

Reactive Applications with Eclipse Vert.x

**Julien Viet
QCon Shanghai**

CNUTCon [上海]
全球运维技术大会

主办方 GeekBang InfoQ
极客邦科技 极客邦科技

50+ 年末充电[⚡]

开发&运维技术干货大盘点

容器

Kubernetes

DevOps

全链路压测

Severless

自动化运维

Service Mesh

Elasticsearch

微服务

使用折扣码「QCon」 优惠报名 咨询电话：13269078023



扫码锁定席位

Julien Viet

Open source developer for 16+ years

@**vertx_project** lead

Principal software engineer at  **redhat**.

Marseille JUG Leader

 <https://www.julienviet.com/>

 <http://github.com/vietj>

 @julienviet

 <https://www.mixcloud.com/cooperdbi/>

Outline

- ✓ Reactive? Vert.x?
- ✓ Foundations
- ✓ Reactive Programming
- ✓ Ecosystem

Reactive?

Vert.x?

Workbook1

Q Search Sheet

😊

Home

Insert

Page Layout

Formulas

Data

Review

View

+ Share

^

Paste

Font

Alignment

Number

Conditional Formatting

Format as Table

Cell Styles

Cells

Editing

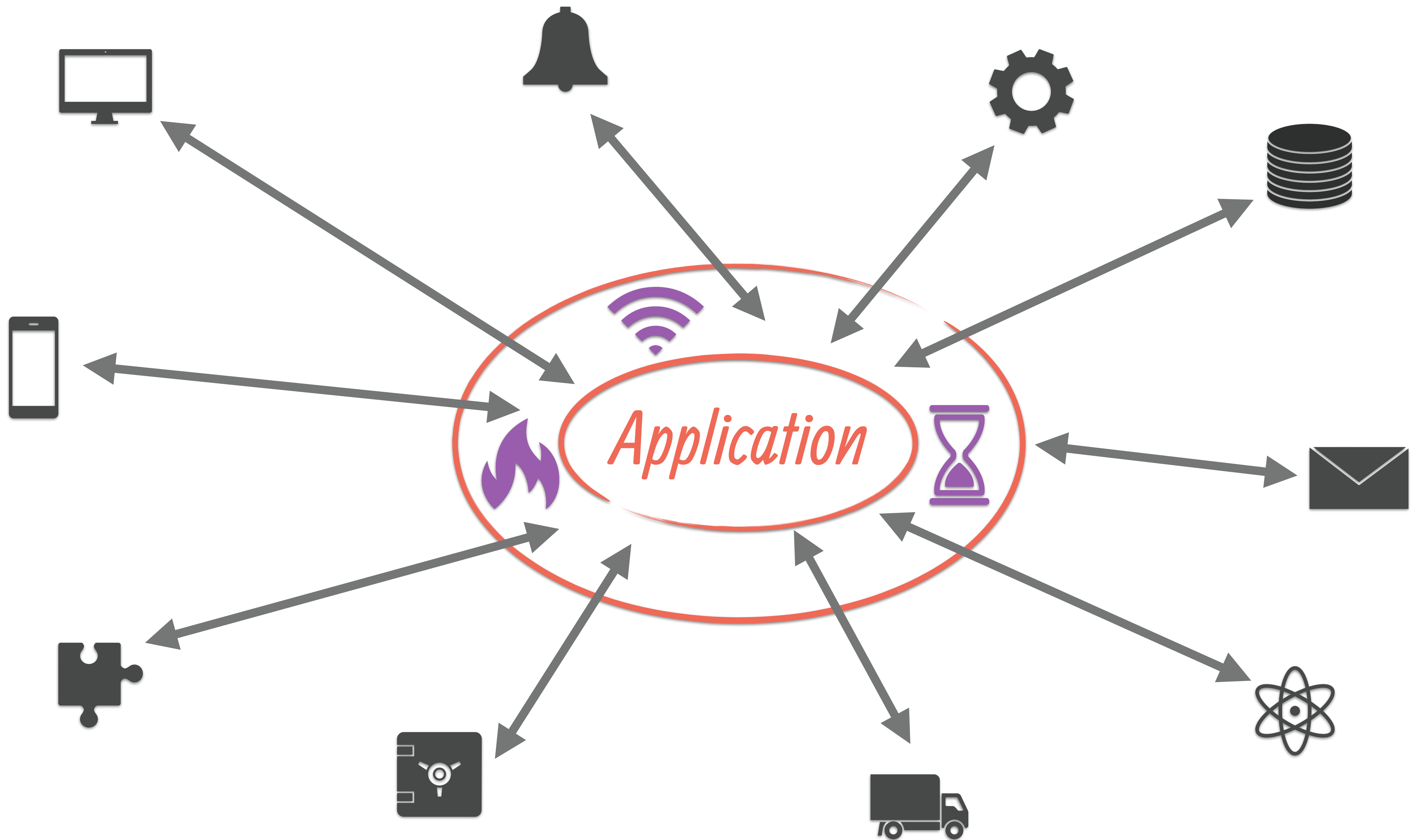
AVERAGE

✖

✓

fx

=A1+A2



Software



Metrics

Availability

Messages

Requests

Reactive

"Responding to stimuli"

*Manifesto, Actor, Messages
Resilience, Elasticity, Scalability,
Asynchronous, non-blocking*

*Data flow
Events, Observable
Spreadsheets*

**Reactive
systems**

**Reactive
streams**

**Reactive
programming**

*Data flow
Back-pressure
Non-blocking*

Akka, **Vert.x**

Akka Streams, Rx v2,
Reactor, **Vert.x**

Reactor, Reactive Spring,
RxJava, **Vert.x**

Eclipse Vert.x

Open source project started in 2012

Eclipse / Apache licensing

A **toolkit** for building **reactive** applications for the JVM

8K ★ on 

Built on top of  Netty


 <https://vertx.io>

 @vertx_project

Reactive foundations

Reactive foundations

- ✓ Concurrency model
- ✓ Scalability model
- ✓ High performance networking
- ✓ Vert.x Event Bus



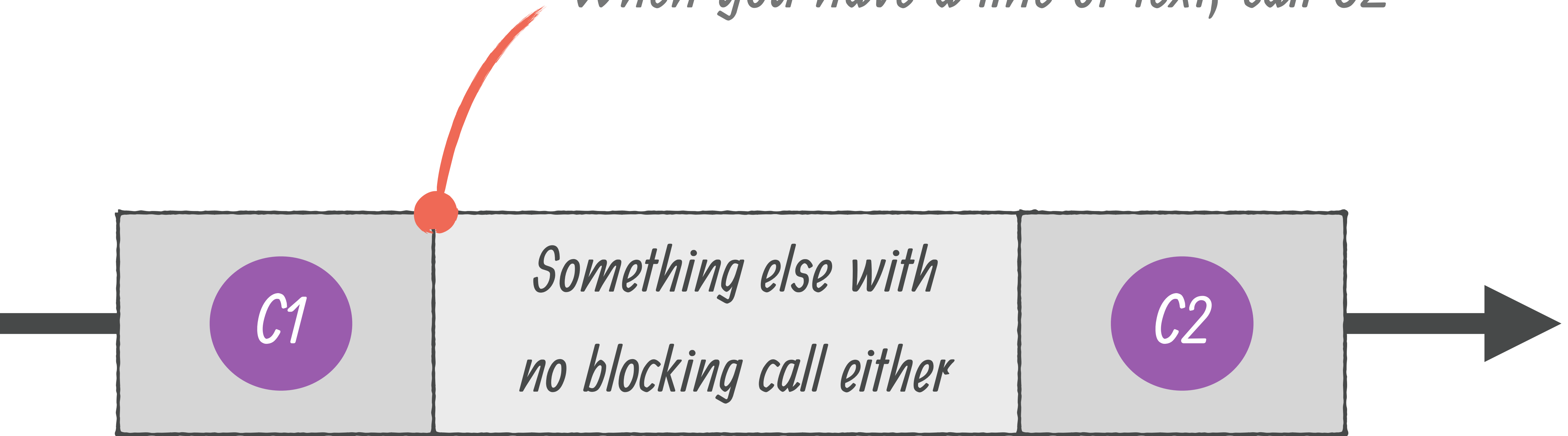
```
while (isRunning) {  
    String line = bufferedReader.readLine();  
    switch (line.substring(0, 4)) {  
        case "ECHO":  
            bufferedWriter.write(line);  
            break  
        // ...  
        // other cases ( ... )  
        // ...  
        default:  
            bufferedWriter.write("Unknown command");  
    }  
}
```

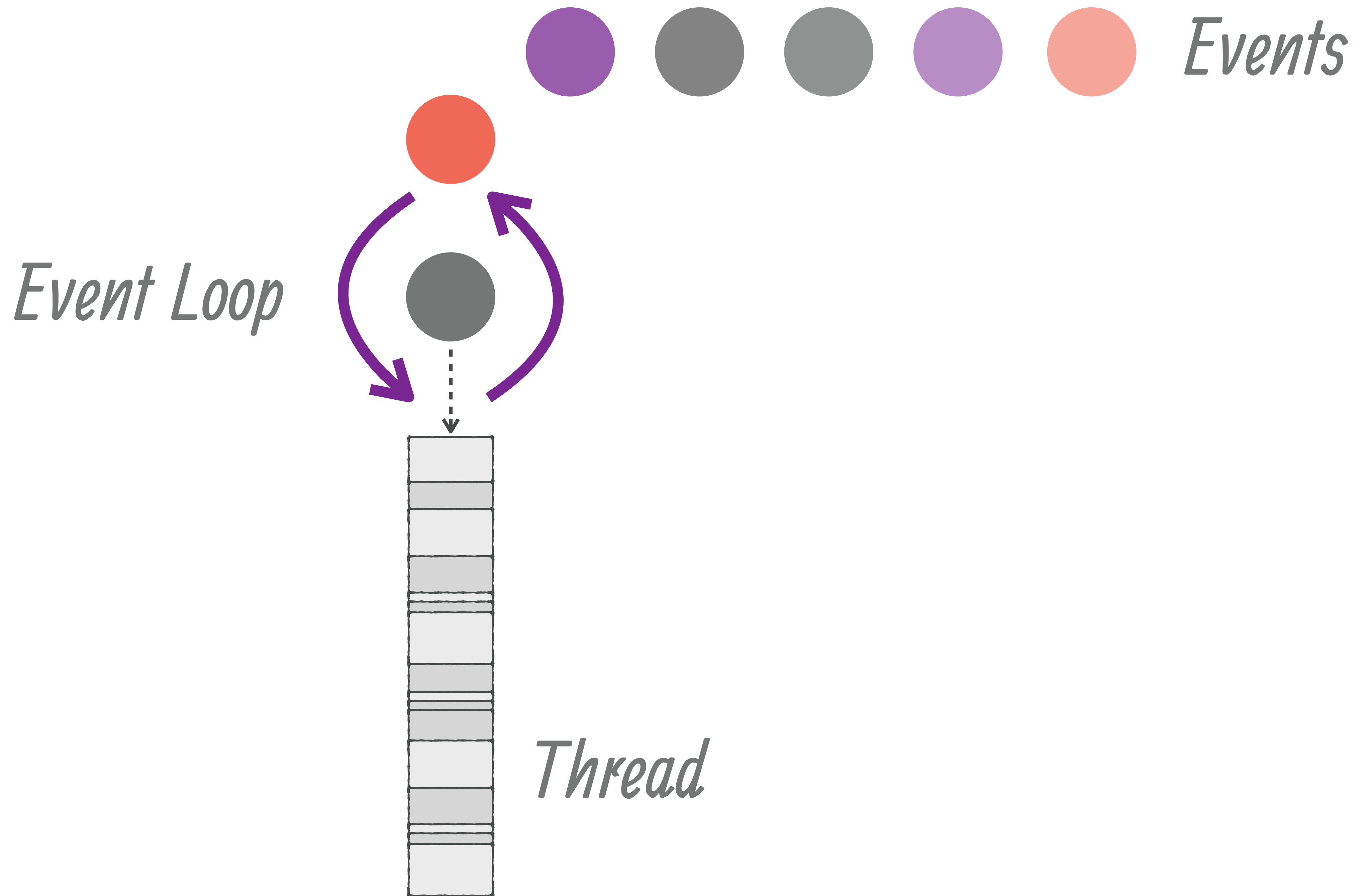


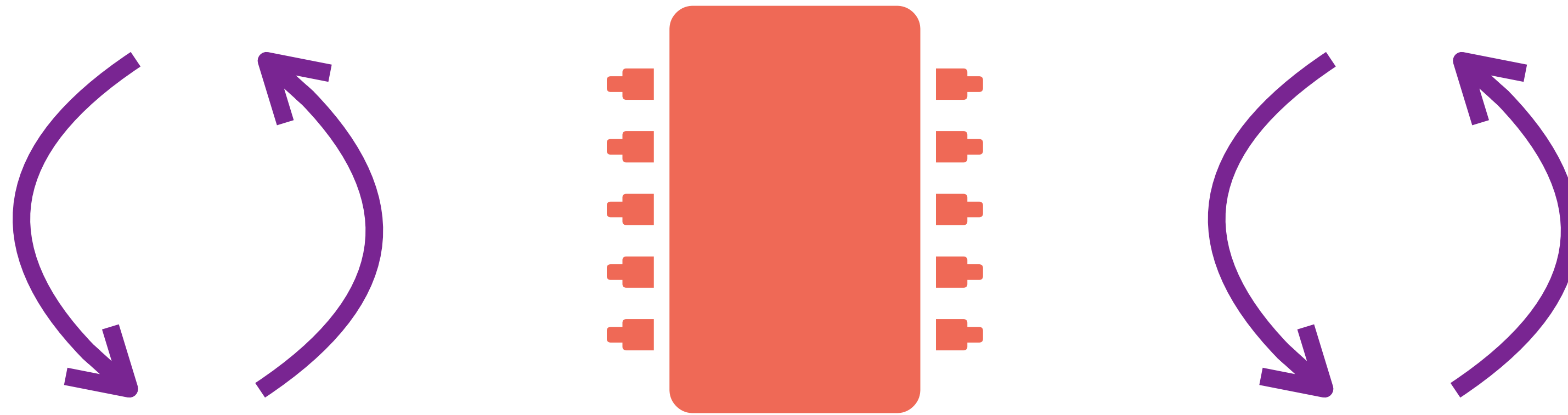
x 1000 =



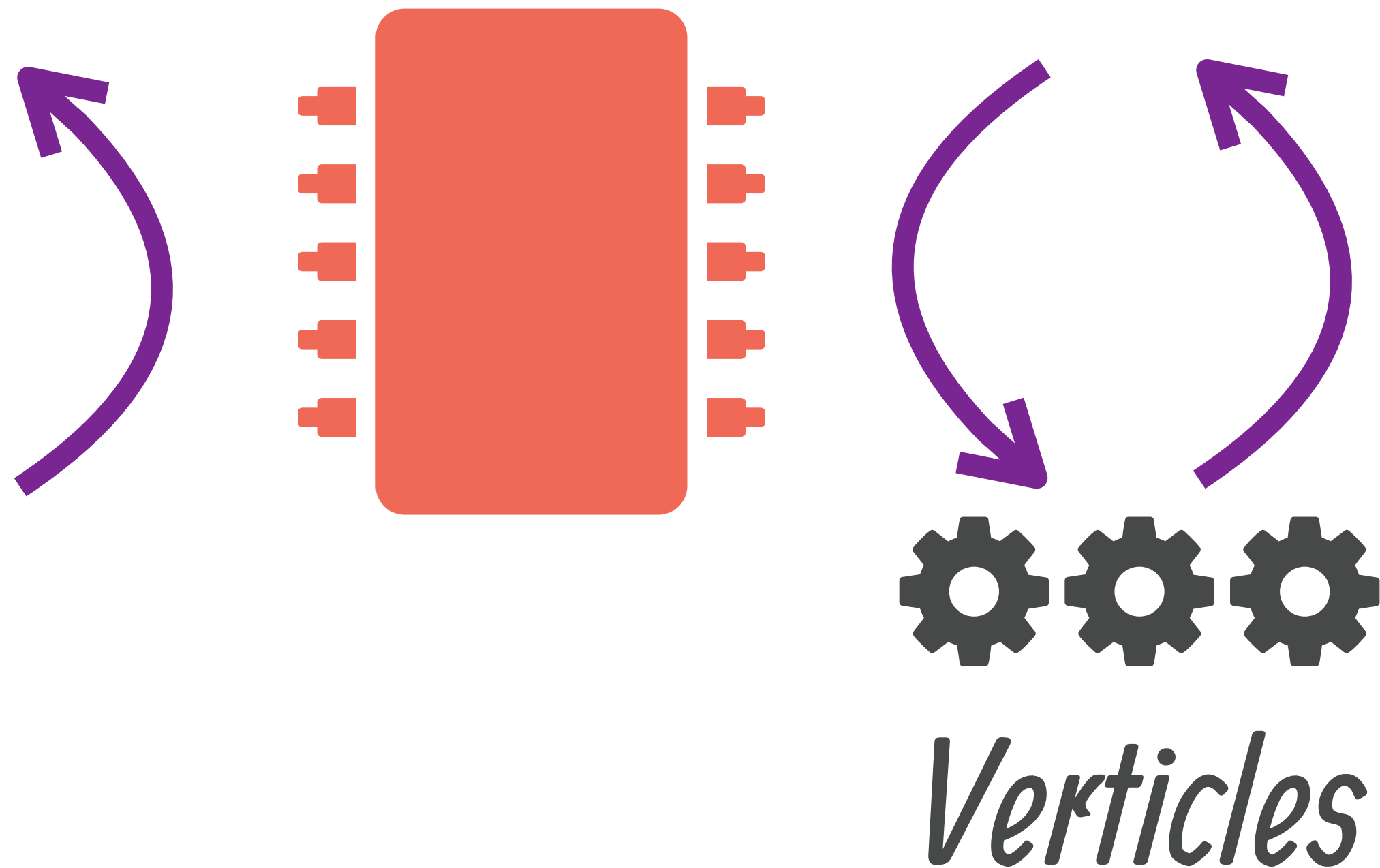
"When you have a line of text, call C2"







2 event-loops per CPU core by default



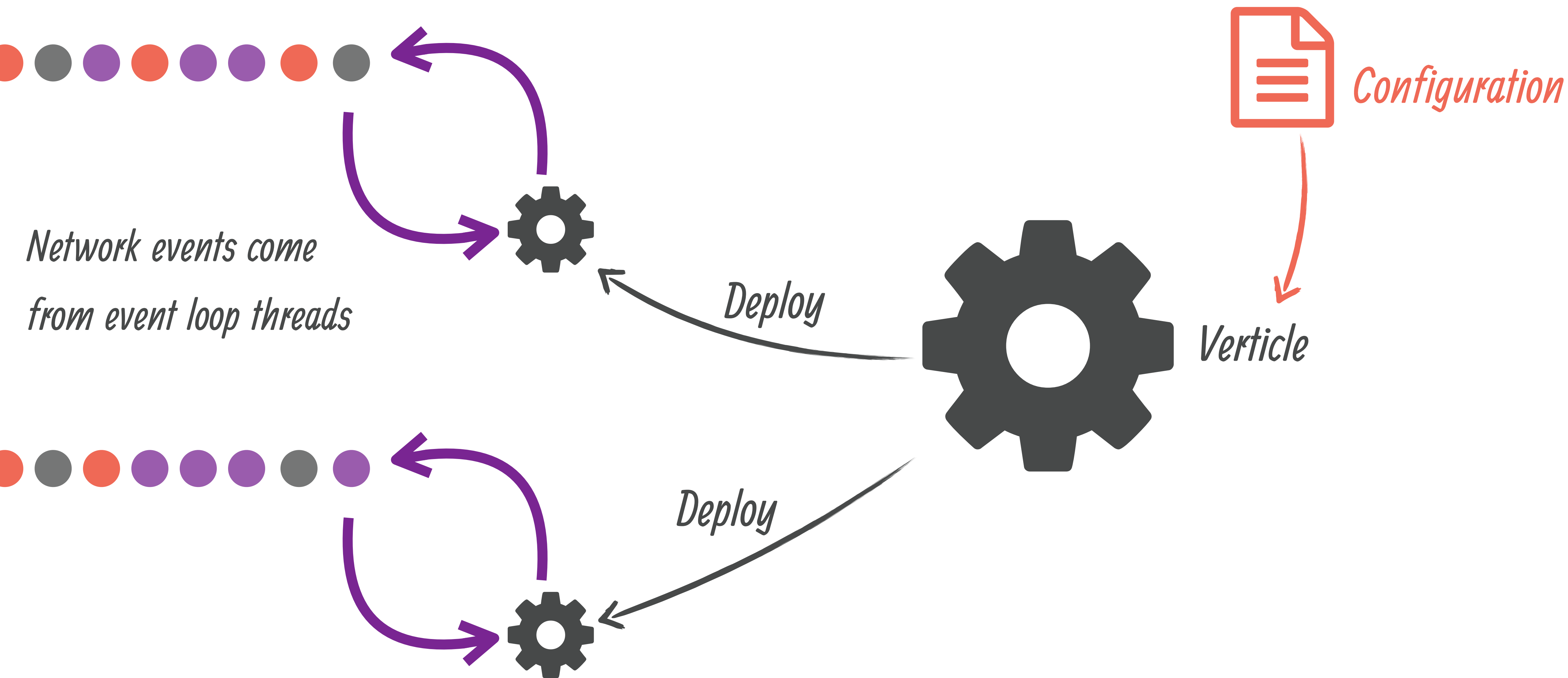
```
public class HttpVerticle extends AbstractVerticle {

    public static void main(String[] args) {
        Vertx vertx = Vertx.vertx();
        vertx.deployVerticle(HttpVerticle.class.getName());
        System.in.read();
        vertx.close();
    }

    @Override
    public void init(Vertx vertx, Context context) {
        // Associates this verticle with
        // its deployment context
        super.init(vertx, context);
    }

    @Override
    public void start() throws Exception {
        HttpServer server = vertx.createHttpServer();

        server.requestHandler(request -> {
            request
                .response()
                .end("Hello World");
        }).listen(8080);
    }
}
```



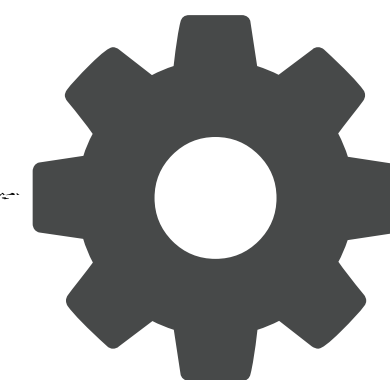
Simplified concurrency model

```
public class ChatVerticle extends AbstractVerticle {  
  
    private Set<ServerWebSocket> webSockets = new HashSet<>();  
  
    @Override  
    public void start() throws Exception {  
        HttpServer server = vertx.createHttpServer();  
  
        server.websocketHandler(webSocket -> {  
  
            websocket.textMessageHandler(msg -> {  
                webSockets.forEach(other -> other.writeTextMessage(msg));  
            });  
  
            websocket.closeHandler(v -> webSockets.remove(webSocket));  
  
            webSockets.add(webSocket);  
        }).listen(8080);  
    }  
}
```

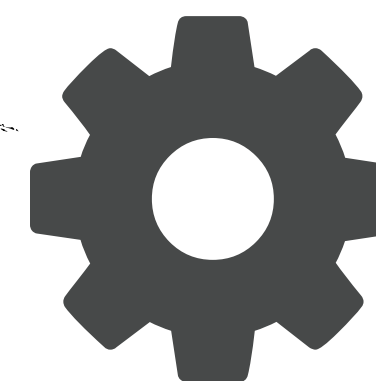
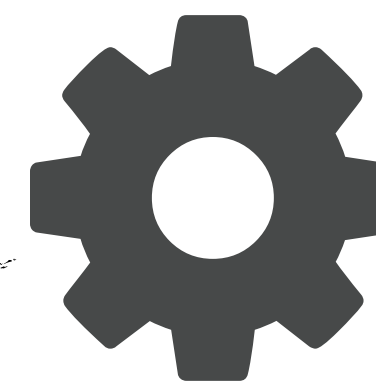
(demo)

Polling dashboard

*Monitoring
dashboard
app*



*Metrics
aggregation*



*Reporting
agents*

High performance networking

Performance

- ✓ Building bottom up
- ✓ Non blocking IO
- ✓ Reactive back-pressure

Pay the right price

- ✓ Very small footprint and startup time
- ✓ Do one thing and do it well
- ✓ Does not solve other (non) problems such as class loading or IoC
- ✓ Modular set of extensions

Non-blocking IO benefits

- ✓ Handle many connections with a few threads
 - focus on protocol concurrency
 - minimise system calls
 - more efficient for pipelined/multiplexed protocols

Non-blocking IO benefits

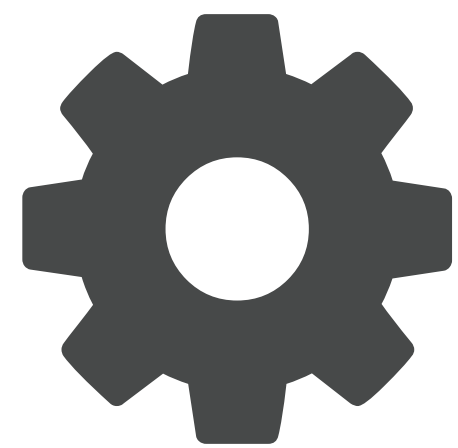
- ✓ Get away from thread pools
 - removes unnecessary bottlenecks
 - easier capacity planning
 - concurrency is protocol intrinsic

Non-blocking IO benefits

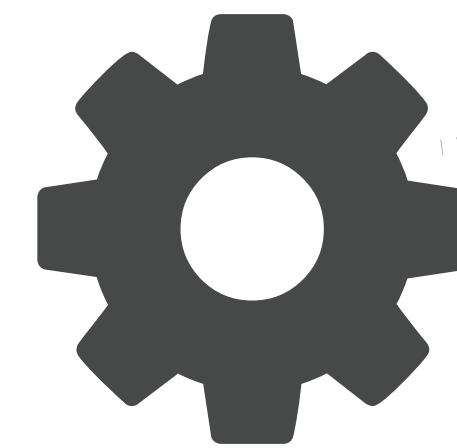
- ✓ Gracefully handle slow connection
 - remain responsive
 - don't impact other connections

Demo: Polling dashboard

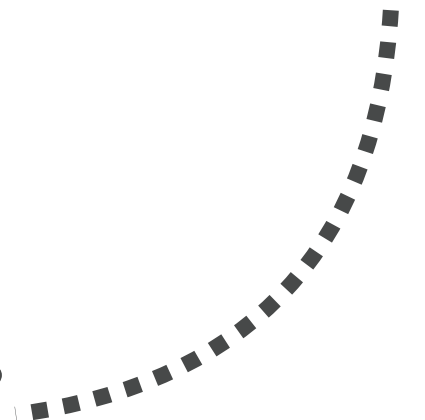
Message passing on the event bus

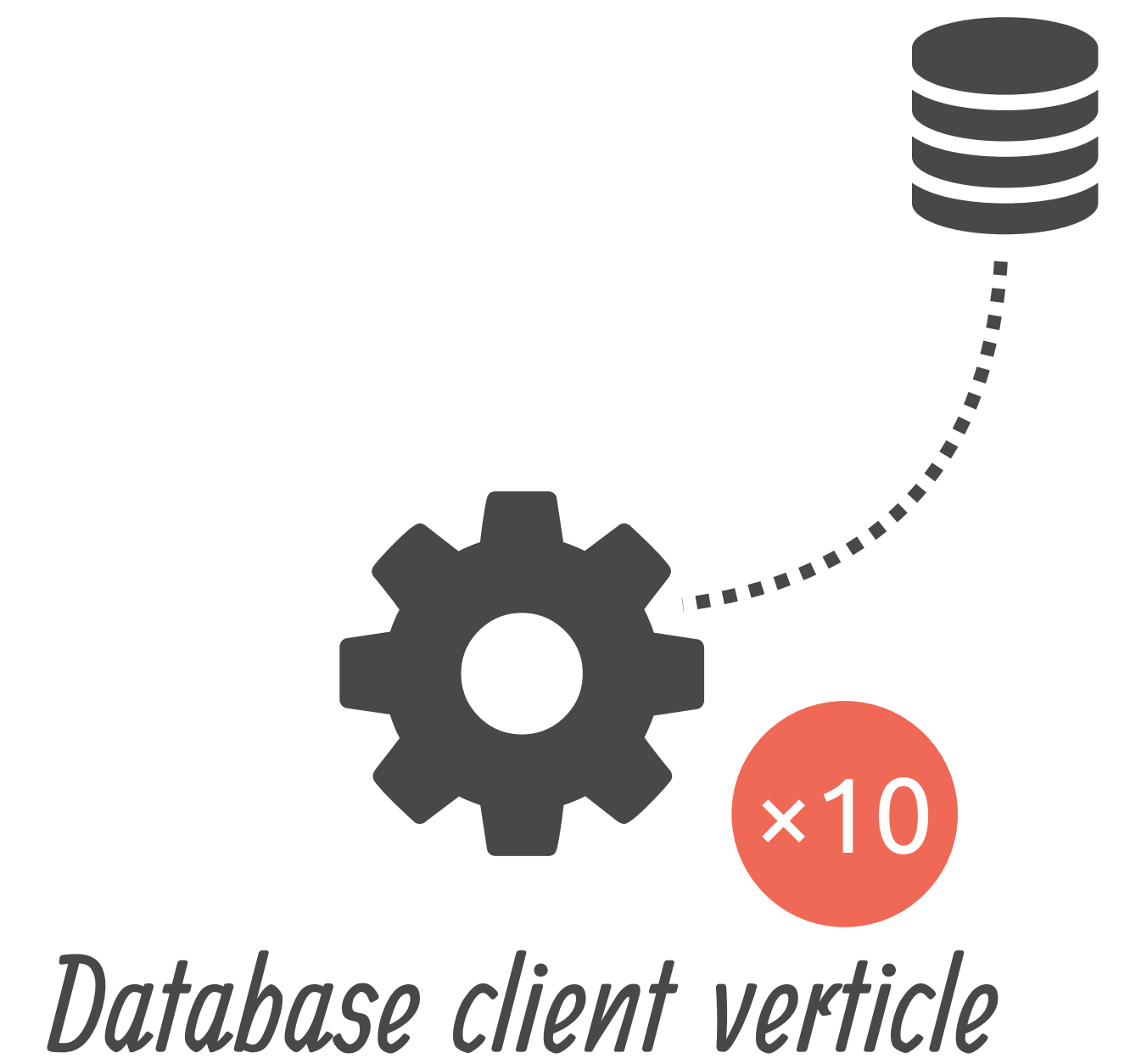
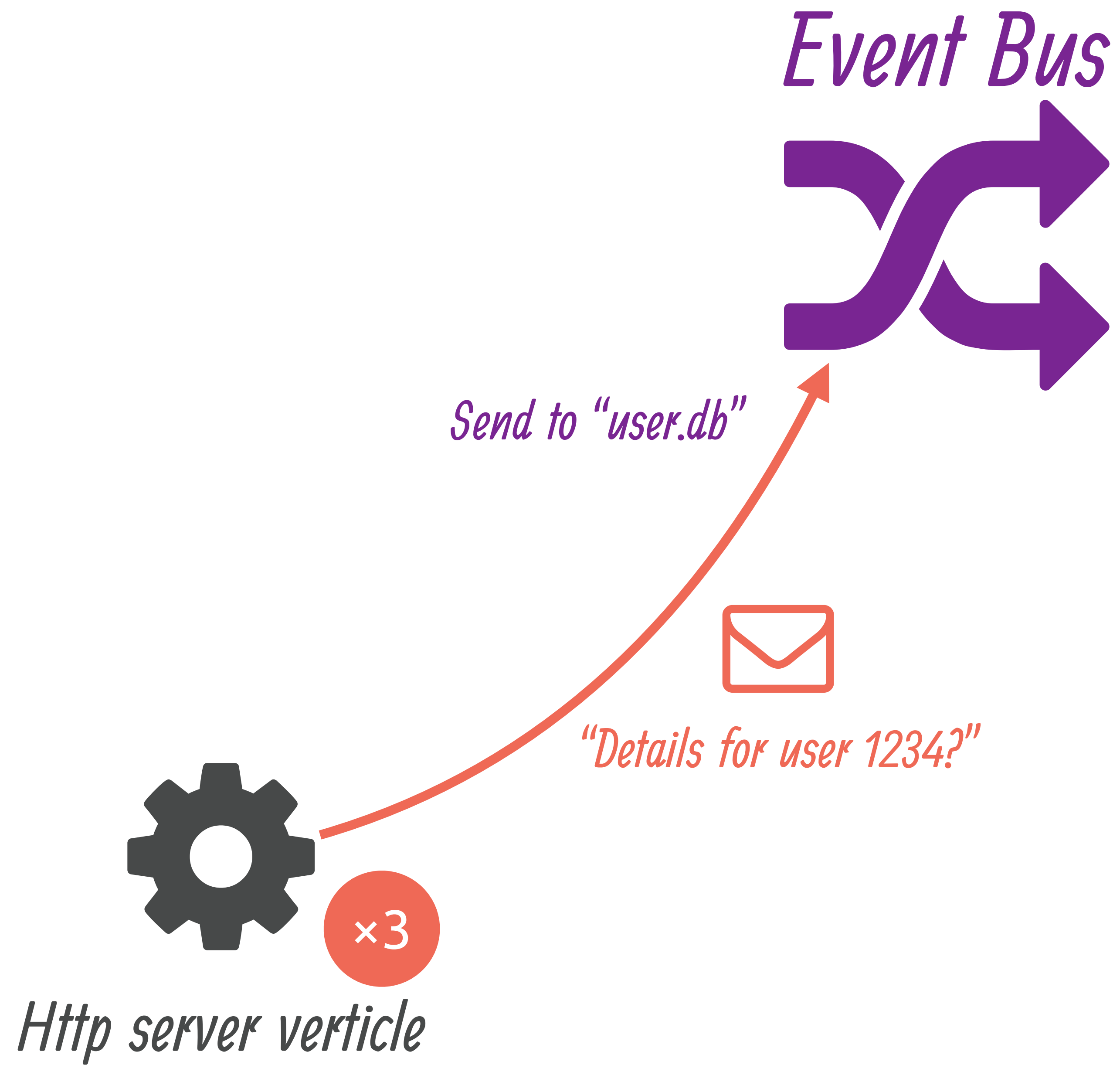


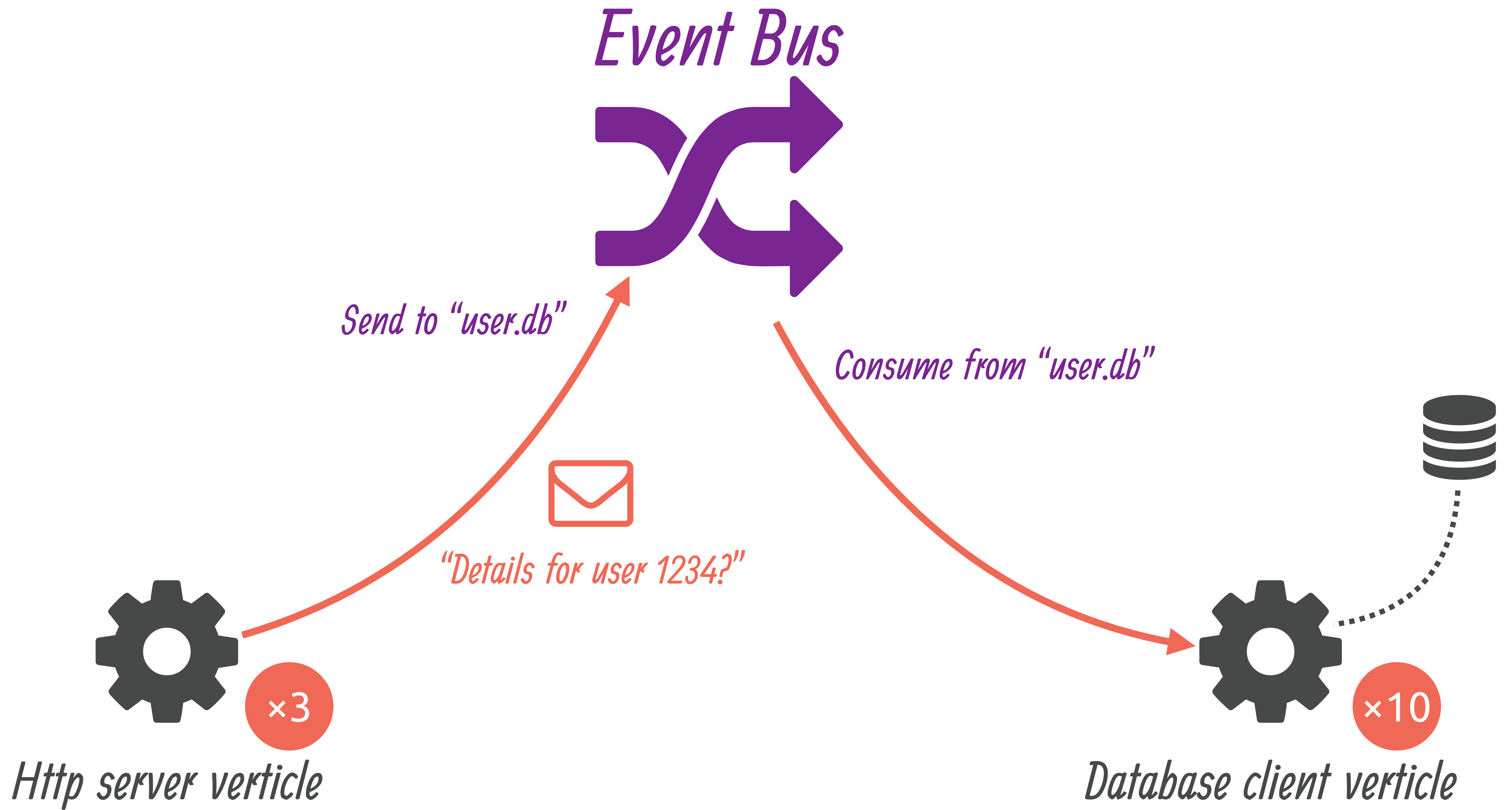
Http server verticle

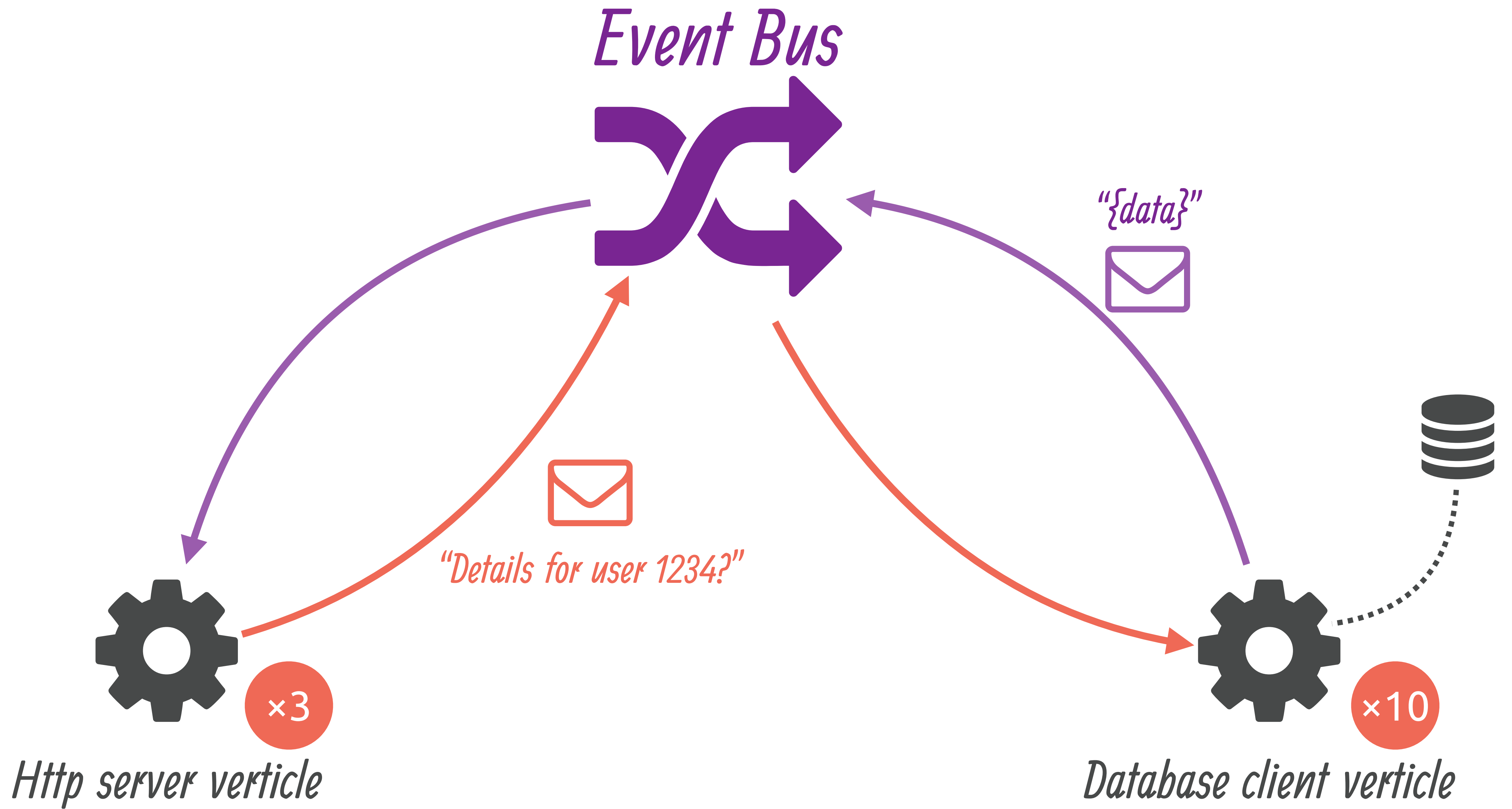


Database client verticle







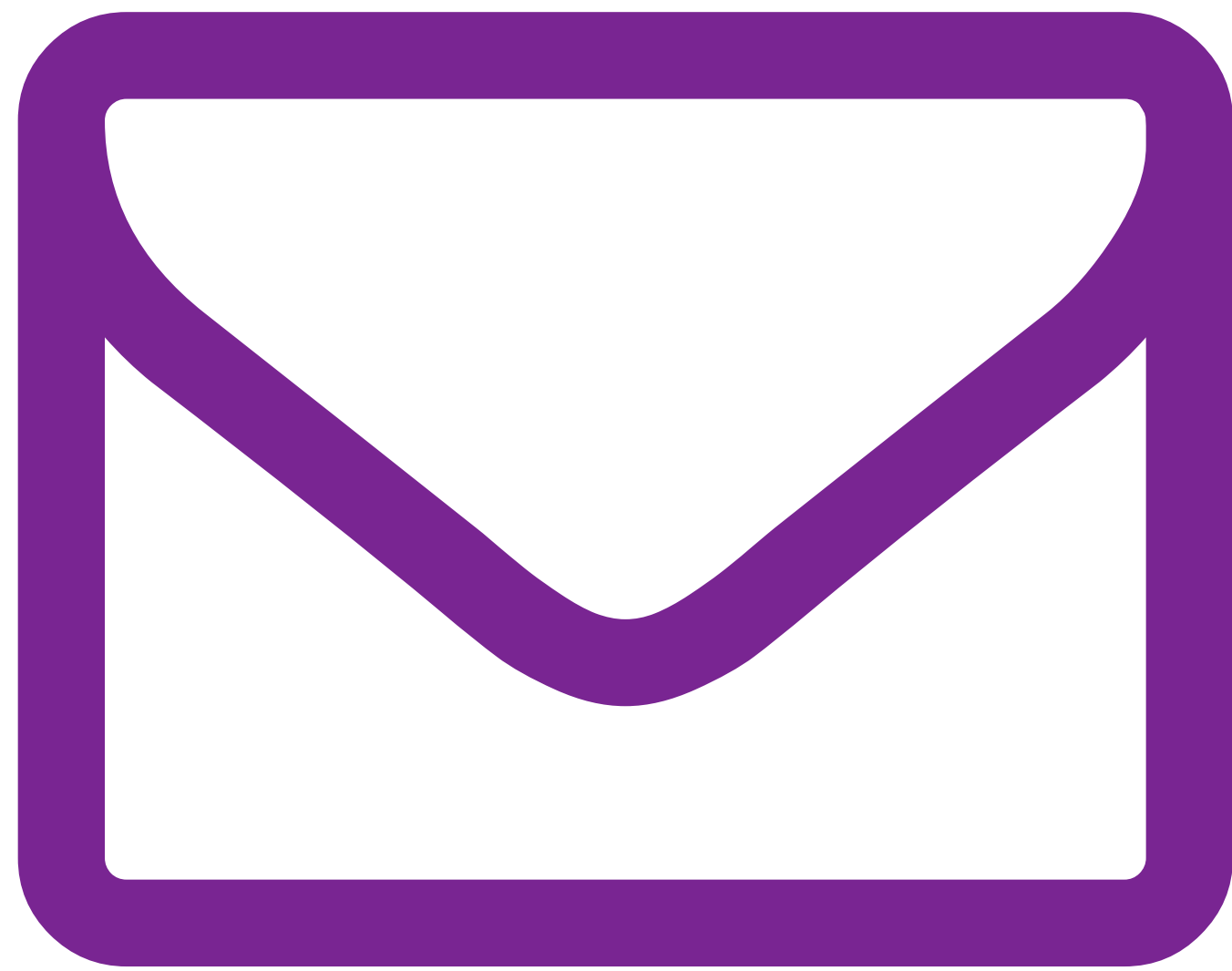




Asynchronous messaging

“foo.bar”, “foo-bar”, “foo/bar”, ...

Point to point (possible response back)
Publish / subscribe



Headers

DeliveryOptions (e.g., timeouts)

Body

Address

Reply address



“Primitive” types

String, int, double, ...

JSON Object/Array

Polyglot applications, clean boundaries

Custom codecs

For advanced usages

Demo: push dashboard



Distributed across Vert.x nodes

Hazelcast, Ignite, Infinispan, ...

TCP bridge interface

Go, Python, C, JavaScript, Swift, C#, ...

SockJS bridge

Seamless frontend / backend messaging

Demo: bridge dashboard

Reactive Programming with Vert.x and RxJava

RxJava

Data and **events** flows

Organising **transformation** of **data** and **coordination** of **events**

Makes most sense with many **sources of events**

Motivation

`Future<List<T>>` is not always appropriate

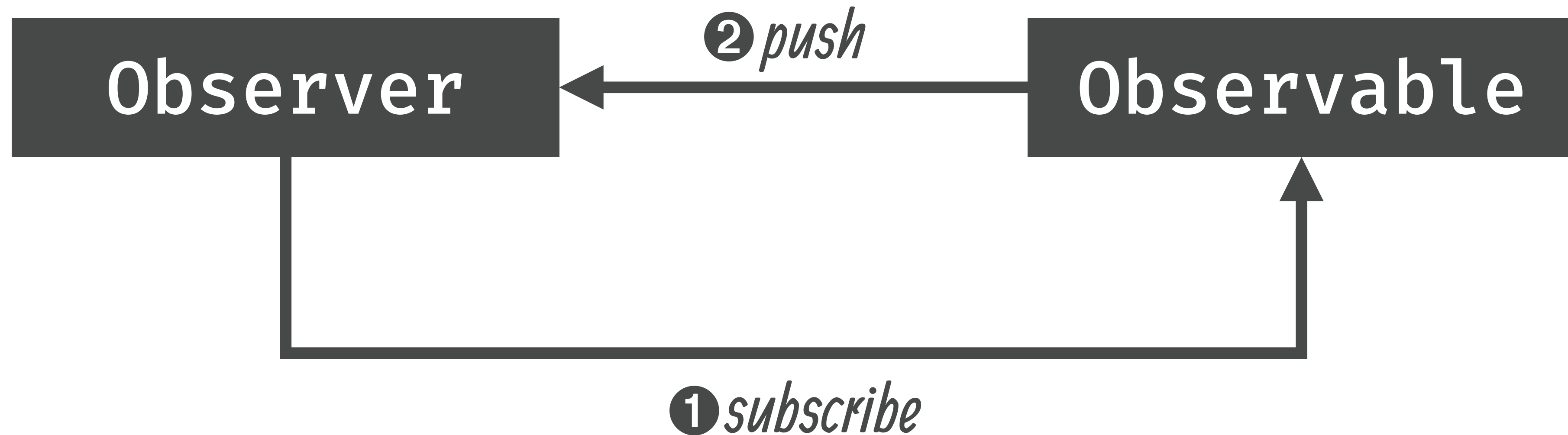
Dealing with **latencies**

Functional programming influence

RxJava 2 types

Reactive	Completable	Maybe<T>	Single<T>	Observable<T>
Interactive	void	Optional<T>	T	Iterable<T>

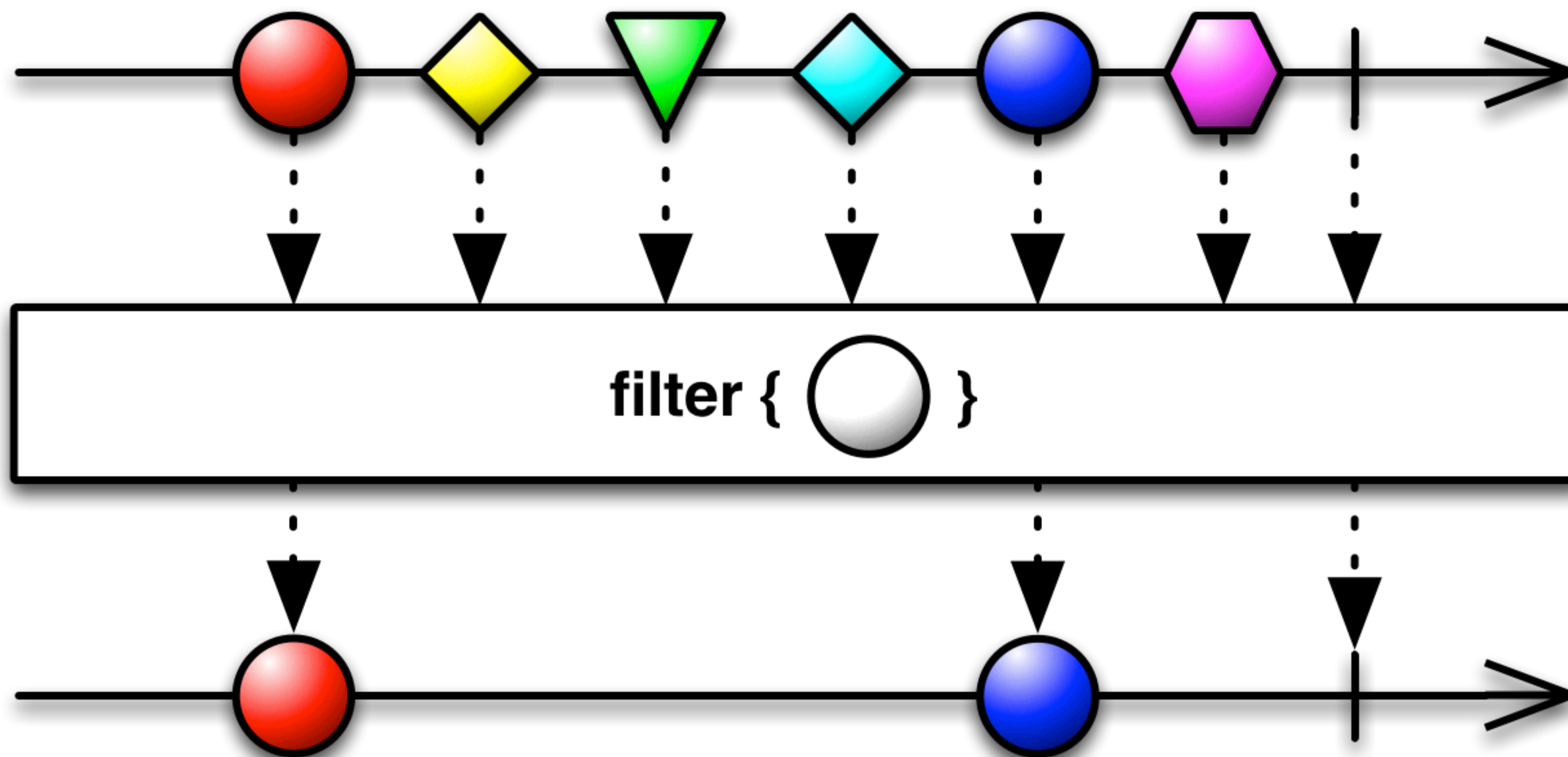
Event push

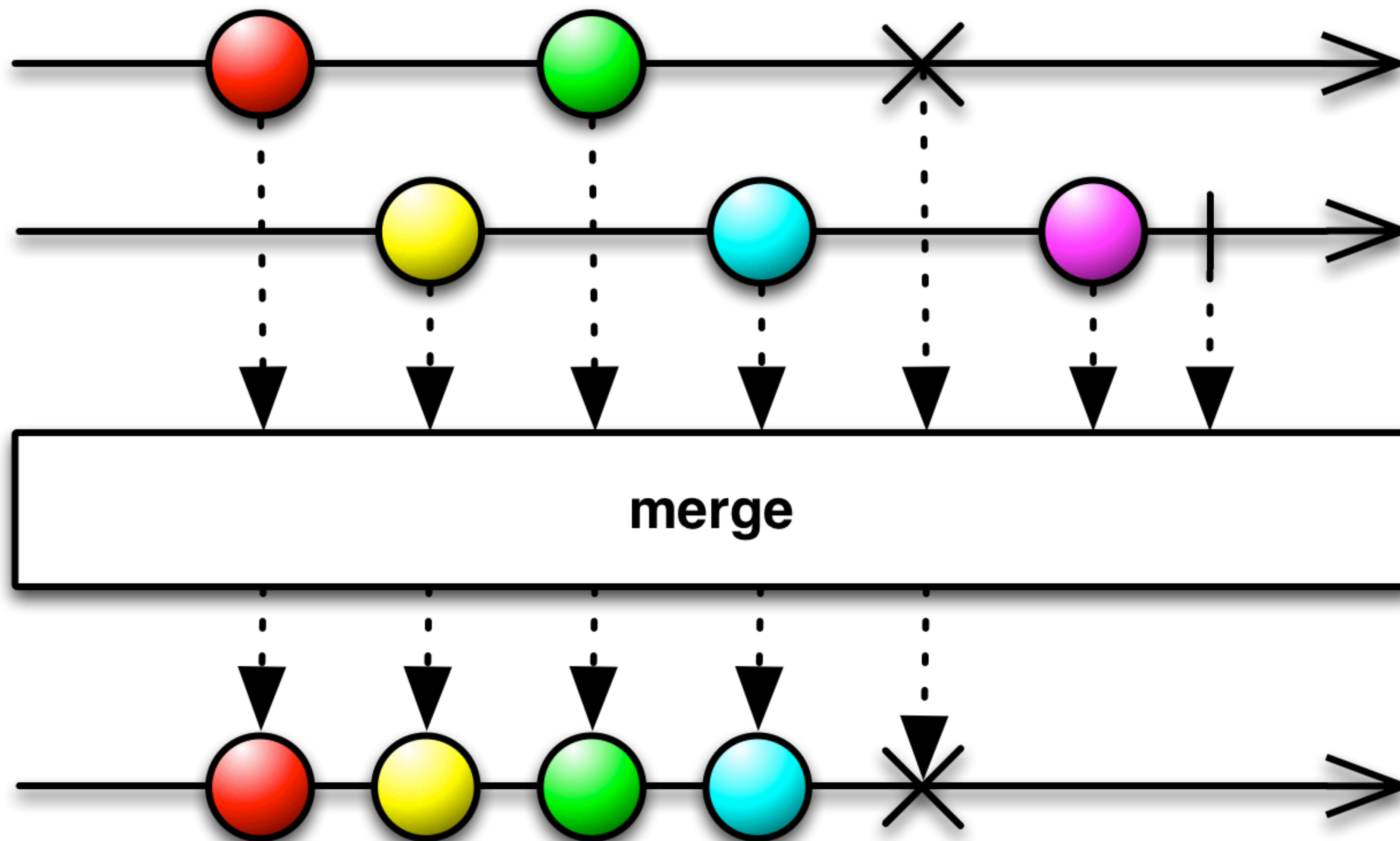


Iterable / Observable

```
try {  
    for (String item : it) {  
        ❶  
    }  
    ❸  
} catch (Throwable e) {  
    ❷  
}
```

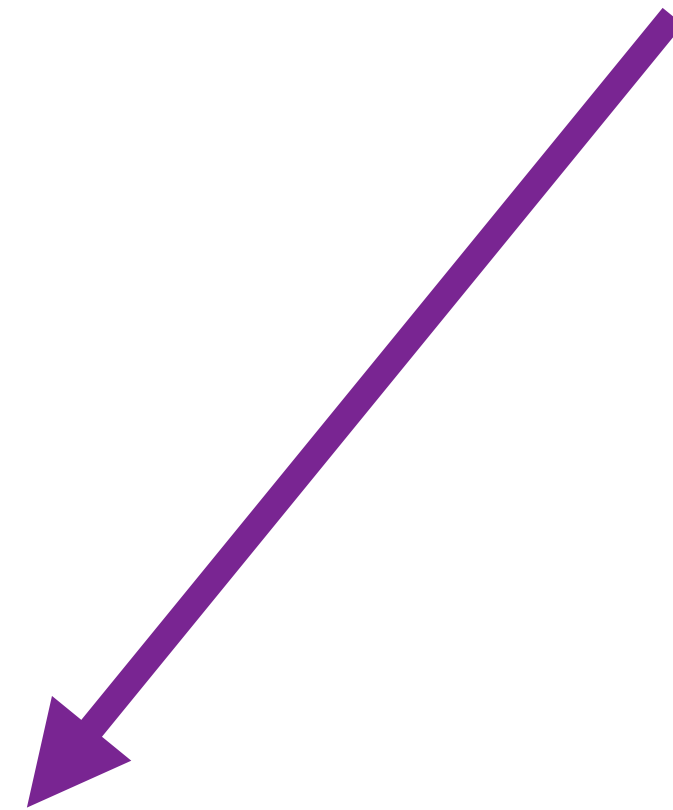
```
observable.subscribe(item ->  
{  
    ❶ // onNext  
}, error -> {  
    ❷ // onError  
}, () -> {  
    ❸ // onComplete  
});
```





Rxified API

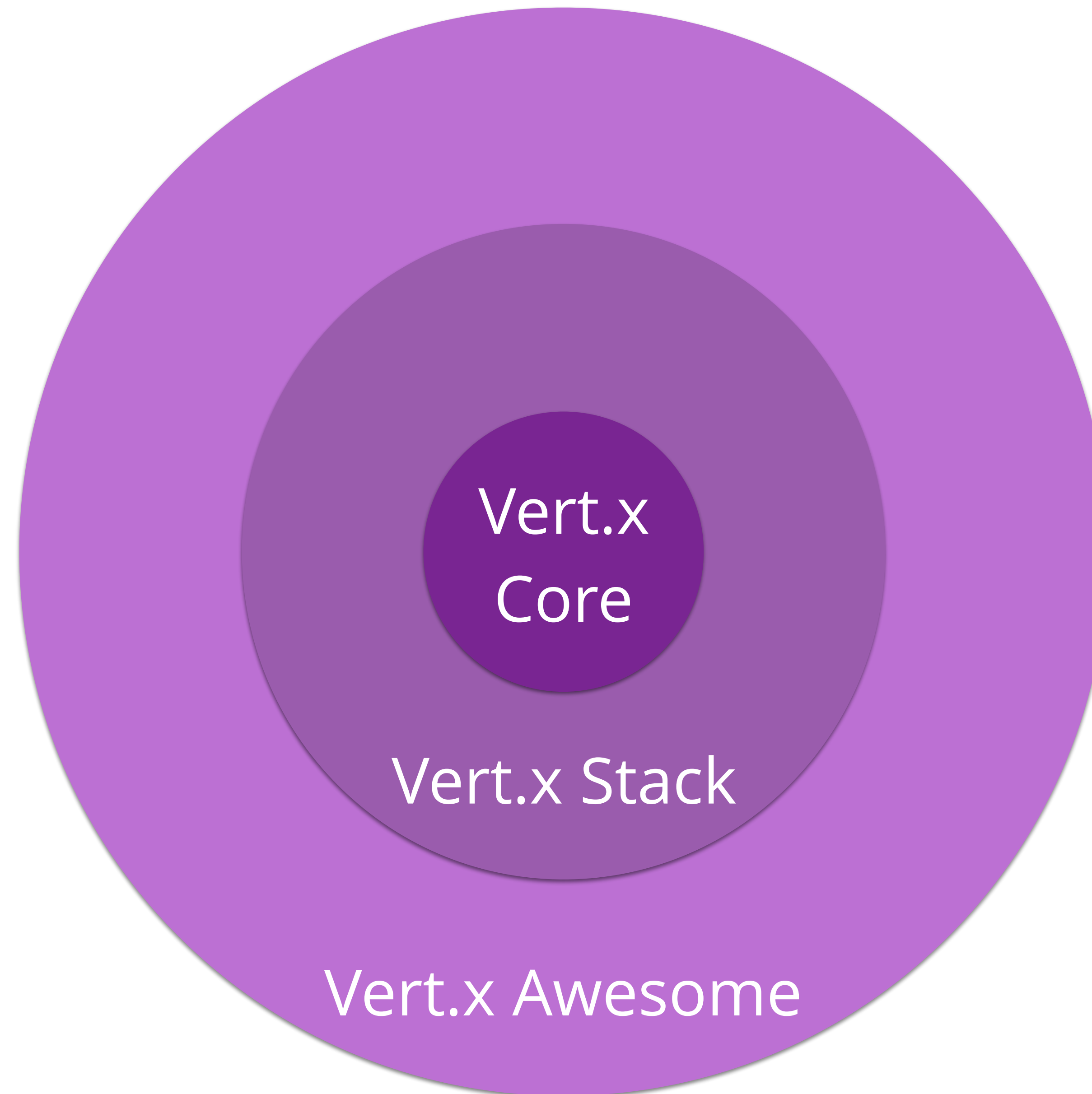
```
void listen(int port, Handler<AsyncResult<HttpServer>> ar)
```



```
Single<HttpServer> rxListen(int port);
```

Demo (RxJava)

Outro



Vert.x
Core

Vert.x Stack

Vert.x Awesome

Geekbang InfoQ

QCon

全球软件开发大会

北京·2019

更多技术干货分享，北京站精彩继续

提前参与，还能享受更多优惠

识别二维码
查看了解更多

2019.qconbeijing.com



极客时间VIP年卡

每天6元, 365天畅看全部技术实战课程

- 20余类硬技能, 培养多岗多能的混合型人才
- 全方位拆解业务实战案例, 快速提升开发效率
- 碎片化时间学习, 不占用大量工作、培训时间





Unified end-to-end reactive model + ecosystem
(not just APIs...)

For *all* kinds of distributed applications
(even the small-scale ones)

Flexible toolkit, not a framework
(your needs, your call)

 *Guide to async programming with Vert.x for Java developers*

<https://goo.gl/AcWW3A>

 *Building Reactive Microservices in Java*

<https://goo.gl/ep6yB9>