



VOXXEDDAYS
SINGAPORE



RED HAT DEVELOPERS

Reactive Microservices with Eclipse Vert.x

@bursutter
vertx.io

<http://developers.redhat.com>

<http://bit.ly/reactivemsa>

Change History

1.0 - JavaOne 2016

2.0 - <http://javaday.org.ua/kyiv/>

2.1 - Added “Be This Tall” and 2 Gene Kim books slides

2.2 - Changed color scheme/art on several slides

2.3 - reordered slides, added links to mono2micro data book and Reactive Tutorial, updated getting started section

2.4 - adding slide for Clement’s book

2.5 - added slide for Balloon Popping demo

java -jar myapp.jar

DropWizard

www.dropwizard.io

JAX-RS API

First to market

DropWizard Metrics

Embeddable
servers:
Jetty



Spring Boot

projects.spring.io/spring-boot

Spring API
(@RestController)

'Starter' POMs:
start.spring.io

Embeddable servers:
Tomcat, Jetty, Undertow



WildFly Swarm

wildfly-swarm.io

Java EE 7 APIs

'Starter' POMs:
wildfly-swarm.io/generator

Embeddable
servers:
WildFly (Undertow)



Vert.x

vertx.io

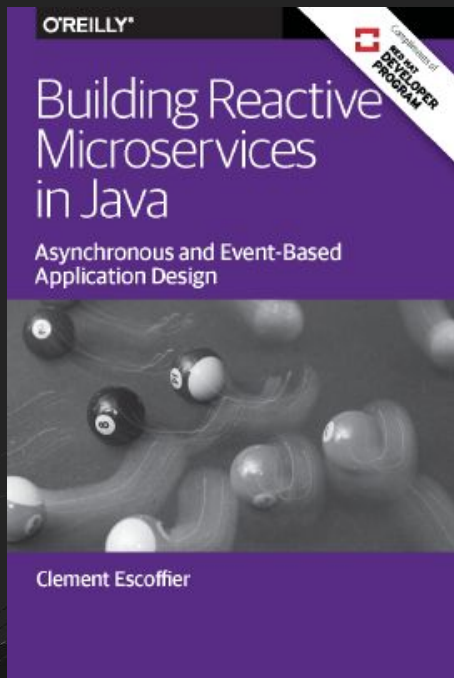
Reactive
Async/non-blocking

vertx run myhttp.java

HTTP, HTTP/2, TCP,
UDP, Websockets,
etc. (out of the box)



<https://developers.redhat.com>



Download Free ebook

Developer Evolution :-)

70s



COBOL
JCL
WFL

80s



C/C++
4GLs
CASE
Netware
RDBMS
SQL
Unix

90s



HTTP/HTML
CGI
GET/POST
Cookies
Java
Servlet
EJB
Windows NT
Solaris/AIX
OOP

00s



MVC - Struts
DI - Spring
ORM - Hibernate
XML
WS-*
JSF
AJAX
Agile
Automated Testing
CI
SVN
Linux

10s

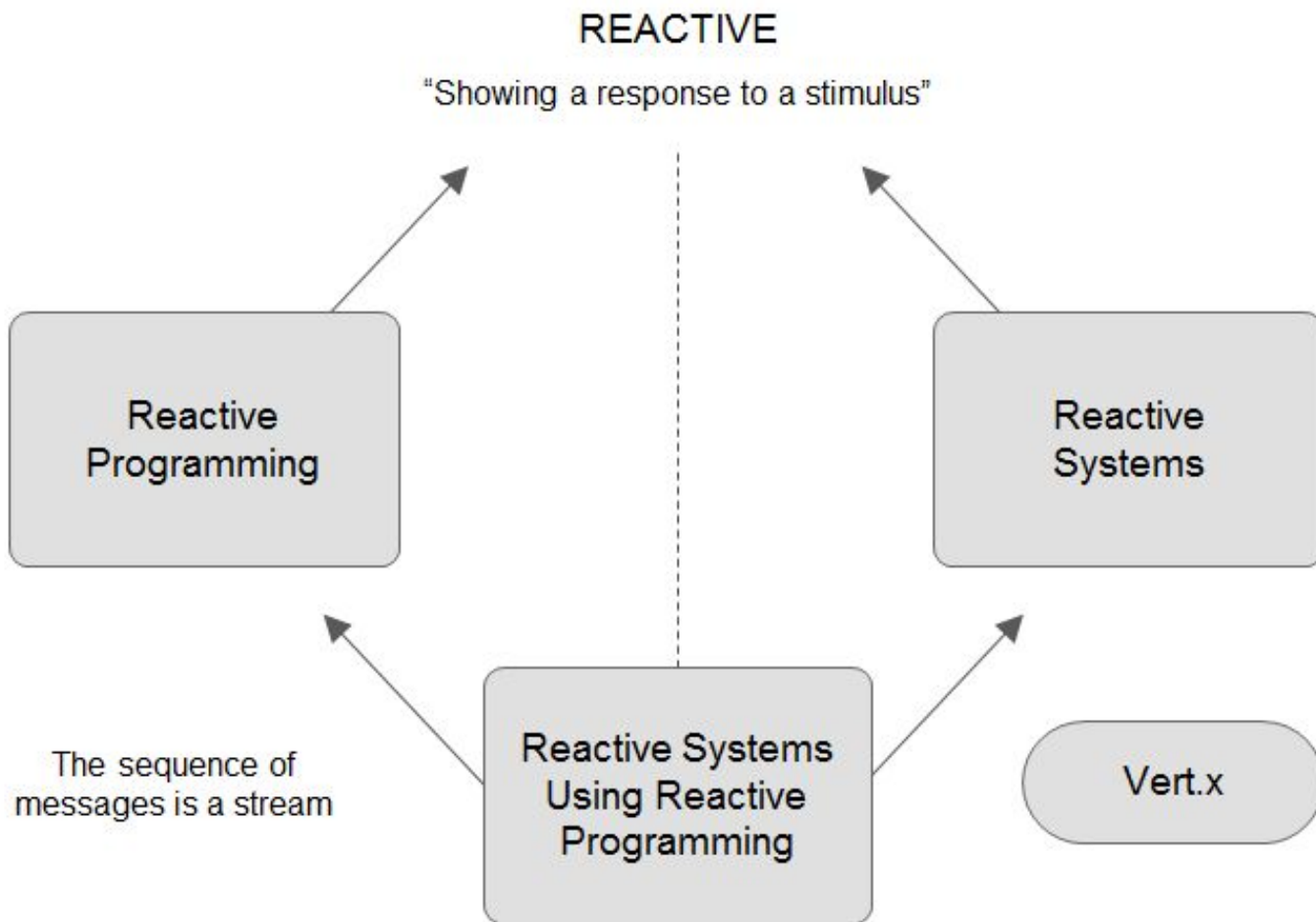


HTML5
Mobile
iOS/Android
Maven/Gradle
Git
Node.js
MongoDB/Redis
Hadoop/Spark
*-aaS/Cloud
Reactive
Functional



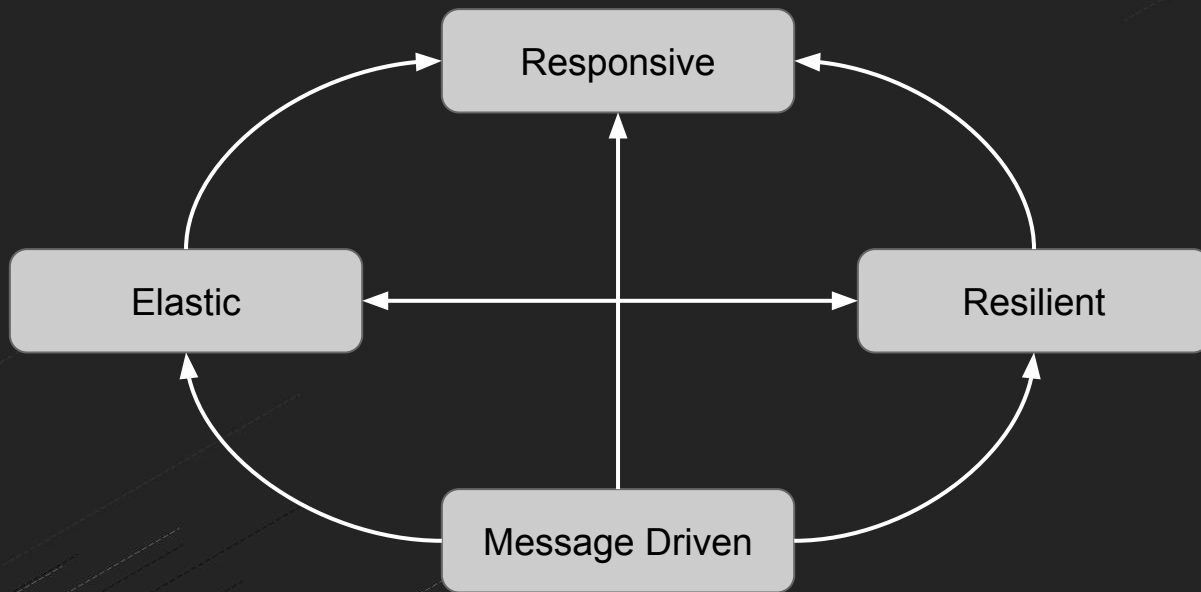
Mythical
Beast

Reactive



Reactive Manifesto

Reactive Systems (not Programming)



**Reactive programming is programming with
asynchronous data streams.**

<https://gist.github.com/staltz/868e7e9bc2a7b8c1f754>

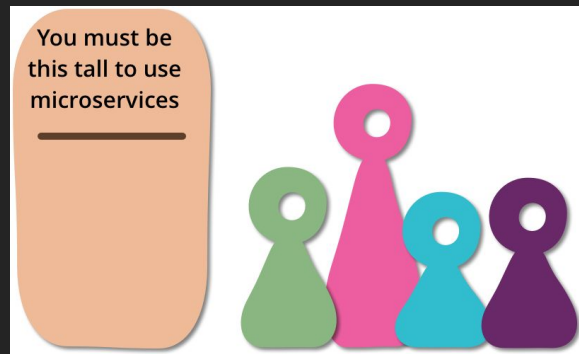
Responsive Demo

<https://github.com/redhat-helloworld-msa/goodbye>

Microservices

You Must Be This Tall

1. Self-Service, on-demand, elastic infrastructure as code
(how many days/weeks to provision a new VM?)
2. Dev vs Ops
(who is on the pager for production app outage?)
3. Automation
(phoenix vs snowflake?)
4. CI & CD
5. Deployment Pipeline



<http://martinfowler.com/bliki/MicroservicePrerequisites.html>

The Phoenix Project

A Novel About IT, DevOps, and Helping Your Business Win

Gene Kim, Kevin Behr and George Spafford

The
DevOps Handbook

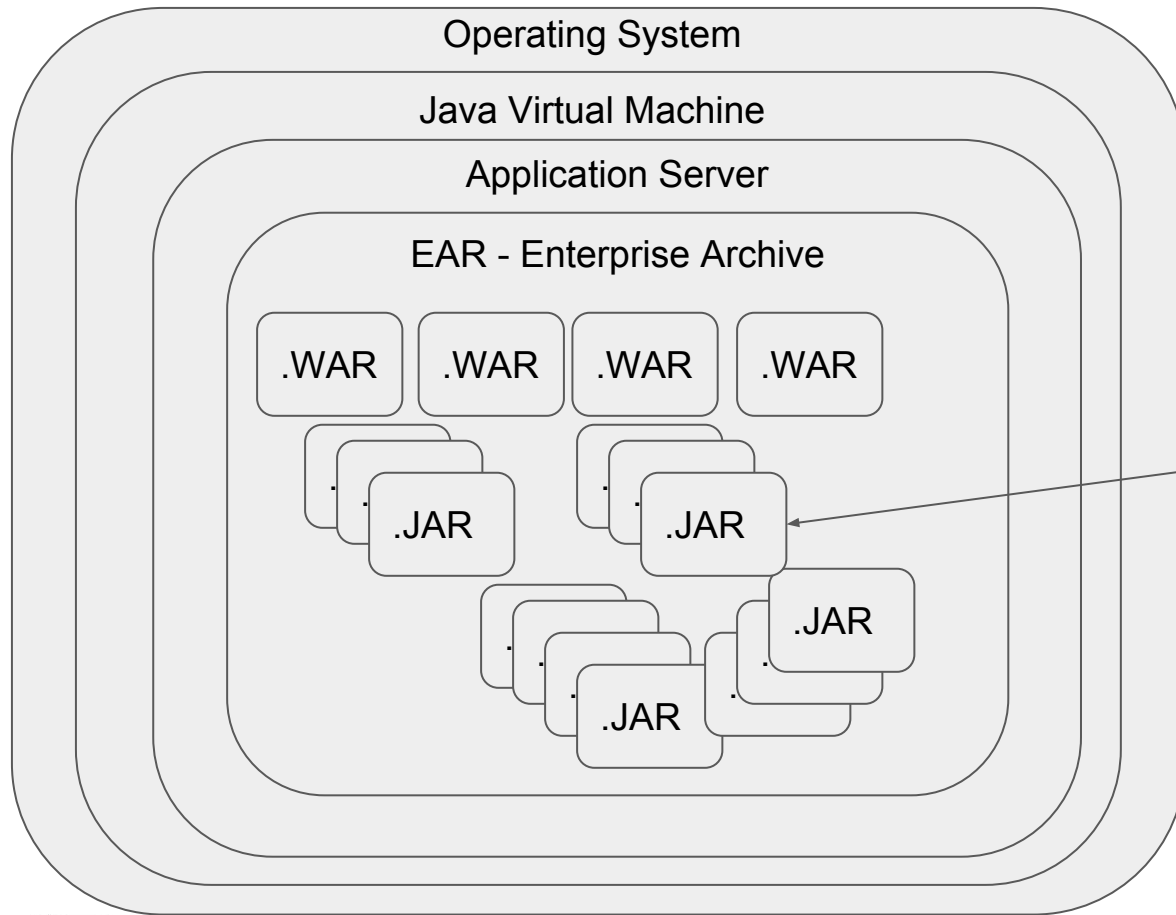
HOW TO CREATE WORLD-CLASS
AGILITY, RELIABILITY, & SECURITY
IN TECHNOLOGY ORGANIZATIONS



GENE KIM,
JEZ HUMBLE,
PATRICK DEBOIS,
& JOHN WILLIS

FOREWORD BY JOHN ALLSPAUGH

TAKE THE DORA DEVOPS X-RAY ASSESSMENT AND SEE WHERE YOU STAND.

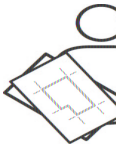




Programmers
(18)



Business
Analysts
(4)



Project
Managers
(2)



Quality
Assurance
(6)



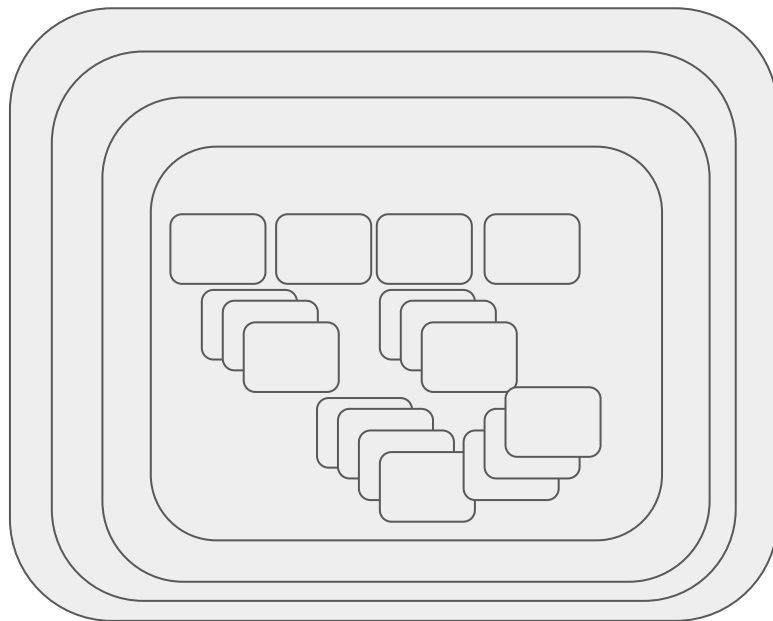
Compliance
(2)

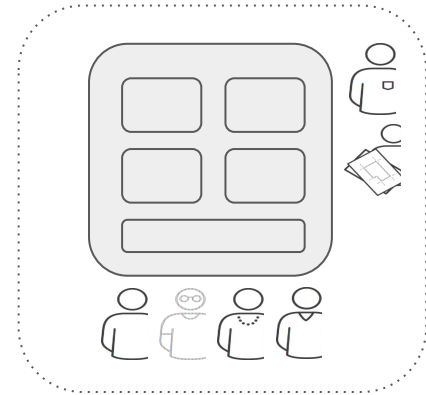
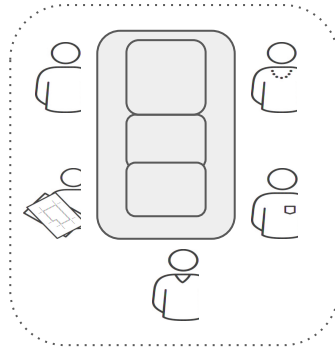
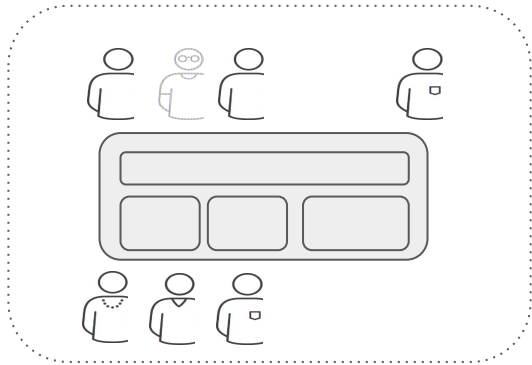


Operators
(6)

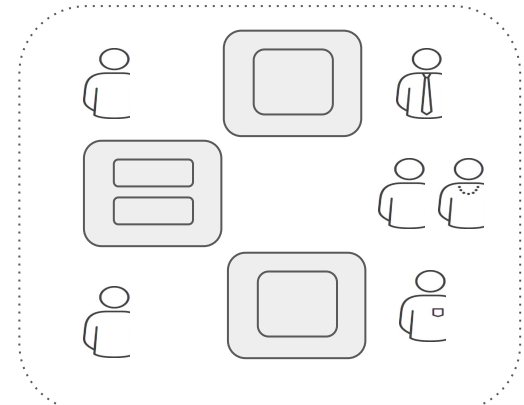
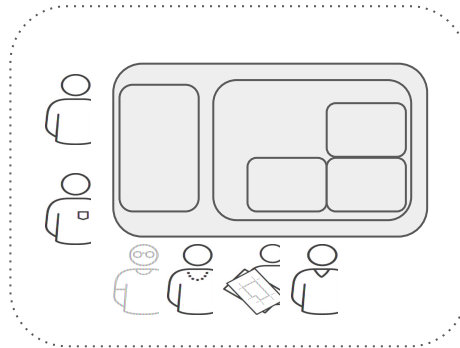
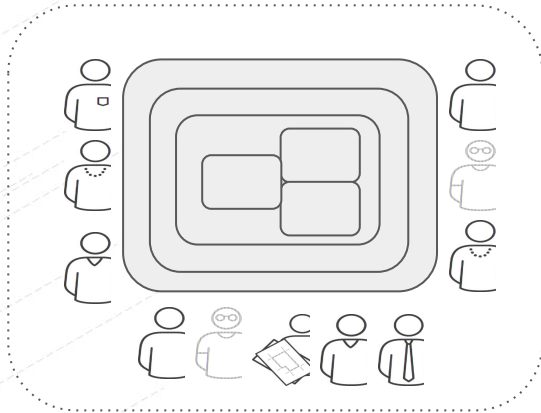


DBAs
(3)



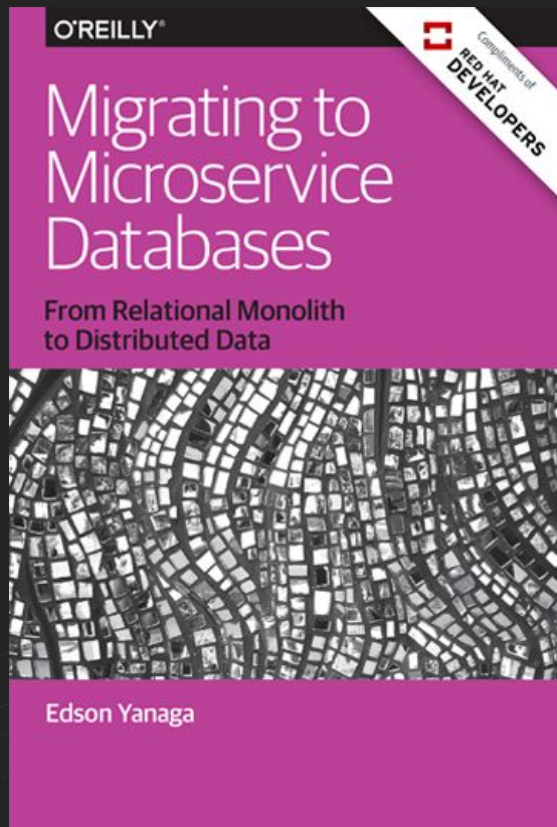


2-Pizza Teams You Build It, You Own it



Microservice Principles/Characteristics

1. Deployment **Independence**: updates to an individual microservice have no negative impact to any other component of the system. Optimized for **Replacement**.
2. Organized around **business** capabilities
3. **Products** not Projects
4. **API-Focused**
5. **Smart** endpoints and dumb pipes
6. Decentralized Governance
7. Decentralized Data Management
8. Infrastructure Automation (infrastructure as code)
9. Design for failure
10. Evolutionary Design

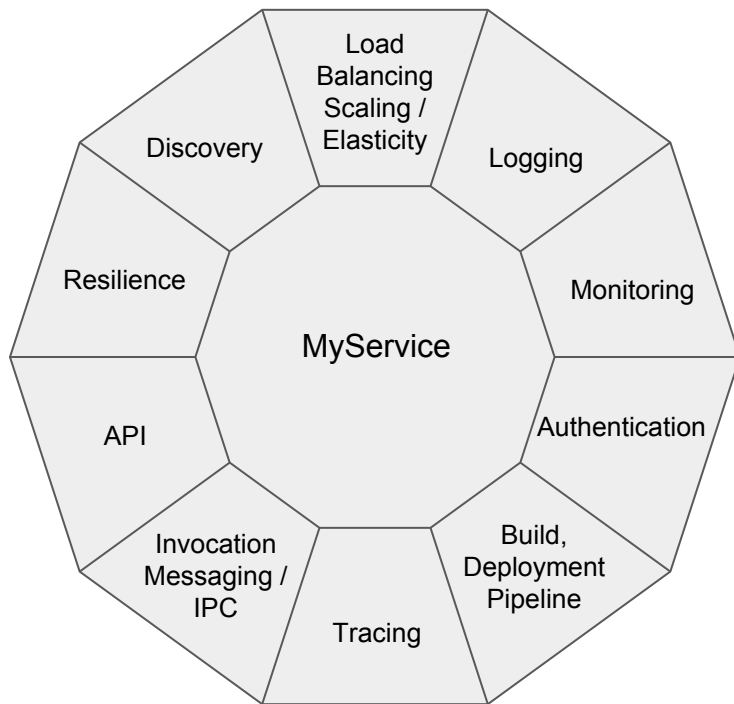


Decentralized **Data** Management

What?

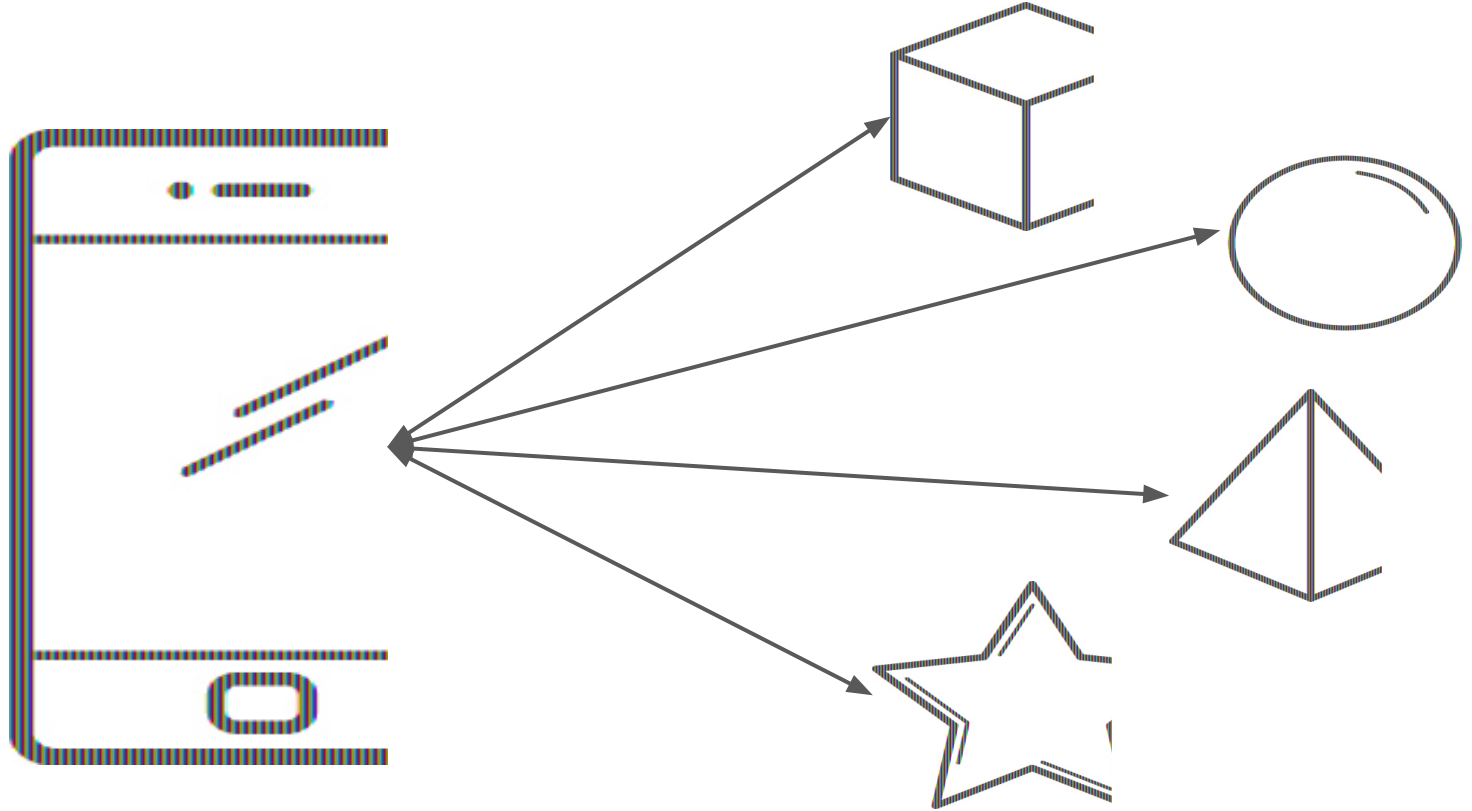
<http://bit.ly/mono2microdb>

Microservice Concerns

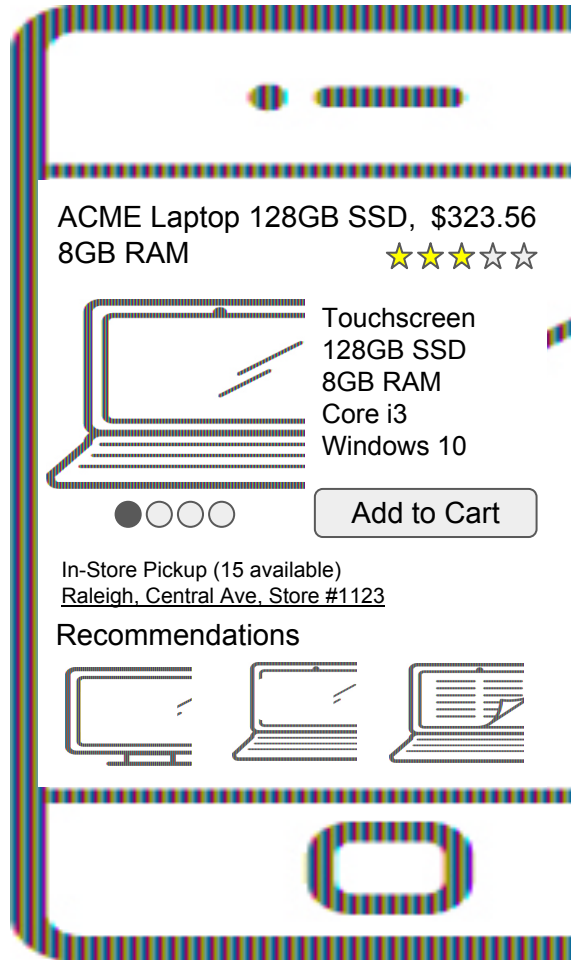


Microservice Patterns

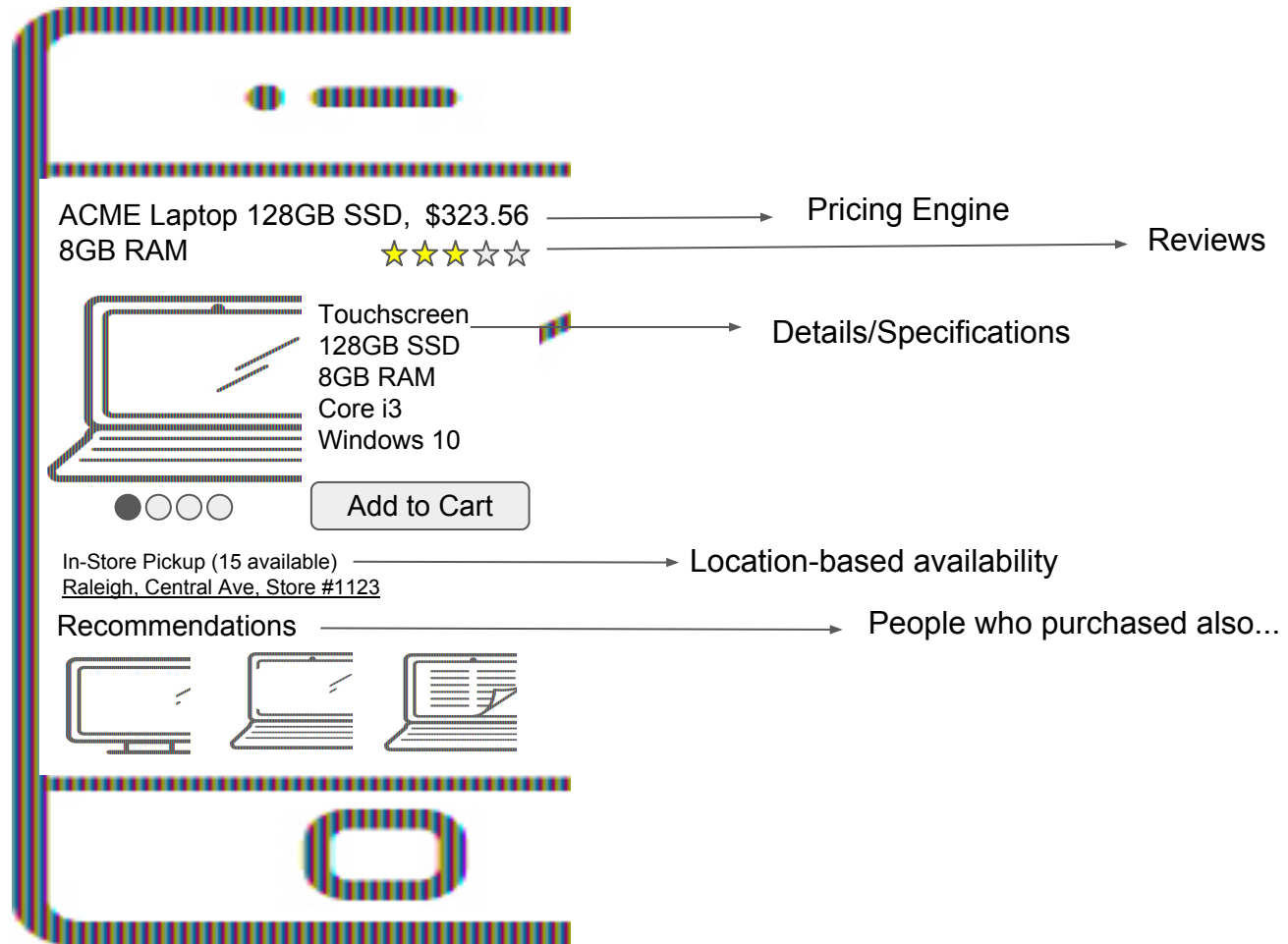
Browser



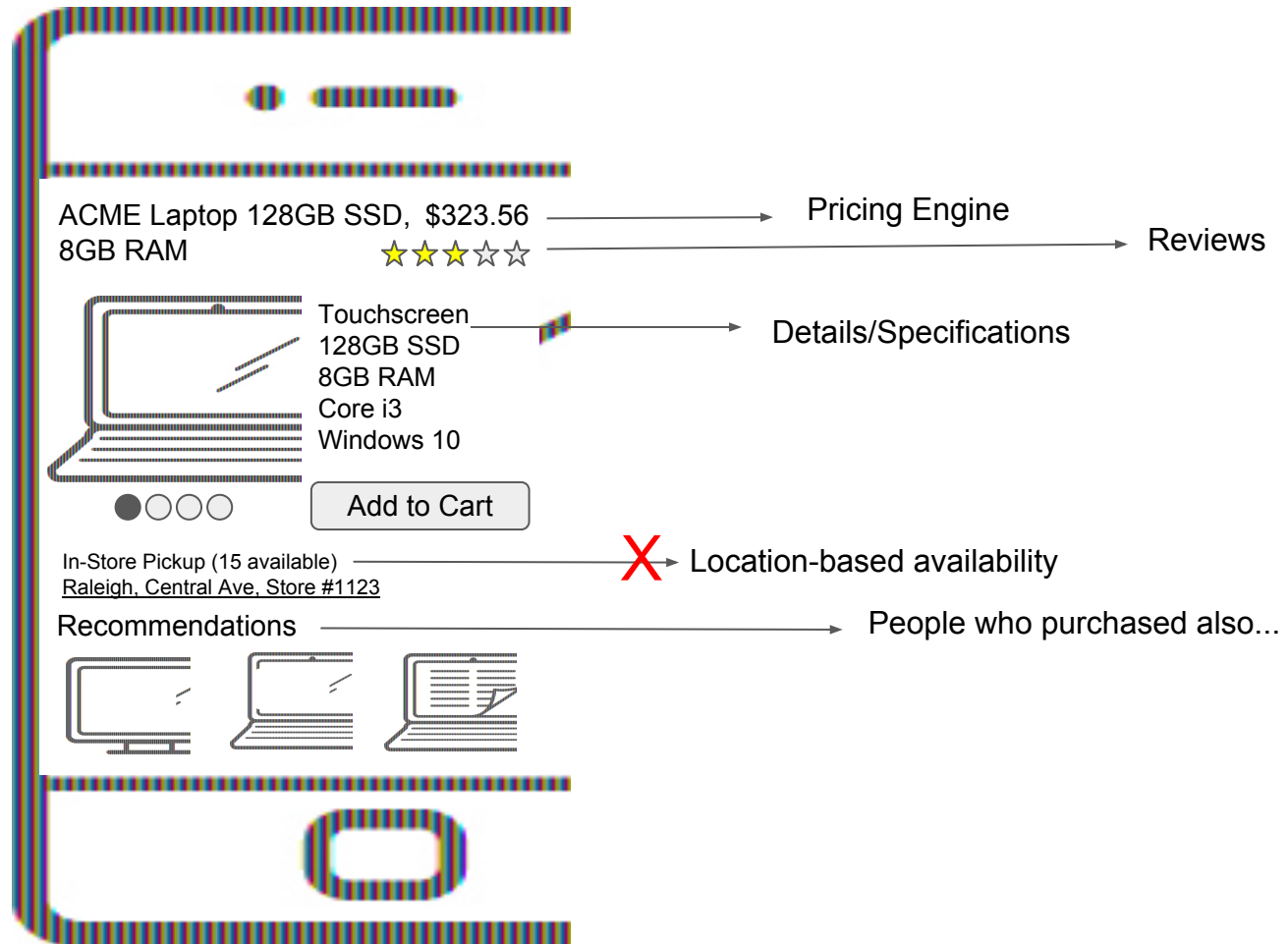
Example



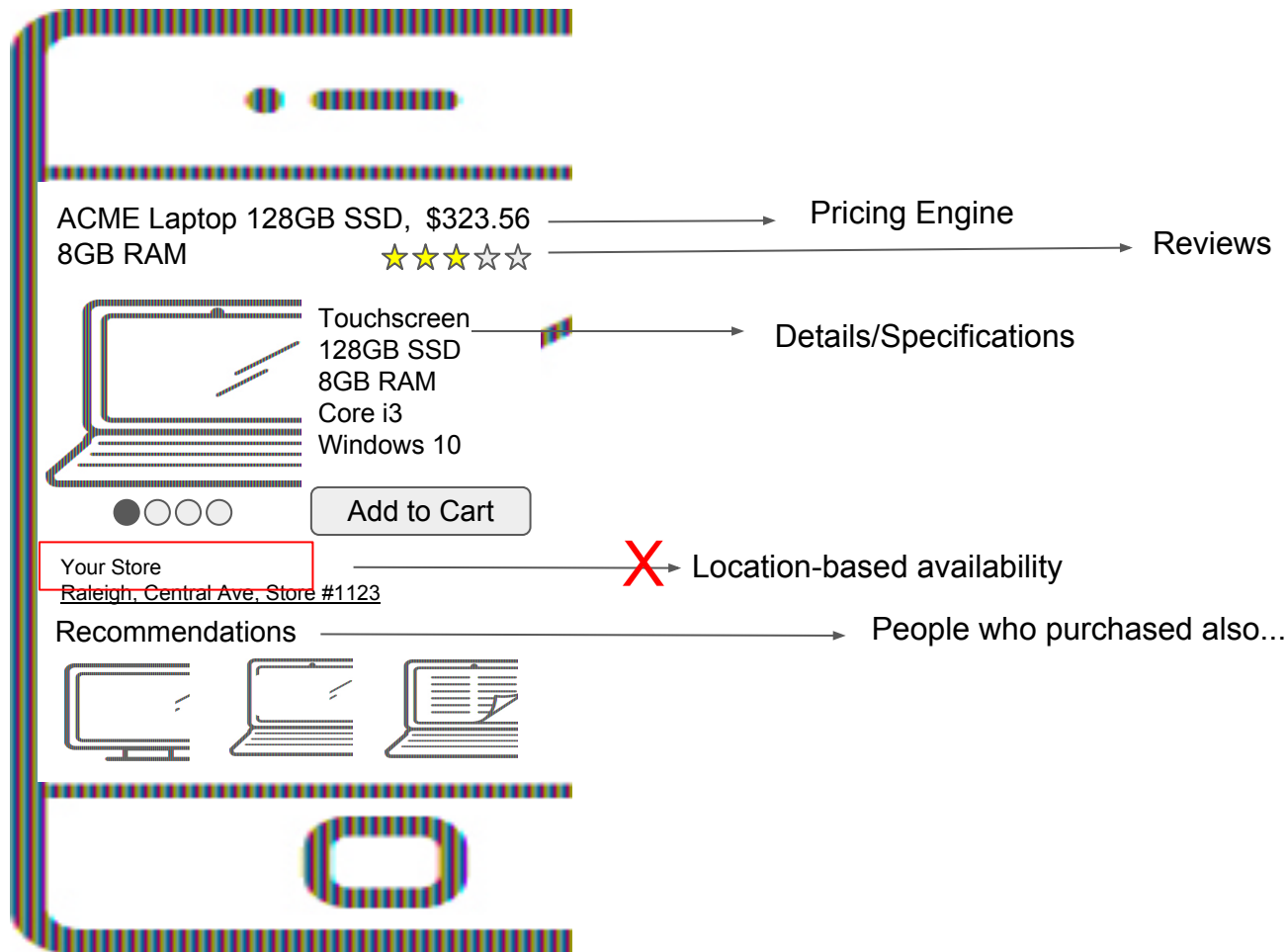
Example



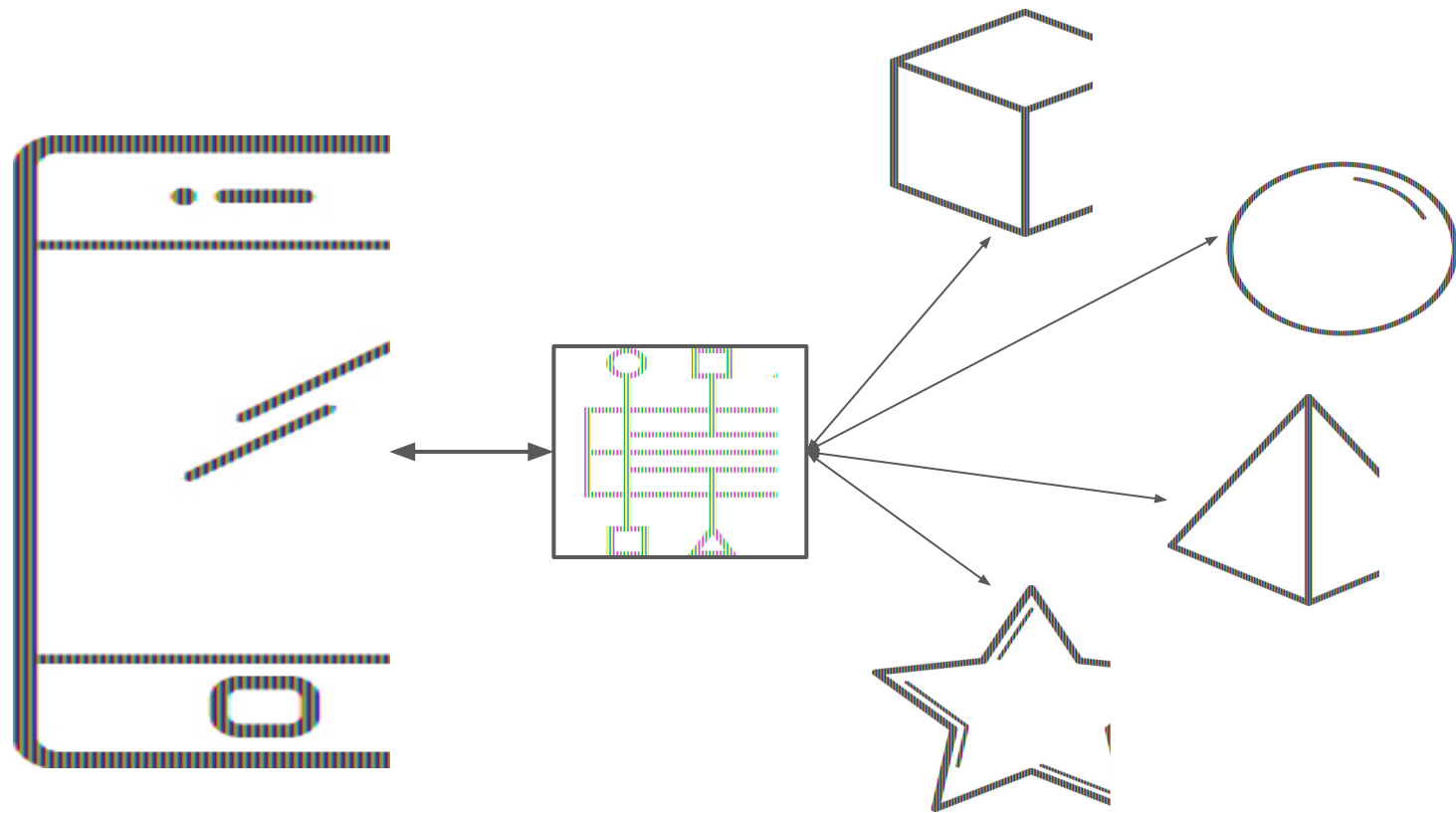
Example (Fail)



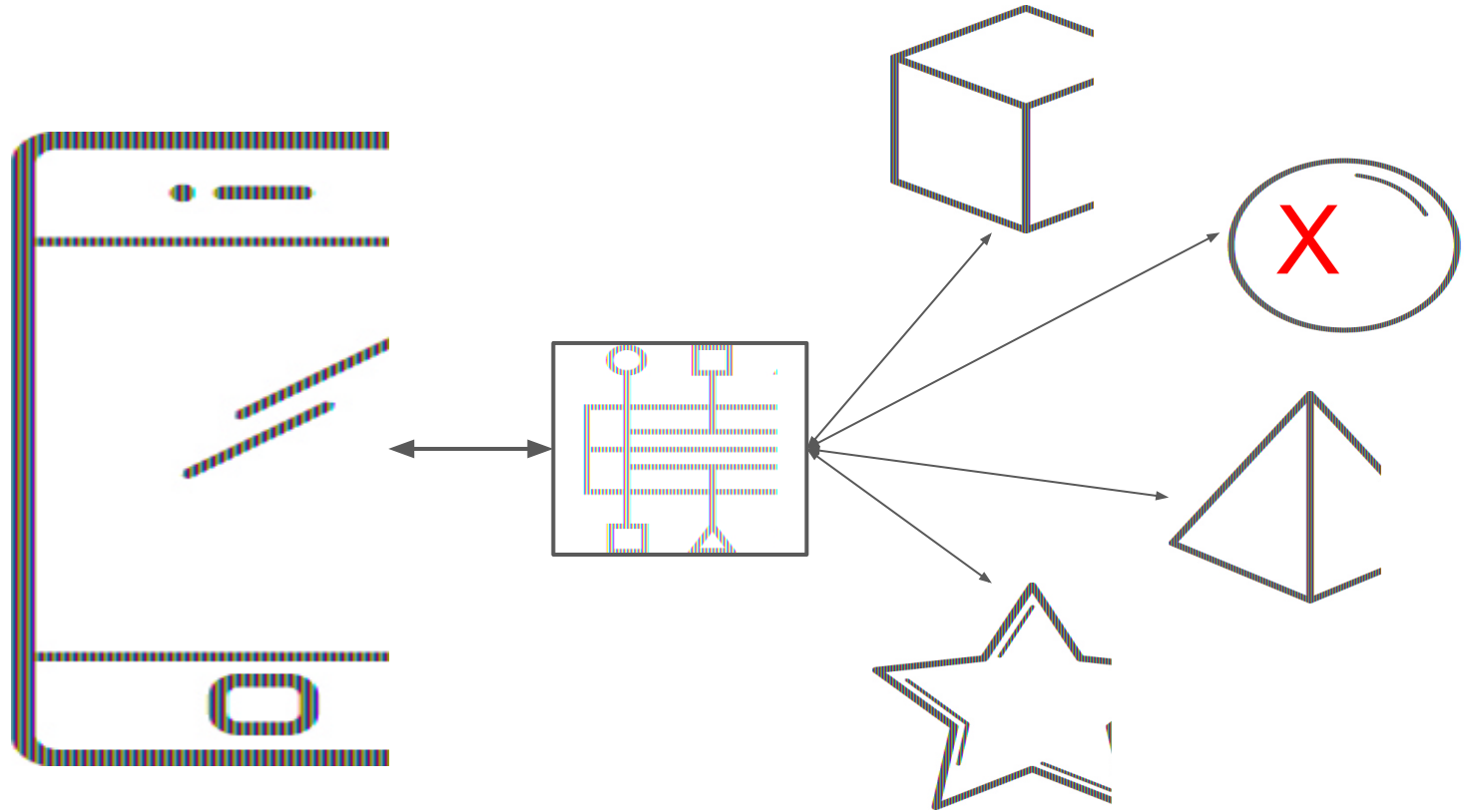
Example (Fallback)



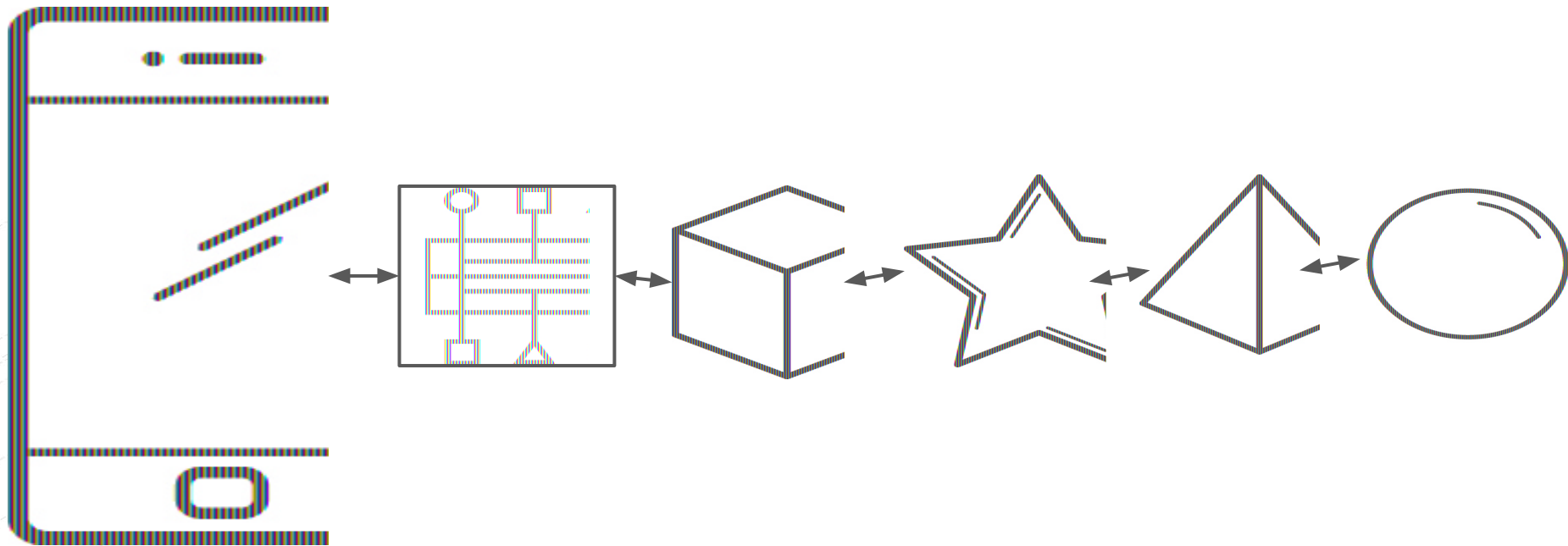
API “Gateway”



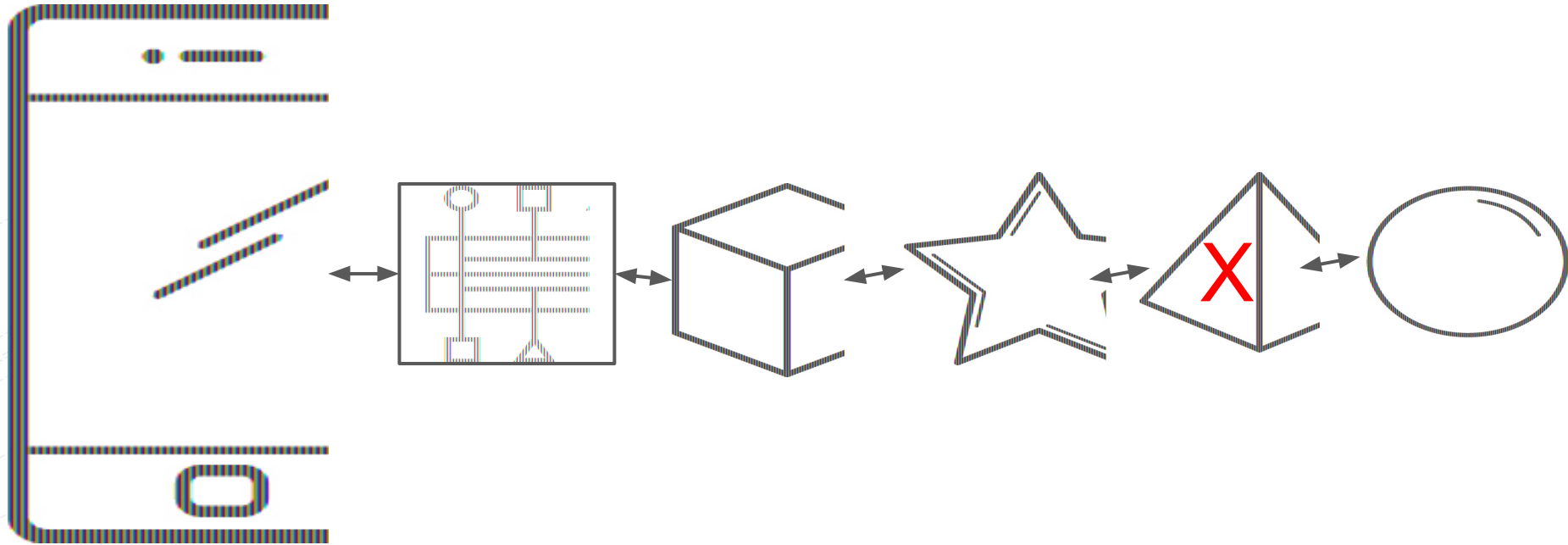
API “Gateway” (Fail)



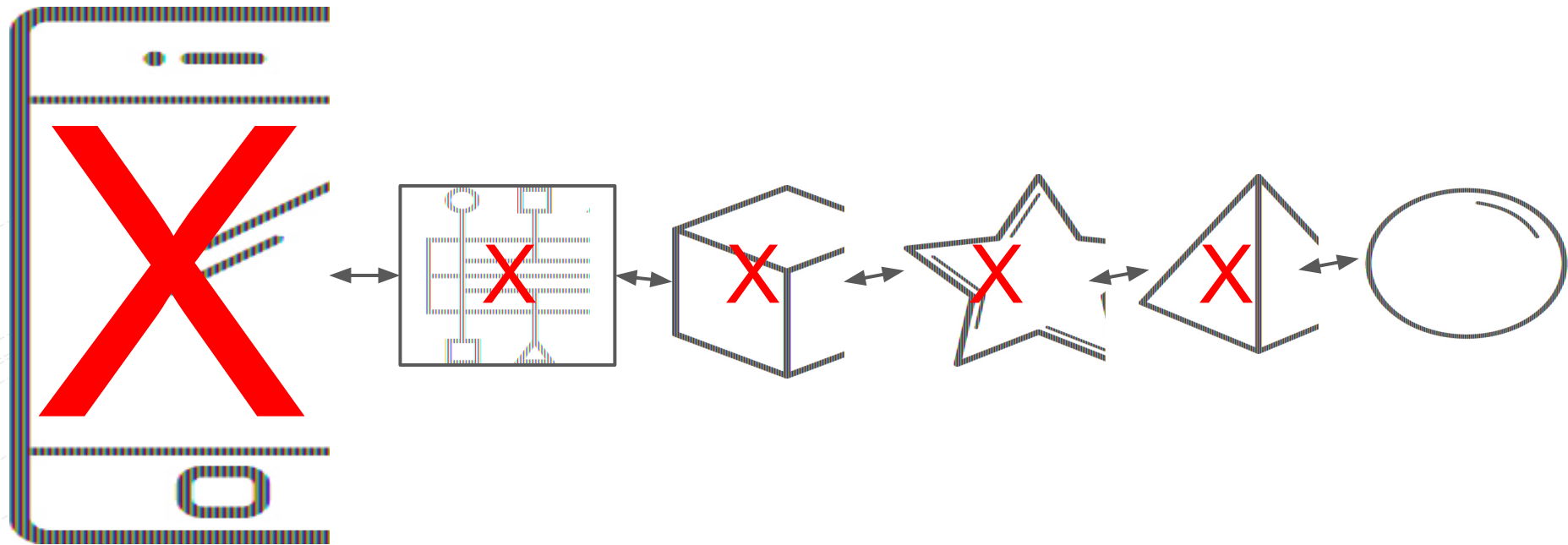
Chaining



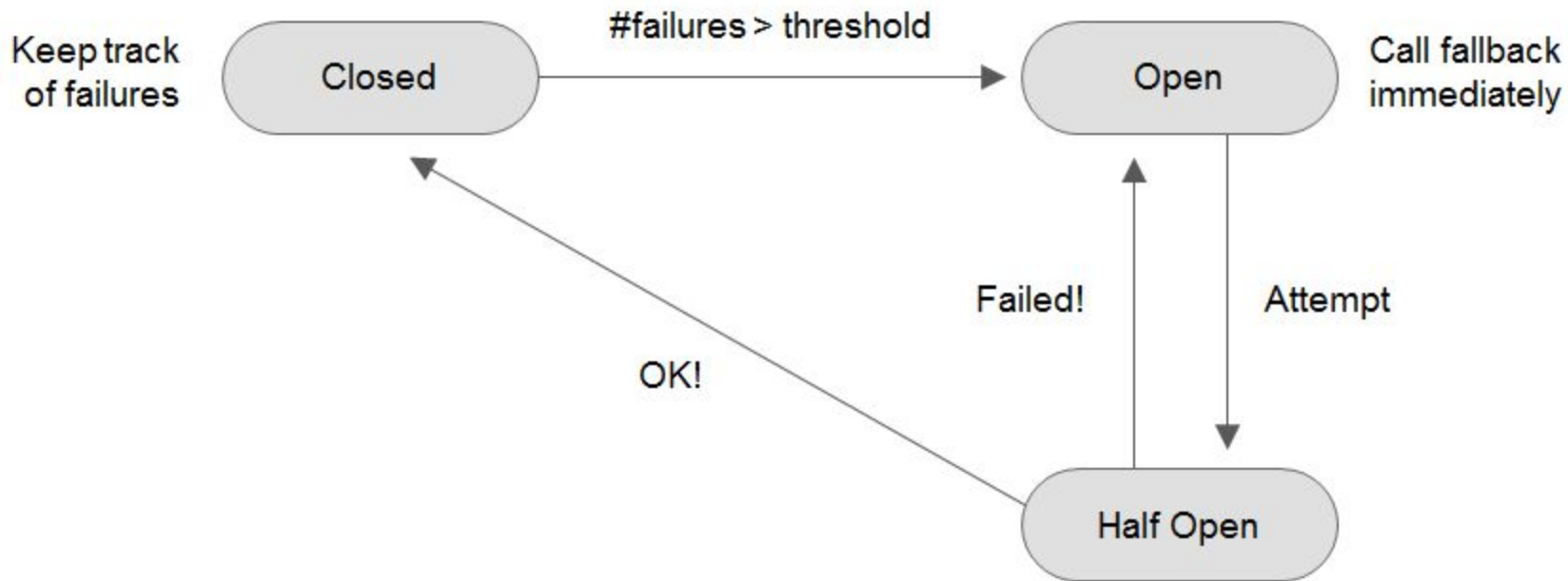
Chaining (Fail)



Chaining (Cascading Fail)



Circuit Breaker



Demo

bit.ly/msa-instructions

<https://github.com/burrsutter/kube4docker>

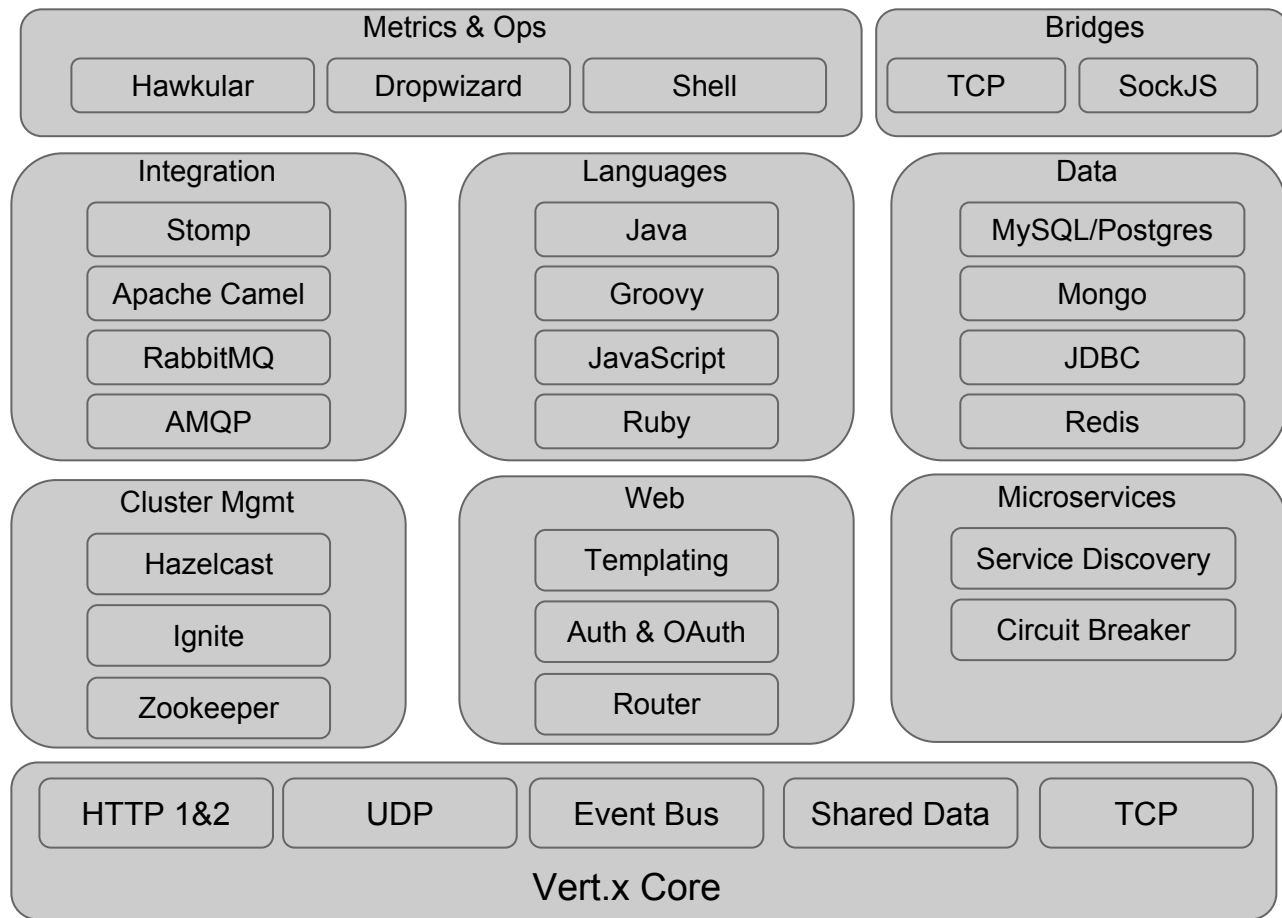
Videos:

https://www.youtube.com/watch?v=SPATMHP-xw8&list=PLuWlr4oKSRUYjjwDOaZOX-54X4_OIUeS



Vertx.io—an Eclipse project

Vert.x is a toolkit
for building reactive
applications on the JVM.



Vert.x

(Un-Opinionated) Toolkit for reactive, async applications

Every user has their own Vert.x way

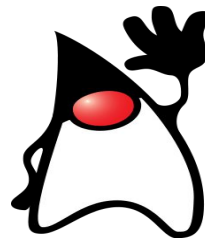
Low-latency microservices that interact using the EventBus

Built in elasticity and resiliency

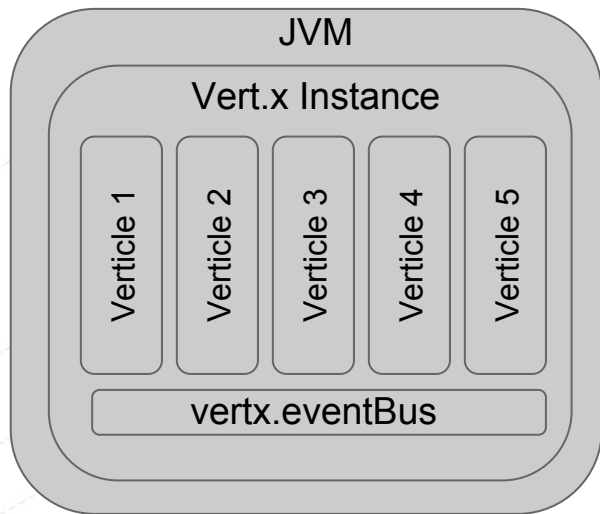
Polyglot: Java, JavaScript, Groovy, Ruby, Ceylon

vertx.io

<http://vertx.io/blog/my-first-vert-x-3-application/>



Vert.x Verticle

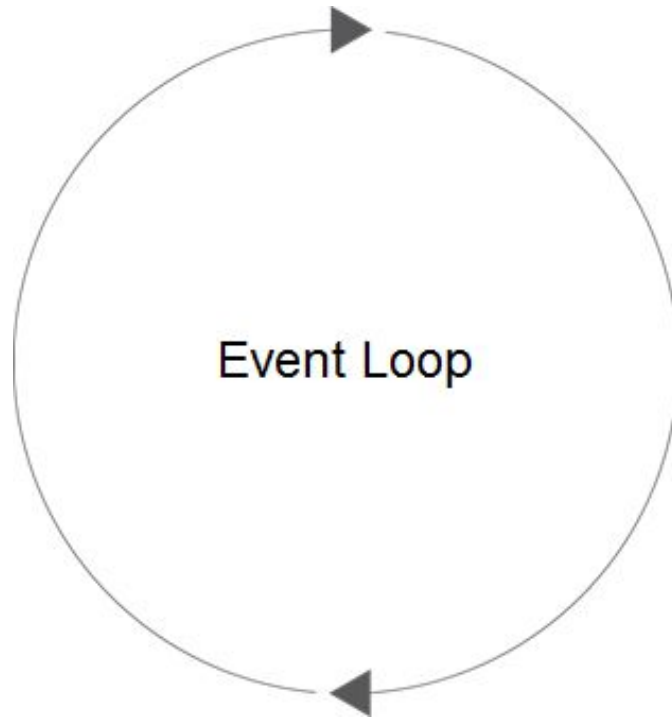


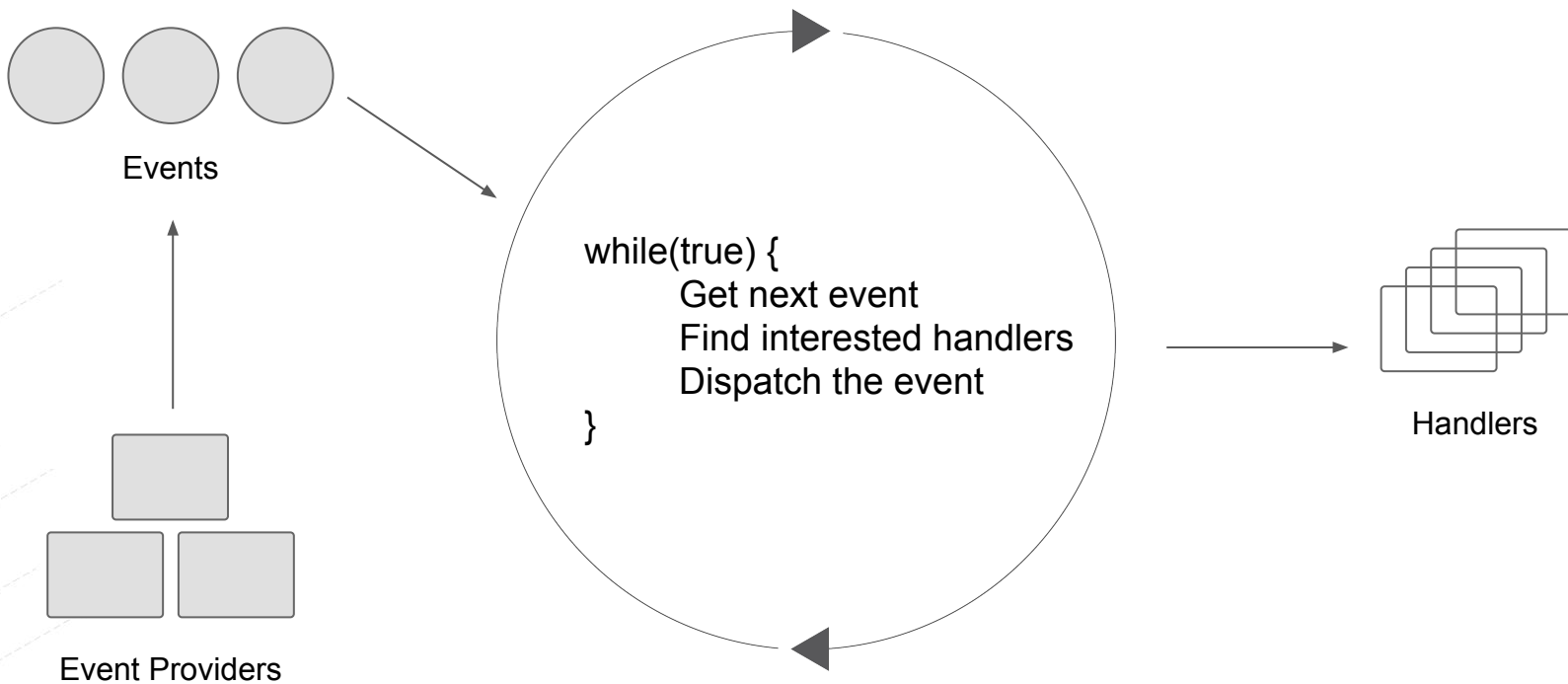
A Verticle is the programmable unit – a component

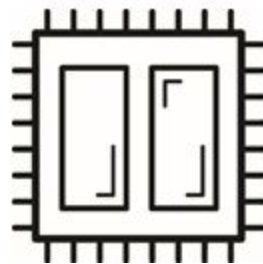
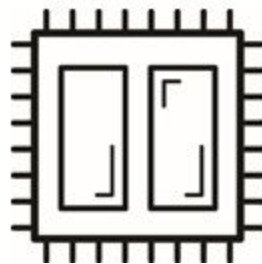
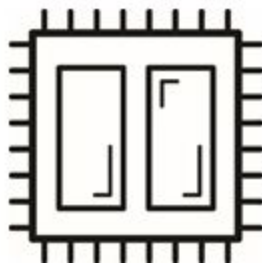
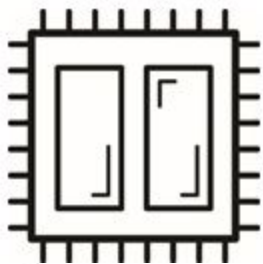
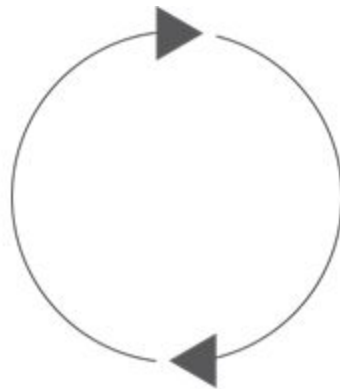
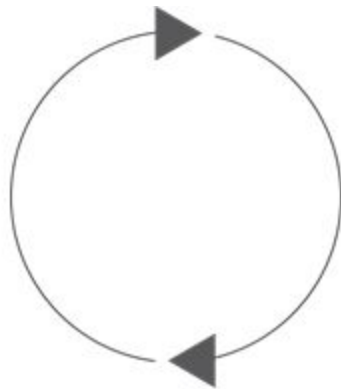
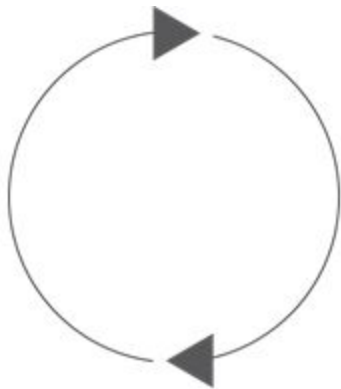
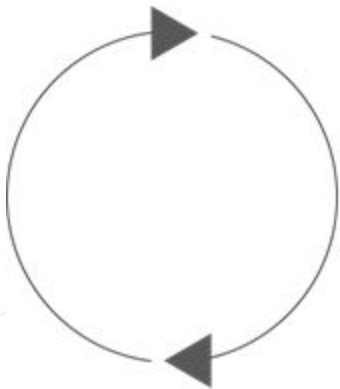
A Verticle is always executed on the same thread

A single thread may execute several verticles – event loop

A Verticle Instance normally starts 1 thread/event-loop per core







Vert.x HTTP Verticle

```
import io.vertx.core.AbstractVerticle;

public class MainVerticle extends AbstractVerticle {

    @Override
    public void start() {
        vertx.createHttpServer()
            .requestHandler(req ->
                req.response().end("Hello Vert.x v3!"))
            .listen(8080);
    }
}
```

Super-Fast Getting Started: Step 1

```
mkdir stuff
```

```
cd stuff
```

```
mvn io.fabric8:vertx-maven-plugin:1.0.13:setup  
-DvertxVersion=3.4.2 -Ddependencies=web
```

```
mvn compile vertx:run
```

Super-Fast Getting Started: Step 2

```
package stuff;

import io.vertx.core.AbstractVerticle;
import io.vertx.ext.web.Router;

public class MainVerticle extends AbstractVerticle {

    @Override
    public void start() {
        Router router = Router.router(vertx);
        router.get("/").handler(request -> {
            String hostname = System.getenv().getOrDefault("HOSTNAME", "unknown");
            request.response().end("Hello " + new java.util.Date() + " on " + hostname);
        });
        vertx.createHttpServer()
            .requestHandler(router::accept)
            .listen(8080);
    } // start
} // MainVerticle
```

Super-Fast Getting Started: Step 3

```
localhost:8080 (mvn compile vertx:run will hot redeploy)
```

Or

```
mvn clean compile package
```

```
java -jar target/demo-1.0-SNAPSHOT.jar
```

Now, launch into Kubernetes/OpenShift

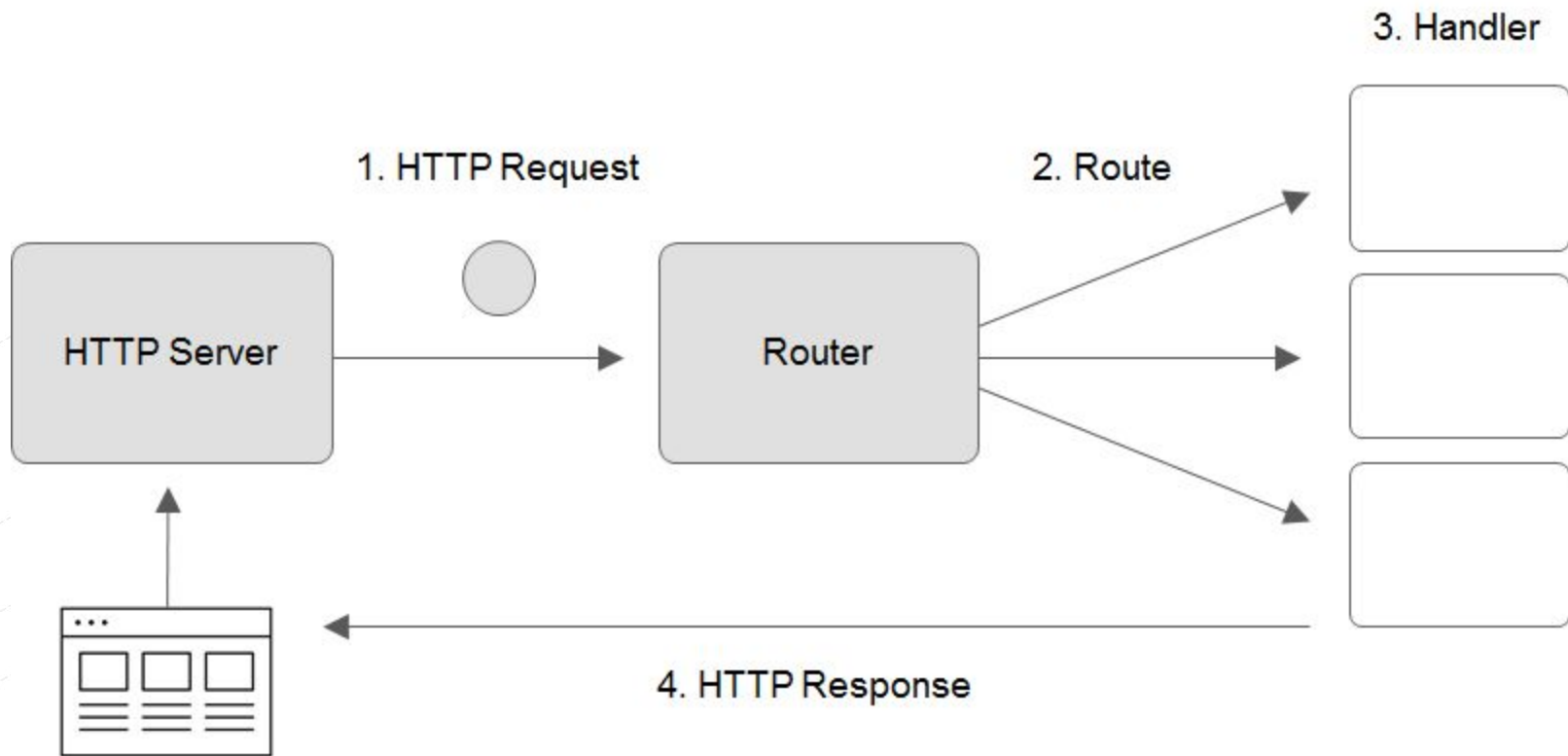
```
mvn io.fabric8:fabric8-maven-plugin:3.5.31:setup
```

```
oc login -u username -p userpassword https://openshift.acme.com:8443
```

```
oc new-project demo
```

```
mvn fabric8:deploy
```

Web Apps & REST

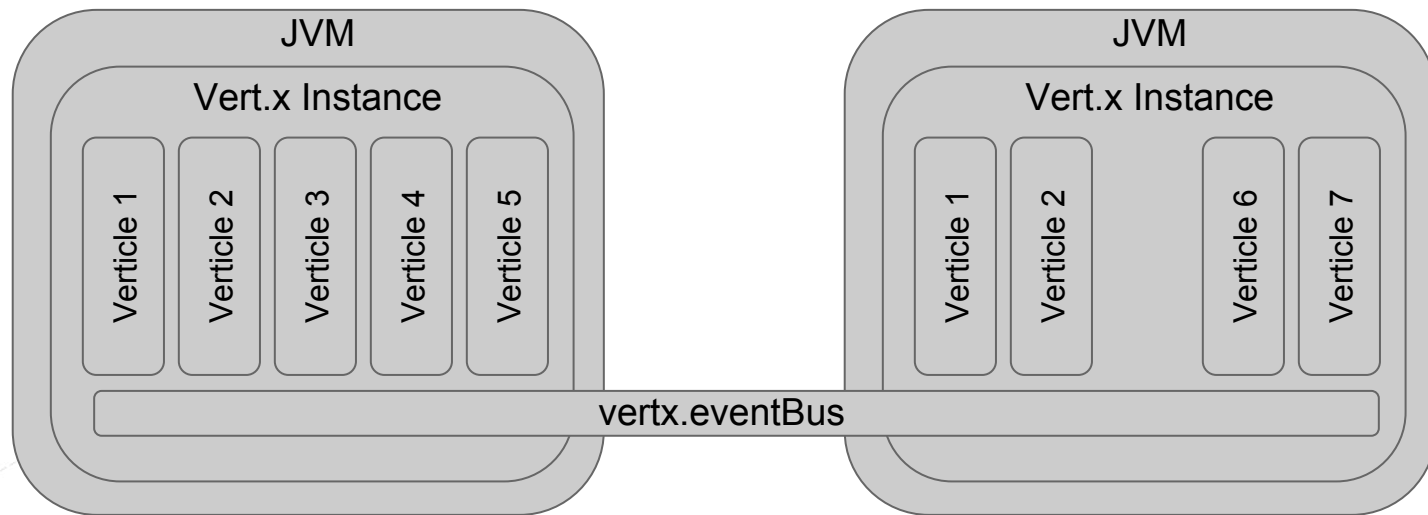


Vert.x HTTP Router

```
router.get("/api/whiskies").handler(this::getAll);
router.get("/api/whiskies/:id").handler(this::getOne);
router.post("/api/whiskies").handler(this::addOne);
router.delete("/api/whiskies/:id").handler(this::deleteOne);
router.put("/api/whiskies/:id").handler(this::updateOne);
// Serve static resources from the /assets directory
router.route("/assets/*").handler(StaticHandler.create("assets"));
```

```
vertx
    .createHttpServer()
    .requestHandler(router::accept)
    .listen(8080, result -> {
        if (result.succeeded()) {
            future.complete();
        } else {
            // could be something else is running on our port
            future.fail(result.cause());
        }
    });
```

Vert.x EventBus



Demo

https://github.com/burrsutter/vertx3_introduction

HTTP getNow() Circuit Breaker

```
CircuitBreakerOptions options = new  
    CircuitBreakerOptions()  
        .setMaxFailures(5)  
        .setTimeout(5000)  
        .setFallbackOnFailure(true);
```

```
CircuitBreaker breaker =  
    CircuitBreaker.create("my-circuit-breaker", vertx,  
        options)  
        .openHandler(v -> {  
            System.out.println("Circuit opened");  
        }).closeHandler(v -> {  
            System.out.println("Circuit closed");  
        });
```

```
Future<String> result = breaker.executeWithFallback(future -> {  
    vertx.createHttpClient().getNow(8080, "localhost", "/",  
        response -> {  
            if (response.statusCode() != 200) {  
                future.fail("HTTP error");  
            } else {  
                response.exceptionHandler(future::fail).bodyHandler(buffer  
                    -> {  
                        future.complete(buffer.toString());  
                    });  
            }  
        });  
    }, v -> {  
        // Executed when the circuit is opened  
        return "Hello (fallback)";  
    });
```

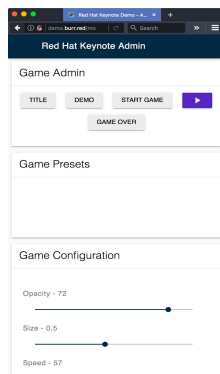
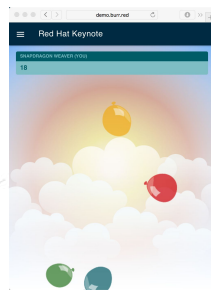
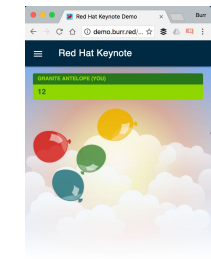
https://github.com/burrsutter/vertx3_introduction/blob/master/7_httpinvoker/Client.java#L13

Demos

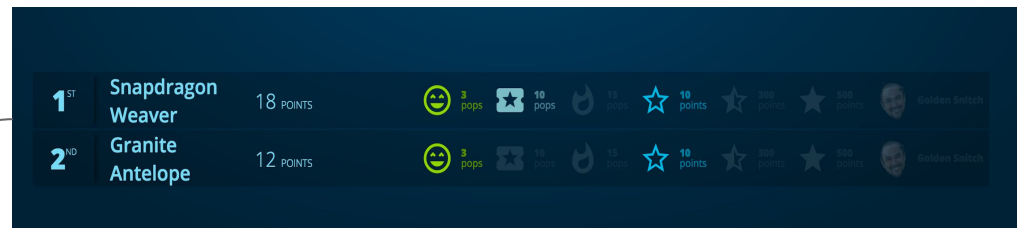
<https://github.com/redhat-reactive-msa>

https://github.com/burrsutter/reactive_tutorial_jfokus17

Recording: <https://youtu.be/ooA6FmTL4Dk?t=1739>

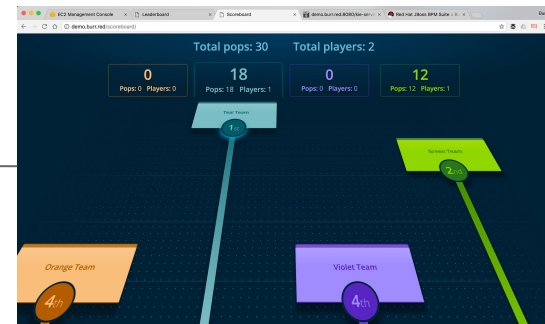
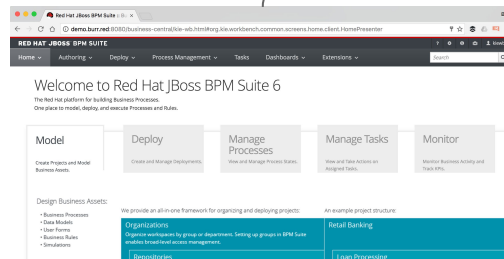


vertex-game-server



vertex-achievement-server

Java EE on JBoss with Drools+jBPM



Balloon Repos

Email me at burr@redhat.com for setup

<https://github.com/burrsutter/vertx-game-server>

<https://github.com/burrsutter/score-server>

<https://github.com/burrsutter/vertx-achievement-service>

<https://github.com/burrsutter/mobile-app>

<https://github.com/burrsutter/mobile-app-admin>

<https://github.com/burrsutter/leaderboard>

<https://github.com/burrsutter/scoreboard>



RED HAT DEVELOPERS

Reactive Microservices with Eclipse Vert.x

@burrsutter

<http://developers.redhat.com>
<http://bit.ly/reactivemsa>

@burrsutter