AMSTERDAM 11-12 MAY 2016

{codemotion}

{ Building microservices with Vert.x

Bert Jan Schrijver

@bjschrijver

Let’s meet

Bert Jan Schrijver

jpoint

MALMBERG
a Sanoma company

.nl.
jug

@bjschrijver

Outline

- Introduction
- Why Vert.x?
- Basic demo
- Deep dive
- Advanced demo
- Microservices with Vert.x
- Vert.x in practice

Introduction

Malmberg is an educational publisher in the Netherlands. Malmberg is building modern, rich and scalable e-learning applications using Java 8, Vert.x, AngularJS and MongoDB, running on Amazon cloud services.

{codemotion}

{ Vert.x basics

@bjschrijver

Vert.x: the basics

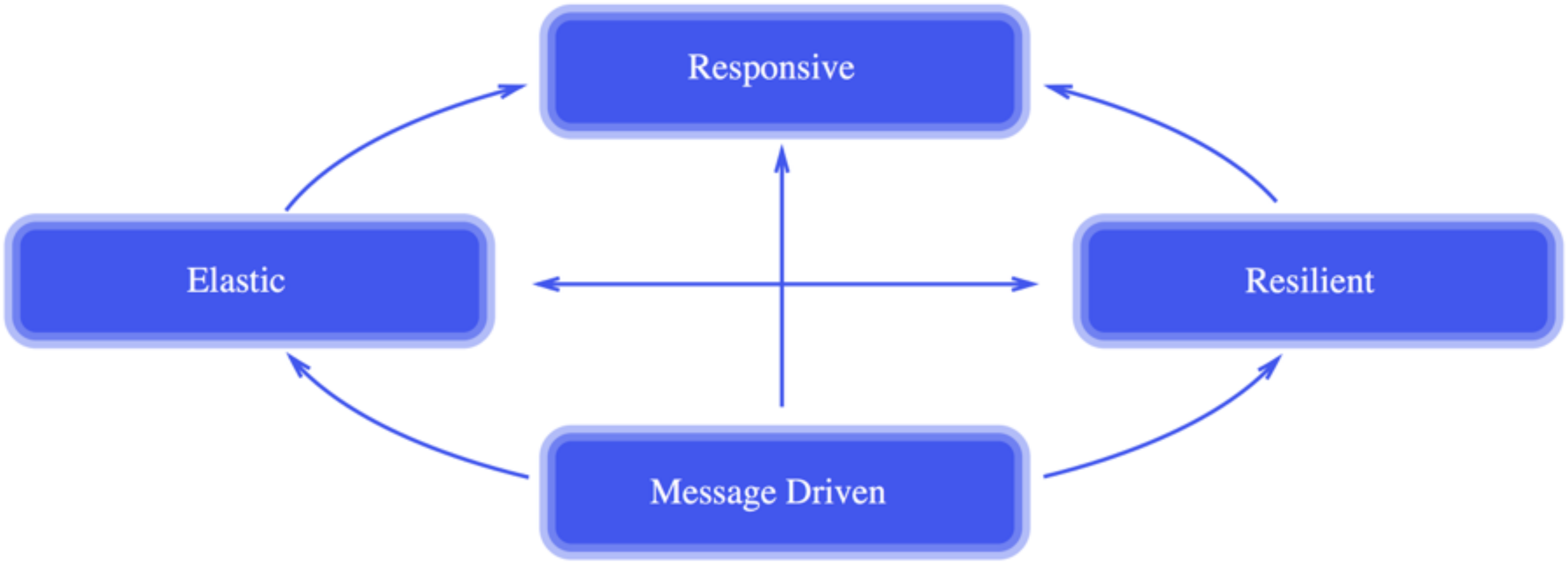
- Toolkit for building reactive applications on the JVM
- General purpose application framework
- Swiss army knife for building modern and scalable web apps
- Event-driven, non-blocking
- Polyglot
- Lightweight and fast
- Fun to work with!

VERT.X

Why did we choose Vert.x?

- We love open source
- Powerful module system
- Reactive characteristics

Vert.x is reactive



{codemotion}

{ Demo

@bjschrijver

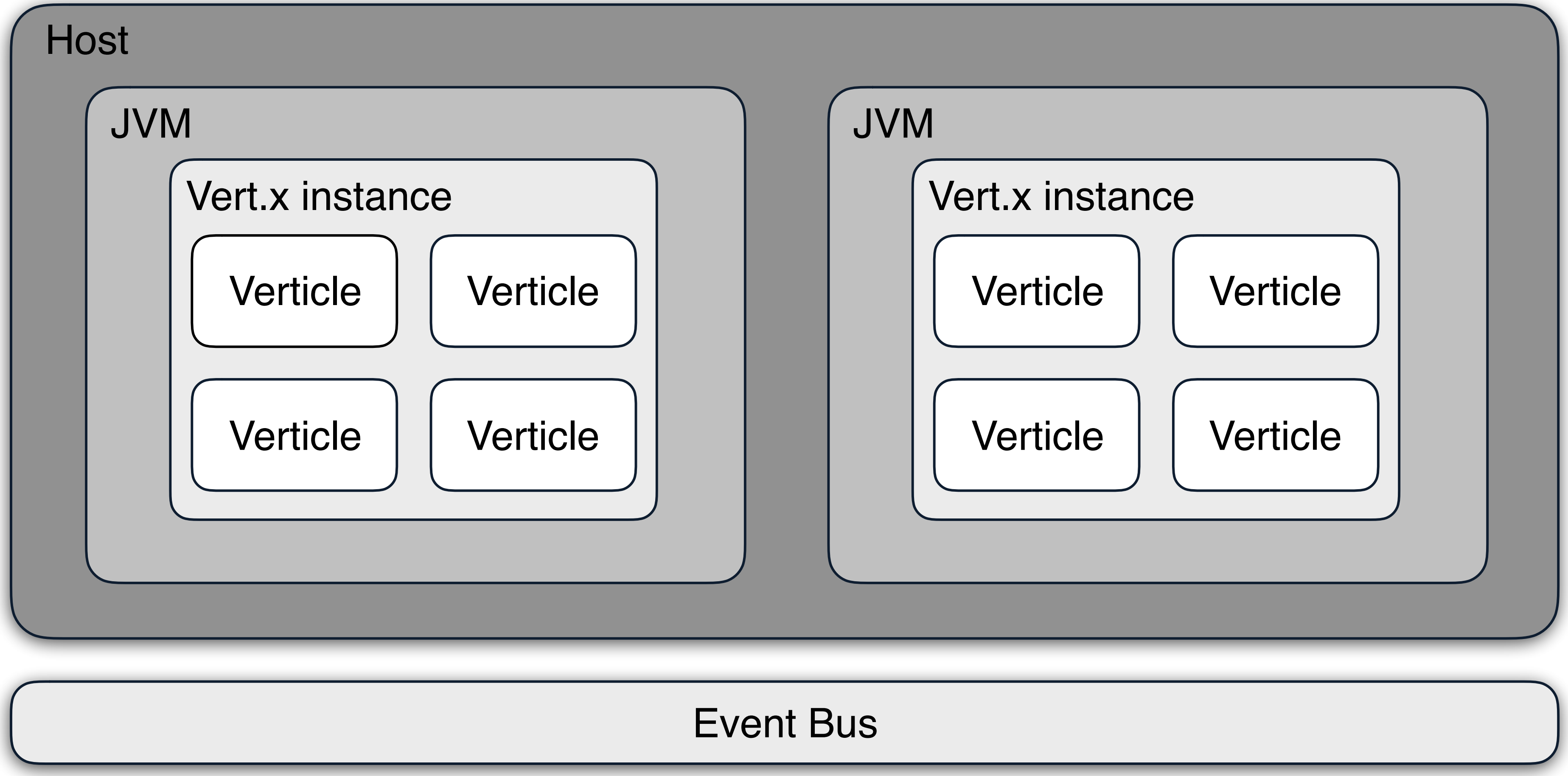
{codemotion}

{ Vert.x in depth

@bjschrijver

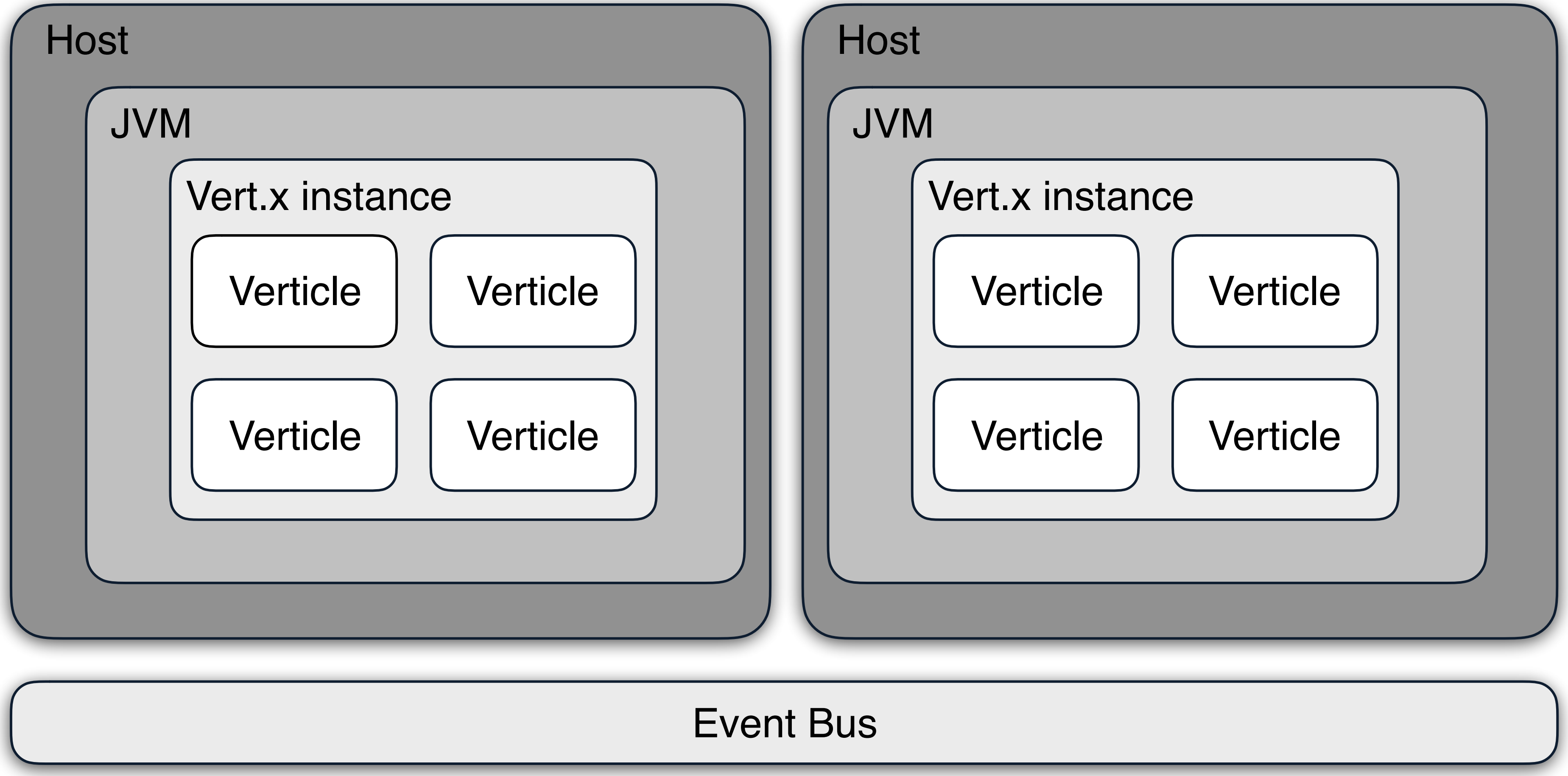
Vert.x in depth

- Verticle, application, instance, JVM:



Vert.x in depth

- Verticle, application, instance, JVM:



Vert.x in depth

- Event driven and non blocking event loop
- Multi-reactor pattern
- Actor-like concurrency
- Distributed event bus
- Management and monitoring built-in

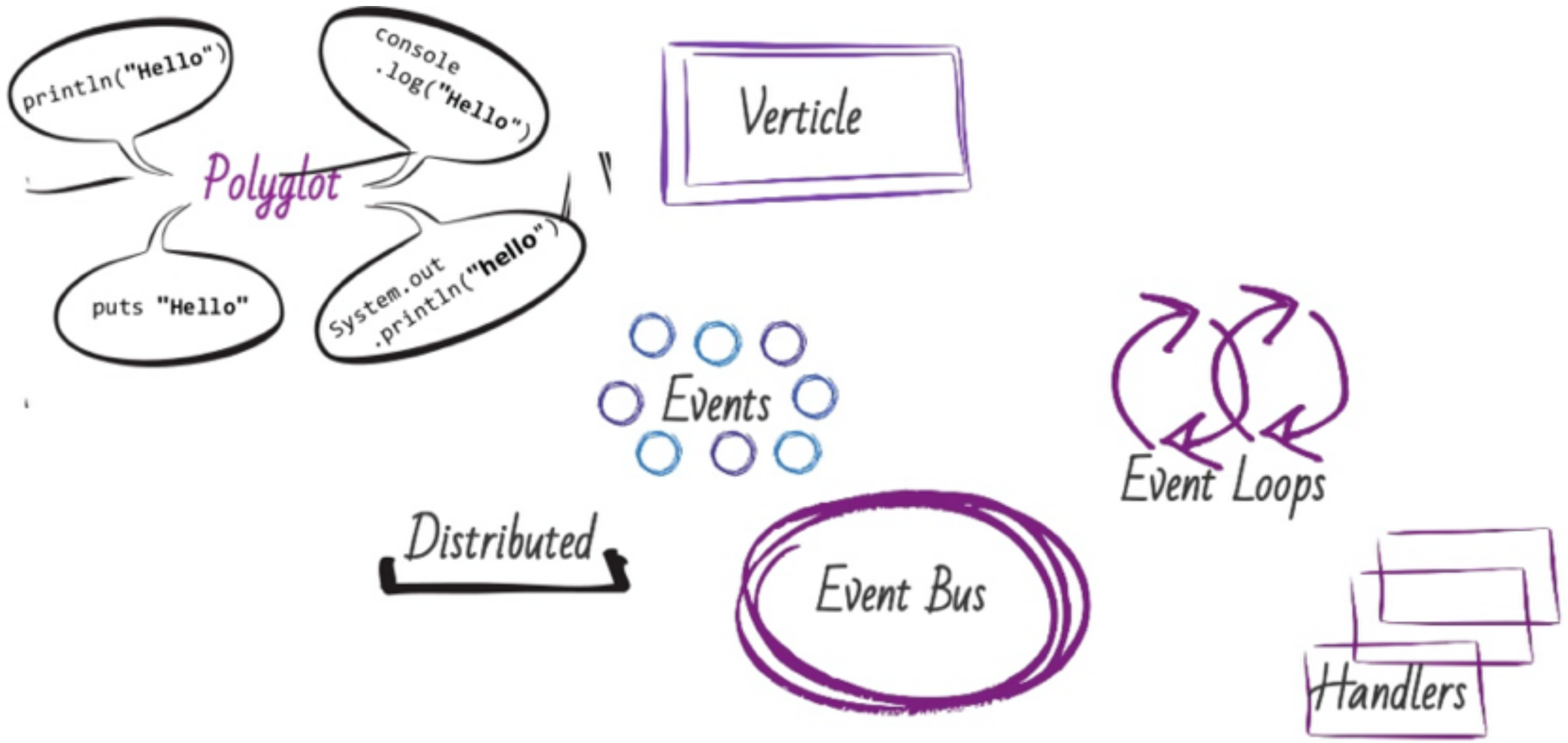
Vert.x event bus

- Both request/response and publish/subscribe
- Messages are received by Handlers
- All Vert.x instances have access to the event bus
- Verticles interact using messages

Vert.x’s golden rule

- Never ever ever block the event loop
- Need to to blocking stuff? Use a worker

Vert.x in one slide



Source: http://www.slideshare.net/clement.escoffier/vertx-31-be-reactive-on-the-jvm-but-not-only-in-java

{codemotion}

{ Demo

@bjschrijver

{codemotion}

{ Vert.x & microservices

@bjschrijver

Anatomy of a microservice

- Small in size, single responsibility
- Runs in its own process
- Has its own data store
- Distributed by default
- Independently develop, deploy, upgrade, scale
- Potentially heterogeneous/polyglot
- Light-weight communication

Anatomy of a Vert.x module

- ✓ Small in size, single responsibility
- ✓ Runs in its own process
- ✓ Has its own data store
- ✓ Distributed by default
- ✓ Independently develop, deploy, upgrade, scale
- ✓ Potentially heterogeneous/polyglot
- ✓ Light-weight communication

{ Vert.x in practice

@bjschrijver

3 years of Vert.x: situation

- Organisation structure:
    - Core platform team (infrastructure)
    - Core modules team (Vert.x modules)
    - Product teams
- Deployment model: AWS, Nginx, MongoDB
- Multiple lightweight artifacts

3 years of Vert.x: deployments

- Rolled our own deployment tooling and open sourced it (https://github.com/msoute/vertx-deploy-tools)
- Controlled from Jenkins
- Zero-downtime deployments
- Microservice deployments

VERT.X
DEPLOY–TOOLS

3 years of Vert.x: experiences

- Very suitable for test driven development
- Integration tests with embedded MongoDB
- Good cooperation from the Vert.x team

3 years of Vert.x: challenges

- Callback hell
    - Java 8 helped a LOT
    - Rx fixes the remaining issues
- Blocking the event loop
- Scaling / JVM overhead
- Debugging
- Static code analysis
- Upgrade from Vert.x 2 to Vert.x 3

{codemotion}

{ Almost there ;-)

@bjschrijver

Summary

- Vert.x: toolkit for reactive applications on the JVM
- Polyglot, event-driven
- Very suitable for building microservices

Resources

- https://github.com/bertjan/vertx3-examples
- http://vertx.io
- http://vertx.io/blog/my-first-vert-x-3-application
- https://github.com/vert-x3/vertx-examples
- http://github.com/msoute/vertx-deploy-tools

Don’t worry, I’ll tweet a link to the slide deck ;-)

{codemotion}

{ Q & A

@bjschrijver

{ Thanks!

@bjschrijver

bertjan@jpoint.nl