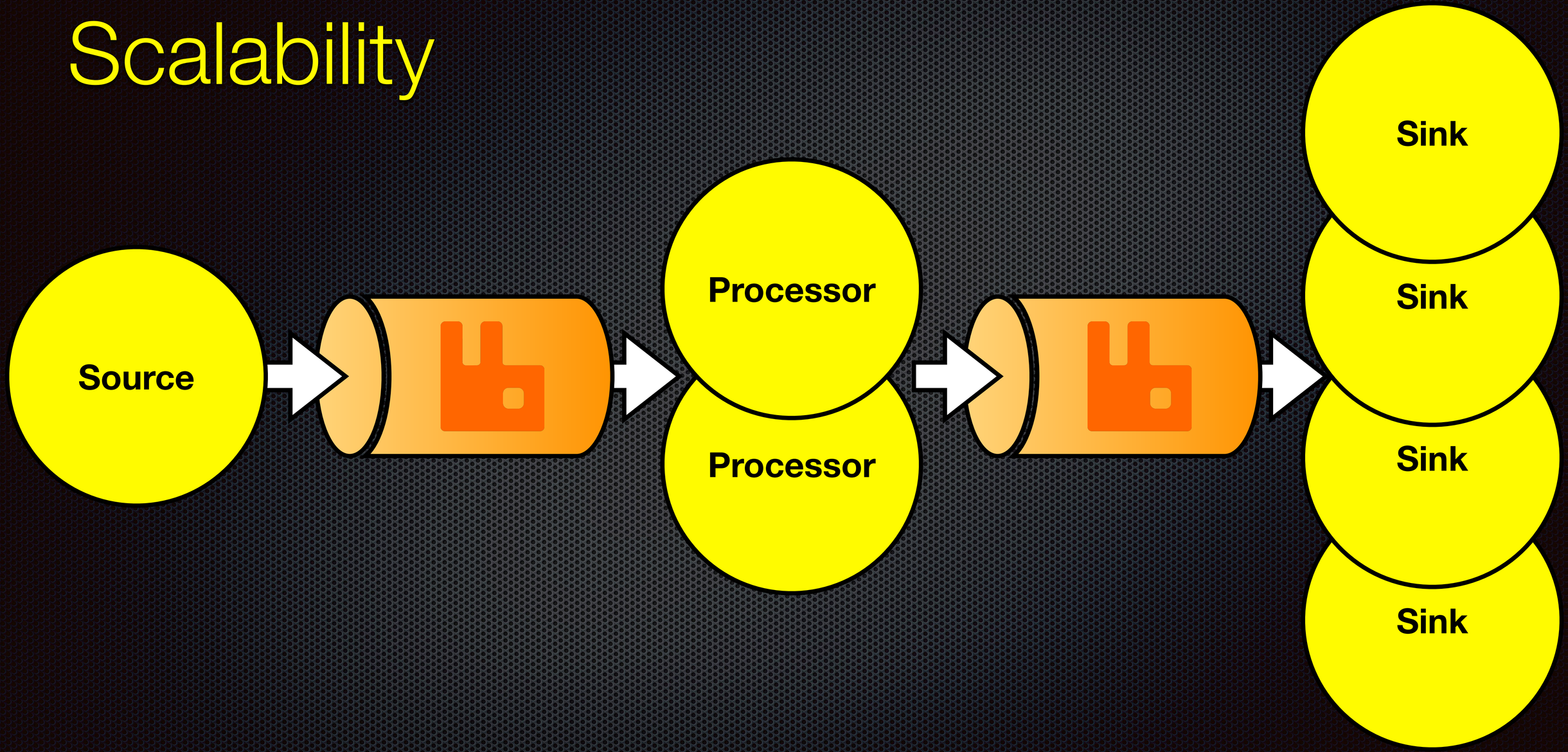

WE KNOW WHERE THE MAGAZINE WILL BE READ. WE NEED NETS, ROPE AND TRANQUILIZER DARTS.
WWW.UNITEDMEDIA.COM
I'LL HAVE TO ASK ENGINEERING TO BUILD ONE OF THESE SPACE STATIONS...
PHOOT
MEN

SCSt + ${messaging.platform} =

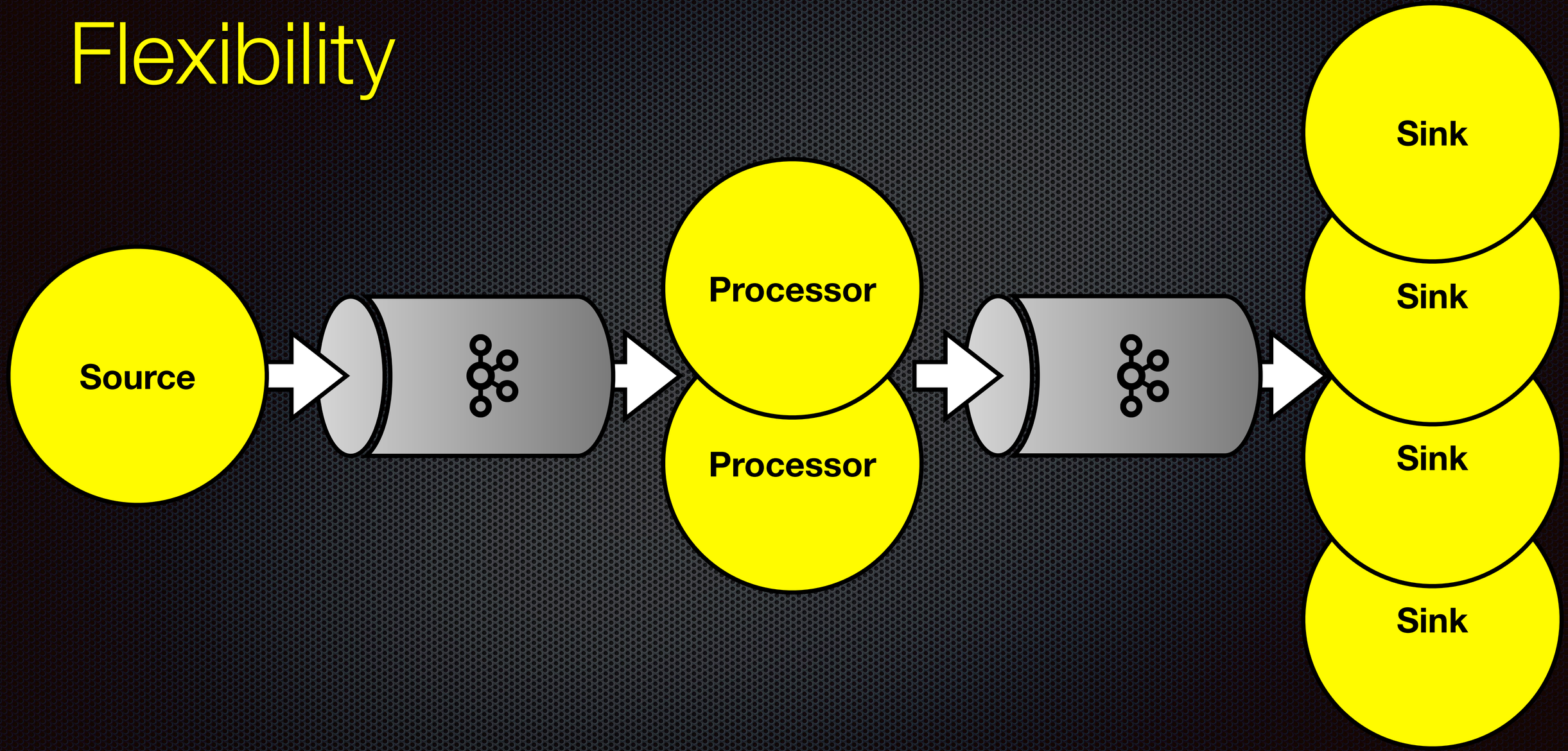
Power



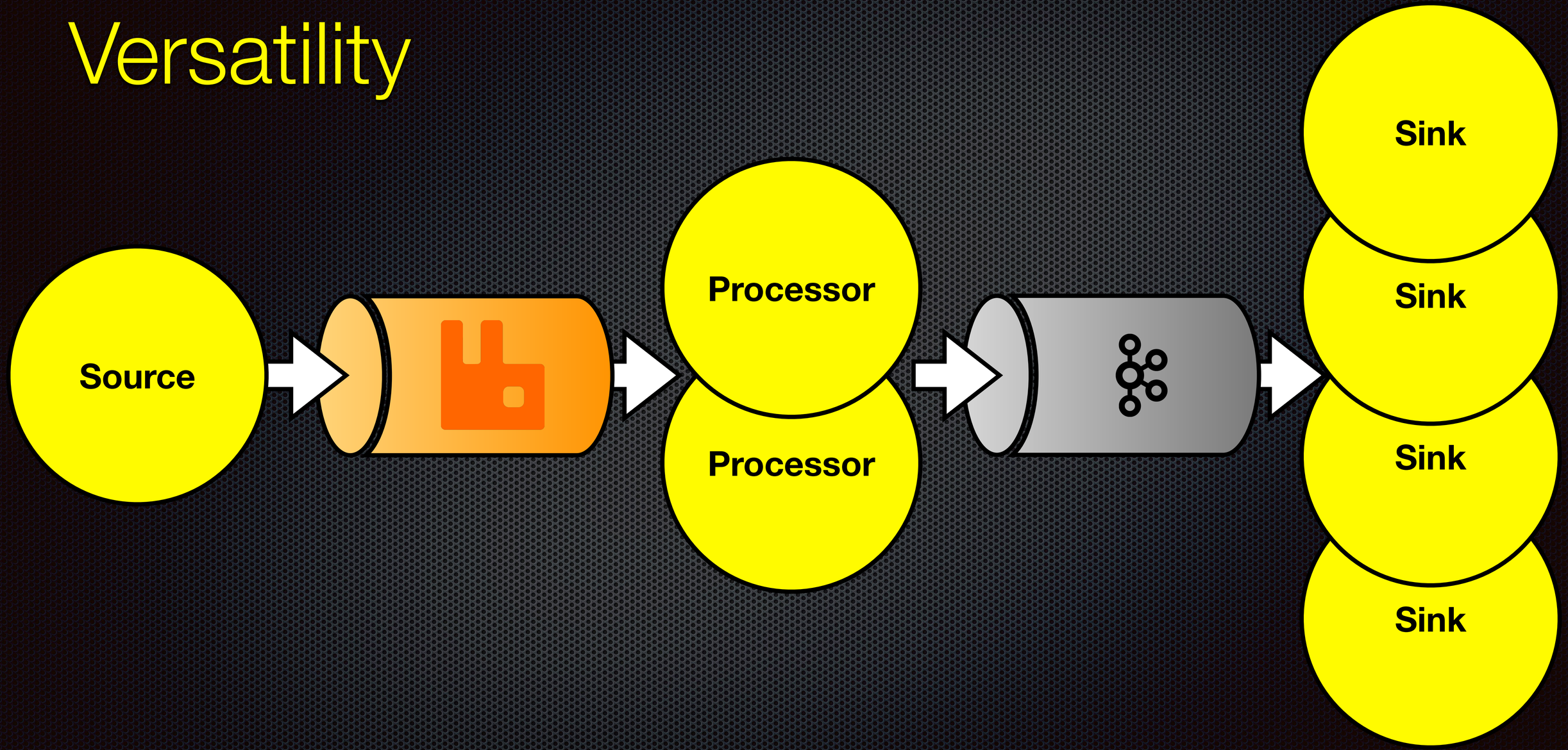
Scalability



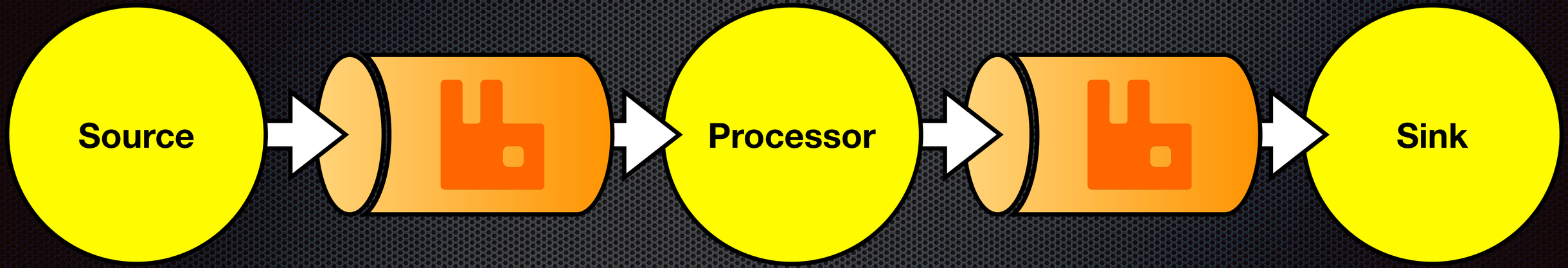
Flexibility



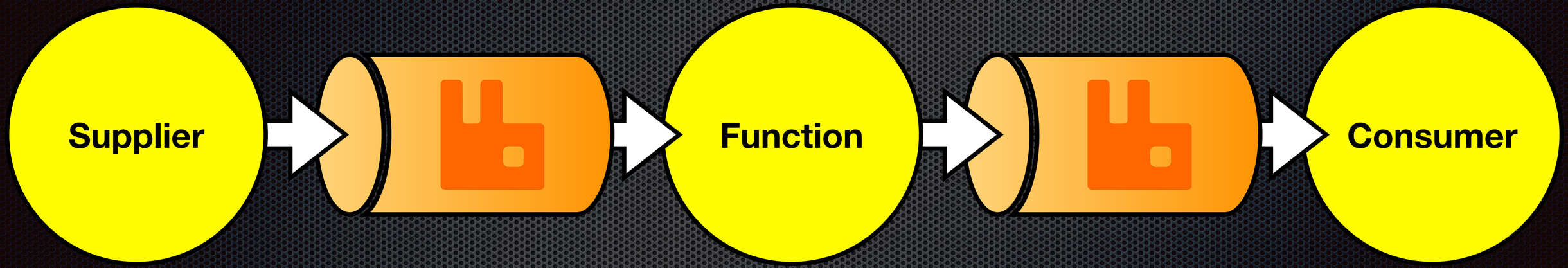
Versatility



Stream Revisited: Legacy



Stream Revisited: Evolution



And a few really useful bits & bobs
(if time permits, if not, please star the repo!)

Let’s code!

@mkheck

www.thehecklers.com

vmware®

Resources

- https://github.com/mkheck/game-of-streams-aircraft-edition
- https://spring.io/projects/spring-cloud-stream
- mark@thehecklers.com, mheckler@vmware.com
- @mkheck on Twitter

Kotlin fan? https://github.com/mkheck/game-of-streams-aircraft-edition-kotlin

In case of emergency, break glass

Source service dependencies

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-kafka</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

Source service properties

```
server.port=0

spring.cloud.stream.default-binder=rabbit
#spring.cloud.stream.poller.fixed-delay=5000

spring.cloud.stream.bindings.sendAircraftPositions-out-0.destination=processor
#spring.cloud.stream.bindings.sendAircraftPositions-out-0.binder=kafka

spring.cloud.stream.kafka.binder.min-partition-count=4
spring.cloud.stream.kafka.binder.auto-add-partitions=true

# DO NOT DO THIS IN PRODUCTION, FOR DEMO PURPOSES ONLY!!!
management.endpoints.web.exposure.include=*
```

Source service code

(imperative)

```
@SpringBootApplication
public class PsourceApplication {

    public static void main(String[] args) {
        SpringApplication.run(PsourceApplication.class, args);
    }

}

@Configuration
class PositionFeed {
    private final WebClient client = WebClient.create("http://localhost:8080");

    @Bean
    Supplier<List<Aircraft>> sendAircraftPositions() {
        return () -> {
            List<Aircraft> aircraftList = client.get()
                    .retrieve()
                    .bodyToFlux(Aircraft.class)
                    .filter(ac -> !ac.getReg().isEmpty())
                    .toStream()
                    .collect(Collectors.toList());

            aircraftList.forEach(System.out::println);

            return aircraftList;
        };
    }
}

@Data
@JsonIgnoreProperties(ignoreUnknown = true)
class Aircraft {
    private Long id;
    private String callsign, reg, flightno, type;
    private int altitude, heading, speed;
    private double lat, lon;
}
```

Source service code

(reactive)

```
@SpringBootApplication
public class PsourceApplication {

    public static void main(String[] args) {
        SpringApplication.run(PsourceApplication.class, args);
    }

}

@Configuration
class PositionFeed {
    private final WebClient client = WebClient.create("http://localhost:8080");

    @PollableBean
    Supplier<Flux<Aircraft>> sendAircraftPositions() {
        return () -> client.get()
                .retrieve()
                .bodyToFlux(Aircraft.class)
                .filter(ac -> !ac.getReg().isEmpty())
                .log();
    }
}

@Data
@JsonIgnoreProperties(ignoreUnknown = true)
class Aircraft {
    private Long id;
    private String callsign, reg, flightno, type;
    private int altitude, heading, speed;
    private double lat, lon;
}
```

Processor service dependencies

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-kafka</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

Processor service properties

```
server.port=0

spring.cloud.stream.default-binder=rabbit
spring.cloud.function.definition=transformACIfilterNullCallsigns

spring.cloud.stream.bindings.transformACIfilterNullCallsigns-in-0.destination=processor
spring.cloud.stream.bindings.transformACIfilterNullCallsigns-in-0.group=processor
#spring.cloud.stream.bindings.transformACIfilterNullCallsigns-in-0.binder=kafka

spring.cloud.stream.bindings.transformACIfilterNullCallsigns-out-0.destination=sink
#spring.cloud.stream.bindings.transformACIfilterNullCallsigns-out-0.binder=rabbit

spring.cloud.stream.kafka.binder.min-partition-count=4
spring.cloud.stream.kafka.binder.auto-add-partitions=true

# Again, DO NOT DO THIS IN PROD!
management.endpoints.web.exposure.include=*
```

Processor service code

(imperative)

```
@Configuration
class AircraftTransformer {
    @Bean
    Function<List<Aircraft>, List<EssentialAircraft>> transformAC() {
        return listAC -> {
            List<EssentialAircraft> listEssential = new ArrayList<>();

            listAC.forEach(ac -> listEssential.add(new EssentialAircraft(ac.getCallsign(),
                    ac.getReg(),
                    ac.getType())));

            listEssential.forEach(System.out::println);

            return listEssential;
        };
    }
}

@Data
@AllArgsConstructor
class Aircraft {
    private Long id;
    private String callsign, reg, flightno, type;
    private int altitude, heading, speed;
    private double lat, lon;
}

@Data
class EssentialAircraft {
    private String callsign, reg, type, mfr;

    public EssentialAircraft(String callsign, String reg, String type) {
        this.callsign = callsign;
        this.reg = reg;
        this.type = type;

        setMfr();
    }

    private void setMfr() {
        mfr = switch (type.substring(0, 2)) {
            case "A2", "A3" -> "Airbus";
            case "BE", "B3" -> "Beechcraft";
            case "B7" -> "Boeing";
            case "C1", "C2", "C4", "C6", "C7", "T2" -> "Cessna";
            case "CR", "E7" -> "Embraer";
            case "LJ" -> "Learjet";
            case "MD" -> "McDonnell Douglas";
            case "PA" -> "Piper";
            default -> "Other";
        };
    }
}
```

Processor service code

(reactive)

```
@Configuration
class AircraftTransformer {
	@Bean
	Function<Flux<Aircraft>, Flux<EssentialAircraft>> transformAC() {
		return aircraftFlux -> aircraftFlux.map(ac -> new EssentialAircraft(ac.getCallsign(),
				ac.getReg(),
				ac.getType()));
		//.log();
	}

	@Bean
	Function<Flux<EssentialAircraft>, Flux<EssentialAircraft>> filterNullCallsigns() {
		return flux -> flux.filter(ac -> null != ac.getCallsign())
				.log();
	}
}

@Data
@AllArgsConstructor
class Aircraft {
	private Long id;
	private String callsign, reg, flightno, type;
	private int altitude, heading, speed;
	private double lat, lon;
}

@Data
class EssentialAircraft {
	private String callsign, reg, type, mfr;

	public EssentialAircraft(String callsign, String reg, String type) {
		this.callsign = callsign;
		this.reg = reg;
		this.type = type;

		setMfr();
	}

	private void setMfr() {
		mfr = switch (type.substring(0, 2)) {
			case "A2", "A3" -> "Airbus";
			case "BE", "B3" -> "Beechcraft";
			case "B7" -> "Boeing";
			case "C1", "C2", "C4", "C6", "C7", "T2" -> "Cessna";
			case "CR", "E7" -> "Embraer";
			case "LJ" -> "Learjet";
			case "MD" -> "McDonnell Douglas";
			case "PA" -> "Piper";
			default -> "Other";
		};
	}
}
```

Sink service dependencies

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-kafka</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-binder-rabbit</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

Sink service properties

```
server.port=0

spring.cloud.stream.bindings.logIt-in-0.destination=sink
spring.cloud.stream.bindings.logIt-in-0.group=sink
spring.cloud.stream.bindings.logIt-in-0.binder=rabbit

spring.cloud.stream.kafka.binder.min-partition-count=4
spring.cloud.stream.kafka.binder.auto-add-partitions=true

# Once more (I promise): DO NOT DO THIS IN PRODUCTION!!!
management.endpoints.web.exposure.include=*
```

Sink service code

(imperative)

```
@SpringBootApplication
public class PsinkApplication {

    public static void main(String[] args) {
        SpringApplication.run(PsinkApplication.class, args);
    }

}

@Configuration
class AircraftLogger {
    @Bean
    Consumer<List<EssentialAircraft>> logIt() {
        return listEssentialAC -> listEssentialAC.forEach(System.out::println);
    }
}

@Data
@AllArgsConstructor
class EssentialAircraft {
    private String callsign, reg, type;
    private String mfr;
}
```

Sink service code

(reactive)

```
@SpringBootApplication
public class PsinkApplication {

    public static void main(String[] args) {
        SpringApplication.run(PsinkApplication.class, args);
    }

}

@Configuration
class AircraftLogger {
    @Bean
    Consumer<Flux<EssentialAircraft>> logIt() {
        return eACFlux -> eACFlux.subscribe(System.out::println);
    }
}

@Data
@AllArgsConstructor
class EssentialAircraft {
    private String callsign, reg, type;
    private String mfr;
}
```

Sample data

PlaneFinder API Gateway

```
morpheus :: OReilly/code/planefinder <master> » http :8080
HTTP/1.1 200
Connection: keep-alive
Content-Type: application/json
Date: Wed, 01 Jul 2020 04:18:20 GMT
Keep-Alive: timeout=60
Transfer-Encoding: chunked

[
    {
        "altitude": 35000,
        "barometer": 1012.8,
        "bds40_seen_time": "2020-07-01T04:18:15Z",
        "callsign": "UPS499",
        "category": "A4",
        "flightno": "",
        "heading": 99,
        "id": 1,
        "is_adsb": true,
        "is_on_ground": false,
        "last_seen_time": "2020-07-01T04:18:19Z",
        "lat": 38.712469,
        "lon": -89.874146,
        "polar_bearing": 135.762071,
        "polar_distance": 20.111867,
        "pos_update_time": "2020-07-01T04:18:19Z",
        "reg": "N450UP",
        "route": "YVR-SDF",
        "selected_altitude": 35008,
        "speed": 477,
        "squawk": "2437",
        "type": "B752",
        "vert_rate": 0
    }
]
```