Monolith to Microservices

A Practical Guide to the Journey with RHOAR
OpenShift Commons Briefing

James Falkner
Technical Marketing Manager
Red Hat Middleware

RELATED OPENSHIFT COMMONS BRIEFINGS

commons.openshift.org/events.html

- #97 **RHOAR Explained**
  - September 27 - John Clingan
- #96 **Building Cloud Native Apps with Spring Boot and RHOAR**
  - September 21 - Thomas Qvarnstrom

MIGRATION AND MODERNIZATION APPROACHES

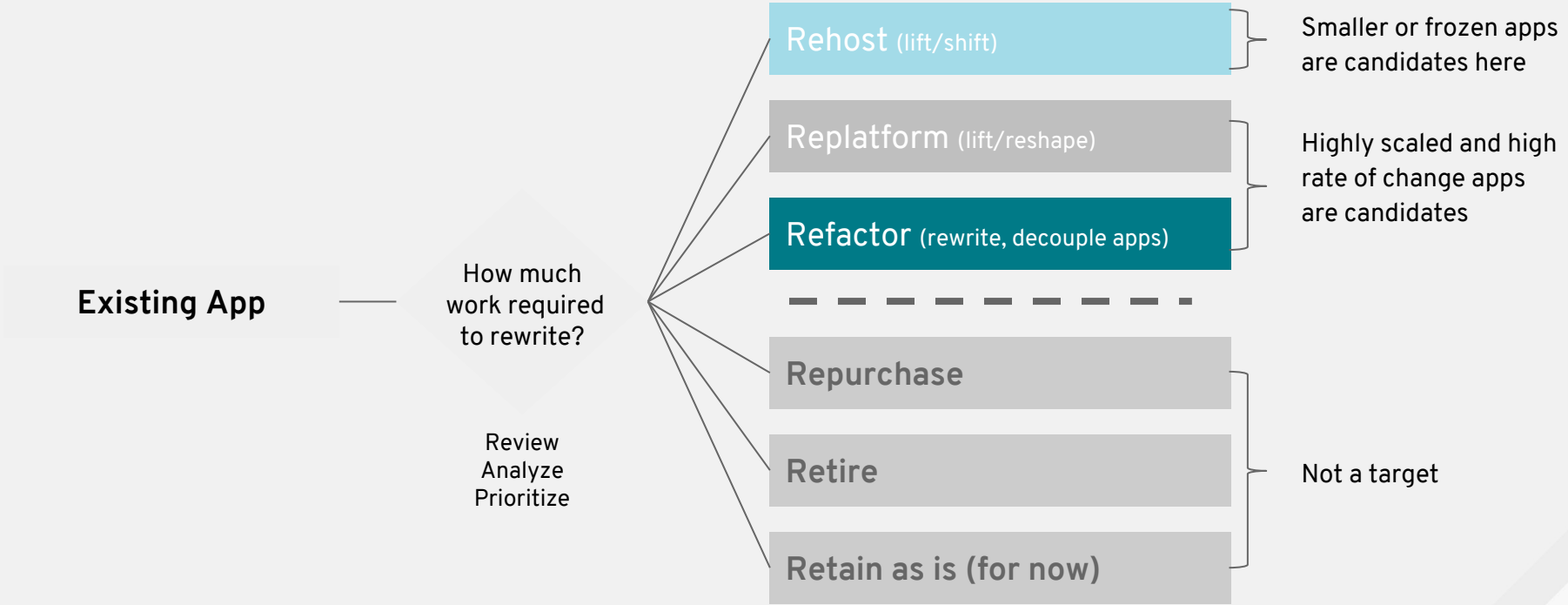
MODERNIZING EXISTING APPS

- Reuse existing functionality and data as much as possible
- Move existing workloads to a modern deployment platform
- Apply new processes, products, and technology to existing apps

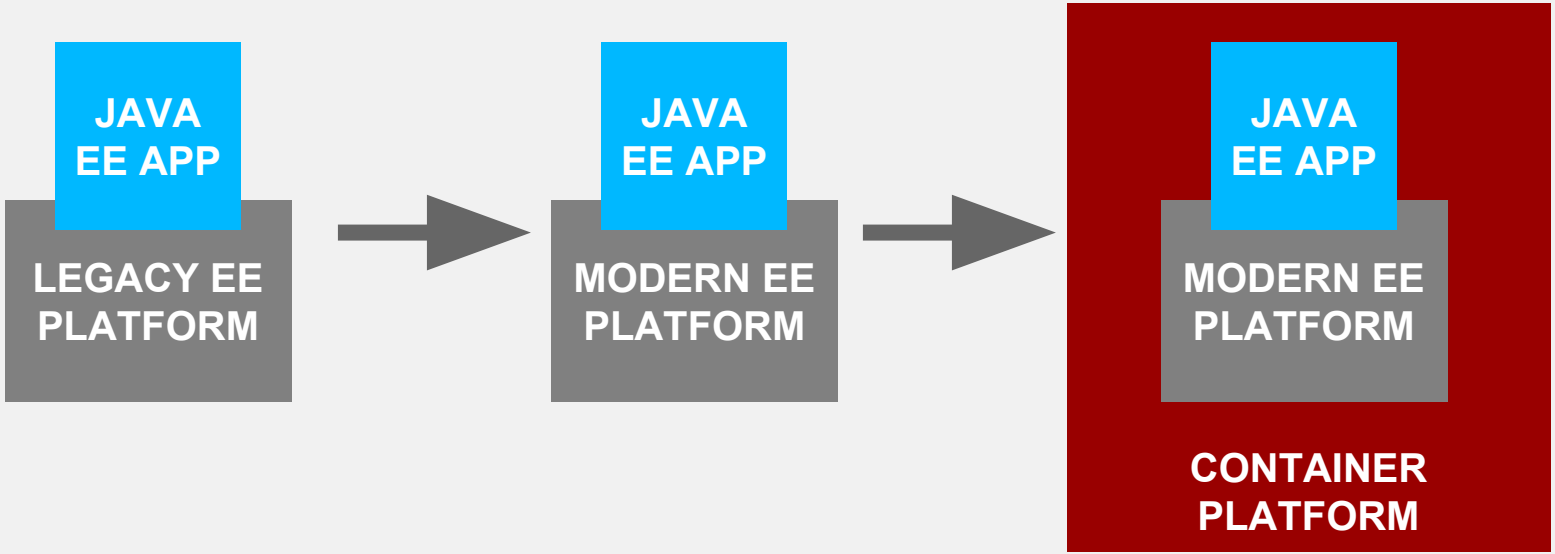
DEVELOPING NEW APPLICATIONS

- API-centric polyglot microservices architecture
- Autonomous development teams
- Agile development, continuous deployment, DevOps culture
- Containerized & orchestrated cloud deployments

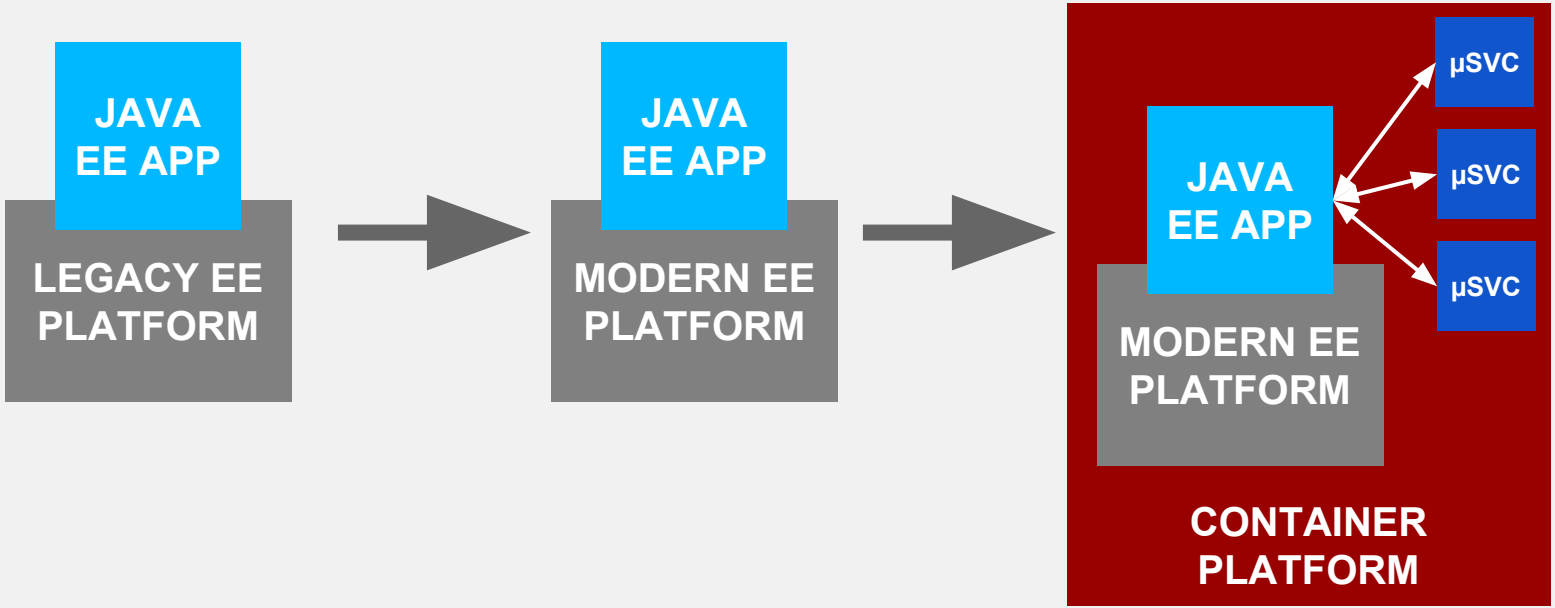
OPTIONS FOR APPLICATION MODERNIZATION



REHOST


JAVA EE APP
LEGACY EE PLATFORM
JAVA EE APP
MODERN EE PLATFORM
JAVA EE APP
MODERN EE PLATFORM
CONTAINER PLATFORM

REPLATFORM


JAVA EE APP
LEGACY EE PLATFORM
JAVA EE APP
MODERN EE PLATFORM
JAVA EE APP
MODERN EE PLATFORM
µSVC
µSVC
µSVC
CONTAINER PLATFORM

Majestic Monolith

https://m.signalvnoise.com/the-majestic-monolith-29166d022228

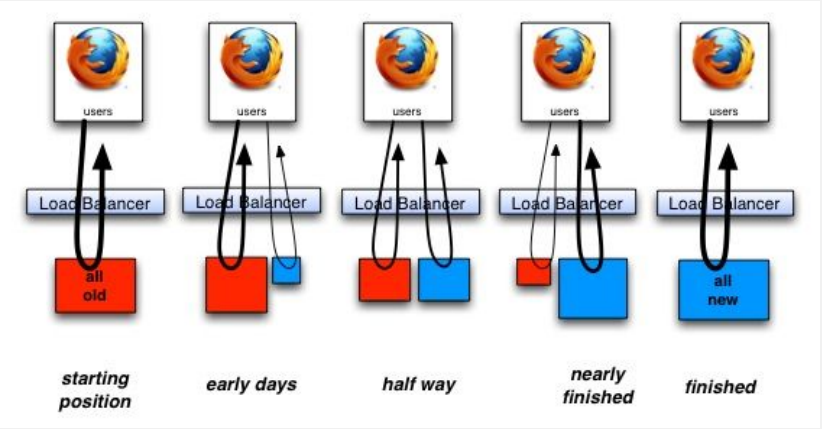
THE FAST MONOLITH AT KEYBANK

- Refactored to REST and JavaScript
- Adopted Container on Red Hat OpenShift
- Automated Testing
- Redefined Dev and Ops Boundaries
- Continuous Deployment Pipeline
- Zero Downtime Release to Production
- Release Cycles From 3 months to 1 Week

Blog: red.ht/2jwR07s

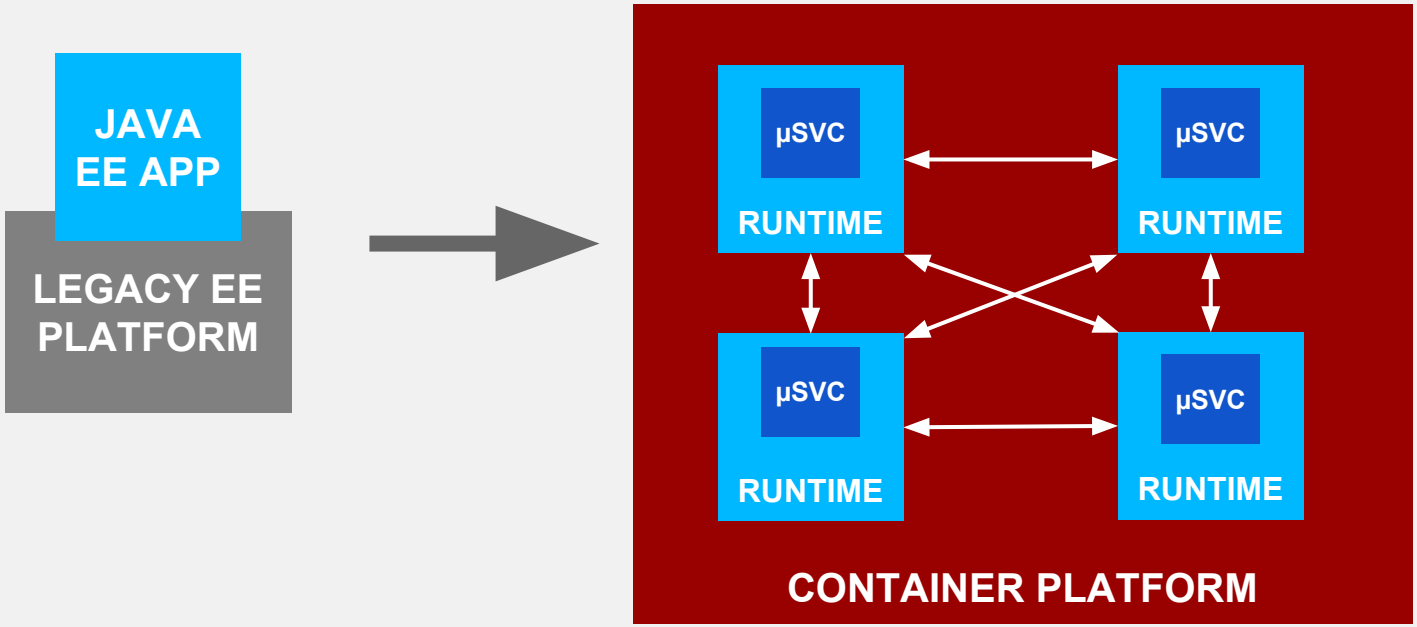
STRANGLING THE MONOLITH

- Strangling - **incrementally** replacing functionality in app with something better (cheaper, faster, easier to maintain).
- As functionality is replaced, “dead” parts of monolith can be removed/retired.
- Includes new functionality during strangulation to make it more attractive to business stakeholders.

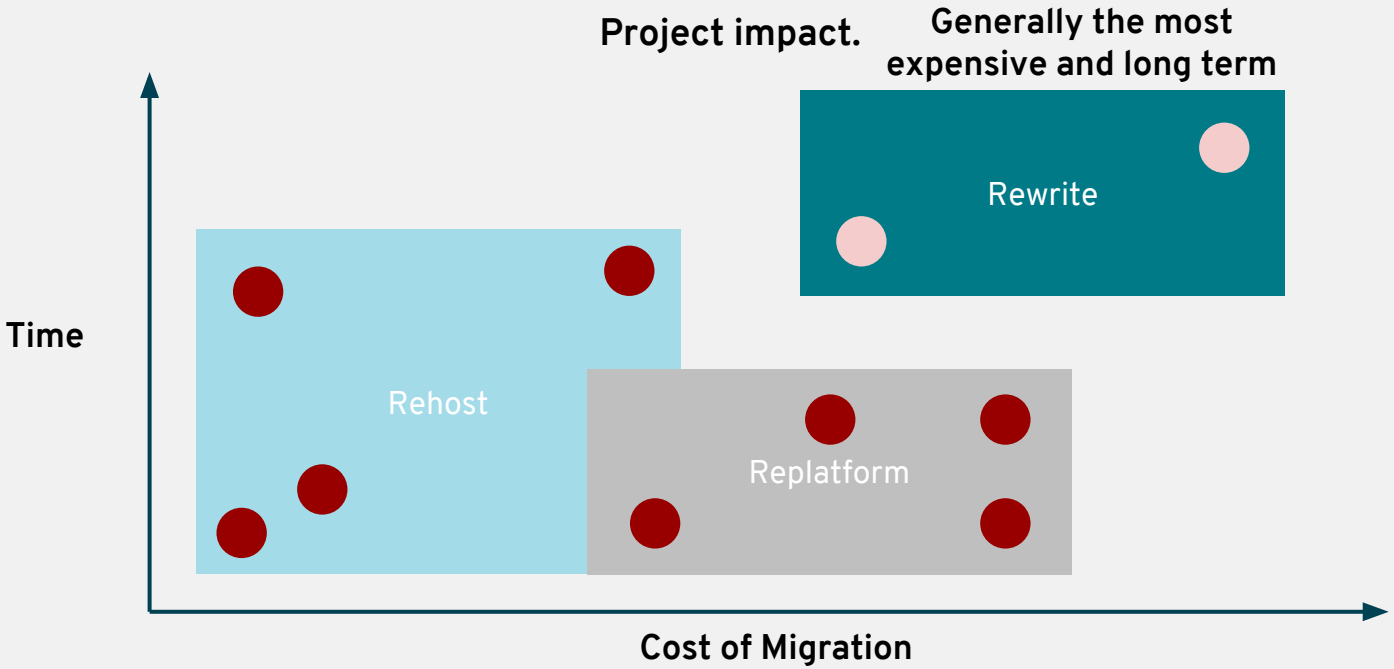


Paul Hammant
“Legacy Application Strangulation: Case Studies”
paulhammant.com

REFACTOR


JAVA EE APP
LEGACY EE PLATFORM
µSVC
RUNTIME
µSVC
RUNTIME
µSVC
RUNTIME
µSVC
RUNTIME
CONTAINER PLATFORM

PATTERNS IN MODERNIZING WORKLOADS



KEY QUESTIONS TO ANSWER

What is your overall business objective for app modernization?

- Biggest perceived risks?

How much are you spending on app maintenance?

- Lack of automation/IT standardization often the culprit

How long does it take to get changes into production?

- And what is your success/fail ratio?

Current skill set?

- Do you need training on newer technology before modernization?
- Consider future availability of skills in the workforce.

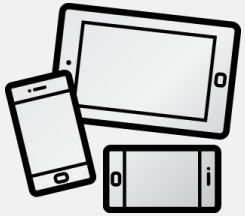
Regulatory/compliance requirements?

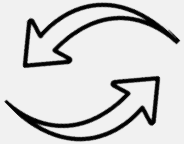
- Regional CCSP workloads
- Data Sovereignty

THE PATH TO MODERN APP DEV

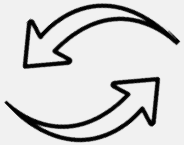
A DIGITAL DARWINISM

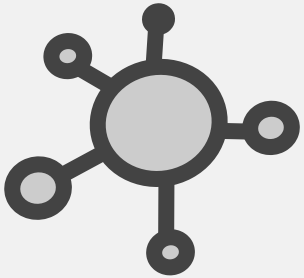
THE NEW DIGITAL ARCHITECTURE





API





asynchronous

event-driven

anti-fragile

serverless

reactive

scalable

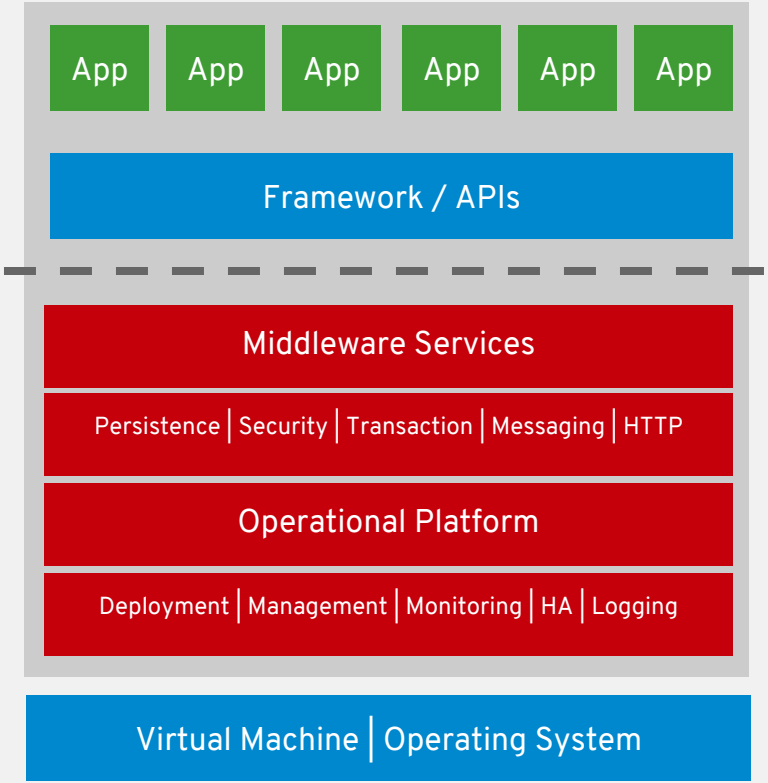
polyglot

velocity

agile

microservices

THE APPSERVER 2000-2014



Java COMPATIBLE ENTERPRISE EDITION

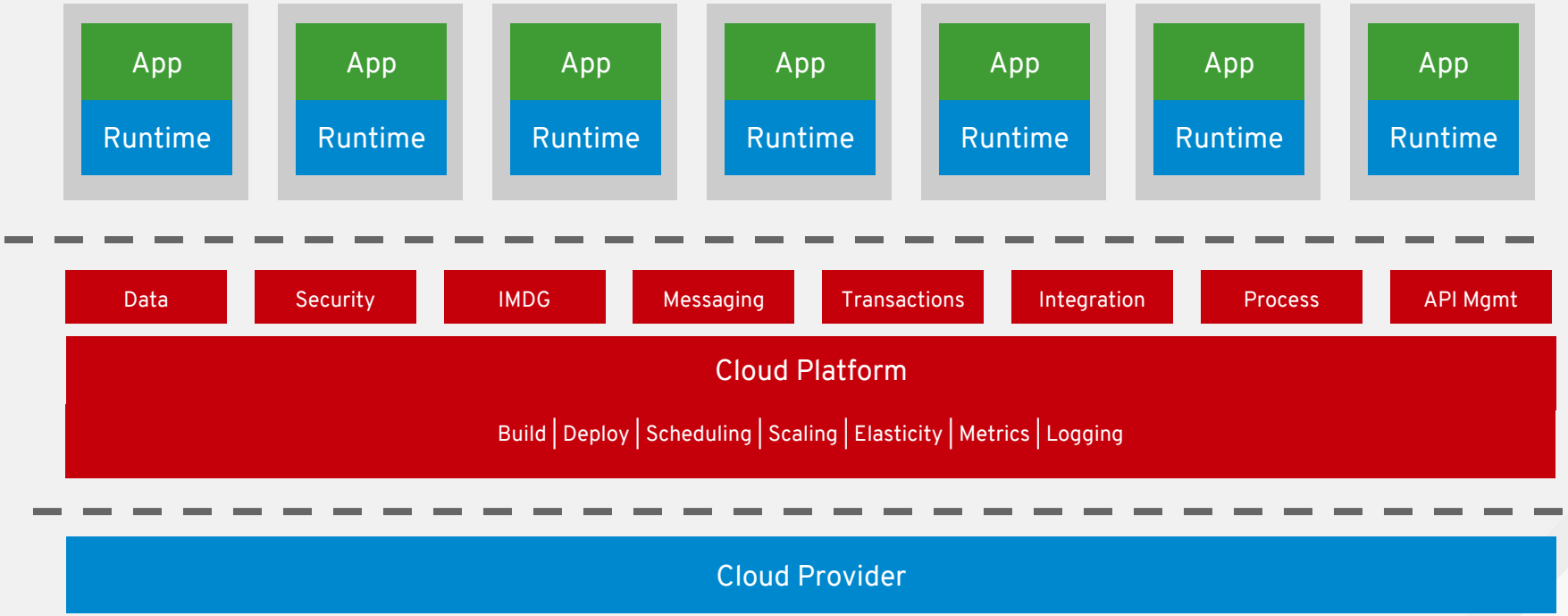
spring by Pivotal

JBoss by Red Hat

ORACLE

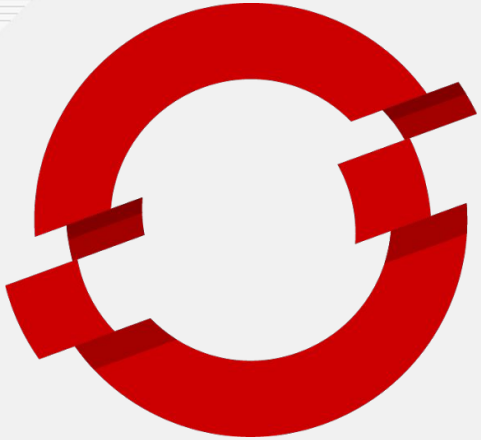
WebSphere software

THE APPSERVER 2014-...



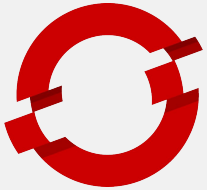
RED HAT OPENSHIFT APPLICATION RUNTIMES

redhat.



RED HAT® OPENSHIFT Application Runtimes

Modern, cloud-native application runtimes and a guided developer experience for organizations that are moving beyond 3-tier architectures and embracing cloud-native application development.



RED HAT® OPENSHIFT Application Runtimes

TESTED AND VERIFIED MICROSERVICES FRAMEWORKS

LAUNCH

- Spring Boot
- Netflix Hystrix
- Netflix Ribbon

SUPPORTED MICROSERVICES RUNTIMES

Reactive	MicroProfile	Server-side JS	Java EE	Java EE Web
Eclipse Vert.x	WildFly Swarm	Node.js	JBoss EAP	Embedded Tomcat

Modern, Cloud-Native Application Runtimes and
A Guided Developer Experience

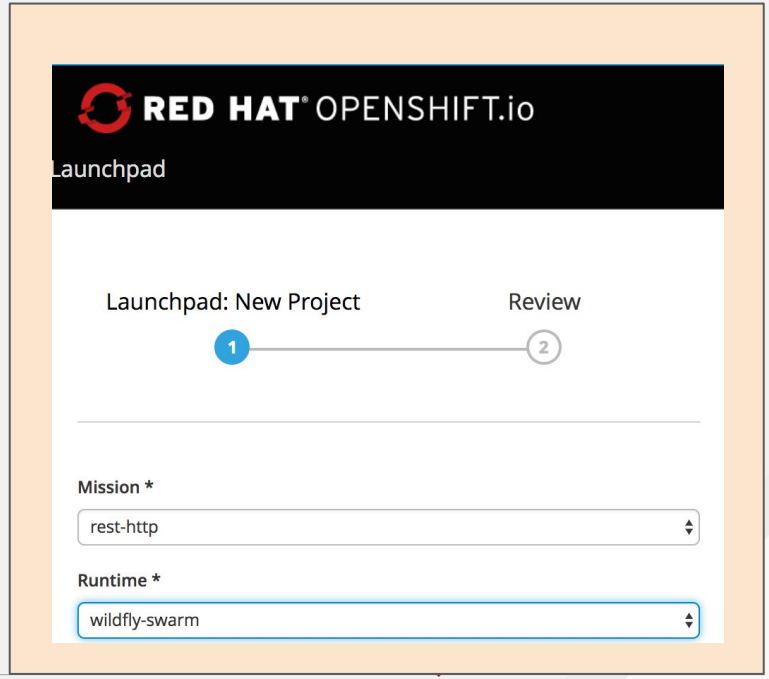
OpenShift Application Runtimes

- **Multiple runtime options**
  - JBoss EAP - existing Java EE / Spring apps.
  - WildFly Swarm / MicroProfile - Java EE centric MSA
  - Spring Boot / Cloud - Spring centric MSA
  - Vert.x - greenfield reactive Java
  - Node.js - greenfield reactive JavaScript
- **OpenShift - Public, Dedicated Public & Enterprise**
- **Tightly integrated with OpenShift & Kubernetes**
- **Tightly Integrated with Red Hat Developer SaaS**
- **3rd-party Integrations - eg. Netflix Ribbon, Hystrix, etc.**
- **Opinionated DevX through S2I, quickstarts and generators**

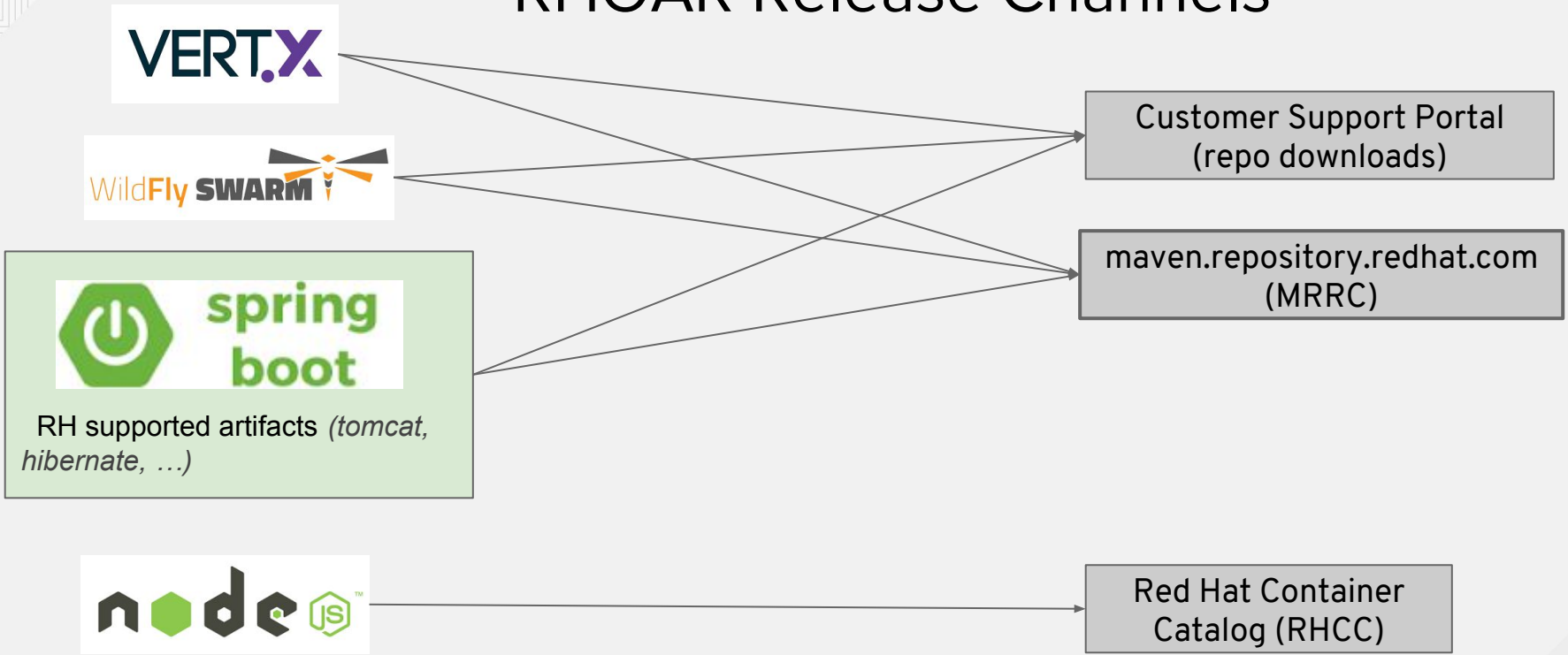
developers.redhat.com/launch

Cloud Native Samples in the Cloud

- Accelerate the learning / evaluation experience
- Collection of cloud native examples
- Leverage the platform
- Runs entirely in OpenShift
  - On Desktop or OpenShift Online
- Spring Boot, Vert.x, WildFly Swarm, Node.js



RHOAR Release Channels



Example: Using RHOAR for WildFly Swarm

```
<project
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://maven.apache.org/POM/4.0.0"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http:/
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.redhat.coolstore</groupId>
    <artifactId>monolith</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>monolith</name>
    <repositories>
        <repository>
            <id>jboss-maven-repository</id>
            <name>Red Hat Maven Repository</name>
            <url>http://maven.repository.redhat.com/ga/</url>
            <layout>default</layout>
            <releases>
                <enabled>true</enabled>
                <updatePolicy>never</updatePolicy>
            </releases>
            <snapshots>
                <enabled>false</enabled>
                <updatePolicy>never</updatePolicy>
            </snapshots>
        </repository>
```

Example: Using RHOAR for WildFly Swarm

```
                </plugin>
            </plugins>
        </build>
    </profile>
</profiles>
<dependencyManagement>
    <dependencies>

        <!-- RHOAR BOM -->
        <dependency>
            <groupId>org.wildfly.swarm</groupId>
            <artifactId>bom-all</artifactId>
            <version>${version.wildfly.swarm}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
```

Example: Using RHOAR for WildFly Swarm

```
        </dependency>

    </dependencies>
</dependencyManagement>
<dependencies>
    <!-- WildFly Swarm Health Check plugin -->
    <dependency>
        <groupId>org.wildfly.swarm</groupId>
        <artifactId>monitor</artifactId>
    </dependency>

    <dependency>
        <groupId>javax</groupId>
        <artifactId>javaee-web-api</artifactId>
        <version>7.0</version>
        <scope>provided</scope>
    </dependency>
```

RUNTIMES: LET THE CODING BEGIN

`github.com/jamesfalkner/rhoar-examples`

Initial starting point on `master` branch

Solution on `solution` branch

redhat.

WILDFLY SWARM

redhat.

- Microservices for Java EE developers
- Combines a small subset of Java EE and microservices technologies
- Package as an uber-jar
- Package only what you need
- Built from WildFly
- Based on standards

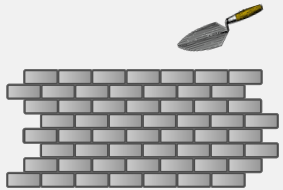
Optimizing Java EE for a Microservices Architecture

Release 1.1

JAX-RS
CDI
JSON-P
Config

Rapidly iterate and innovate

Build consensus



Standardize

JOIN THE MICROPROFILE COMMUNITY

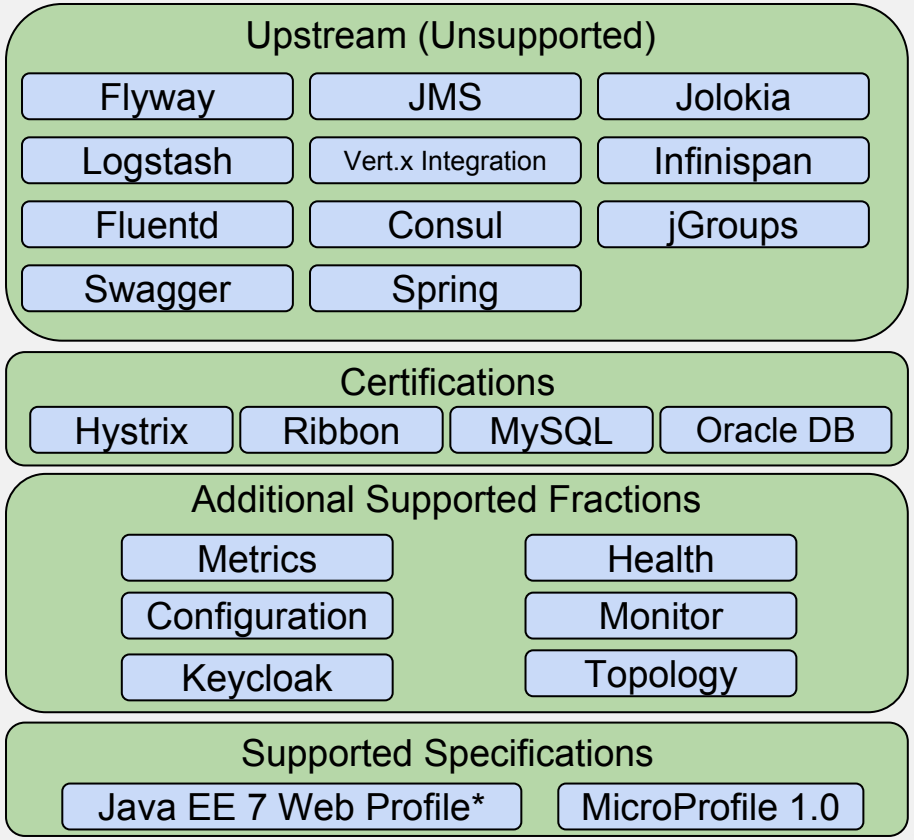
WildFly Swarm in RHOAR

Build microservices

- Embeddable (Fat Jar)
- Lightweight
- Modular & extensible
- Built from WildFly (Trusted and Reliable)

OPENSHIFT

MICROPROFILE™



* Planned

SPRING /

SPRING BOOT

redhat.

- Microservices for Developers using Spring Framework
- An opinionated approach to building Spring applications
- Red Hat Certified with
  - OpenShift Java Runtime
  - JBoss Web Server (Tomcat) embedded web container
- More Red Hat technologies to come

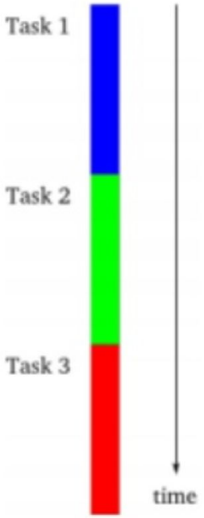
Spring in RHOAR

- **It's the same Spring you know and love**
- Tested and Verified by Red Hat QE
  - Spring Boot, Spring Cloud Kubernetes, Ribbon, Hystrix
- Red Hat components fully supported
  - Tomcat, Hibernate, CXF, SSO (Keycloak), Messaging (AMQ), …
- Native Kubernetes/OpenShift integration (Spring Cloud)
  - Service Discovery via k8s (DNS), Ribbon
  - Spring Config via ConfigMap
- Developer Tooling (launch.openshift.io, starters)
- Additional planned support for
  - Transactions (Naryana), Messaging (AMQ), more

ECLIPSE VERT.X

redhat

Execution Models



Single-Threaded Synchronous Model

- Not much to say here

cs.brown.edu/courses/cs168/s12/handouts/async.pdf

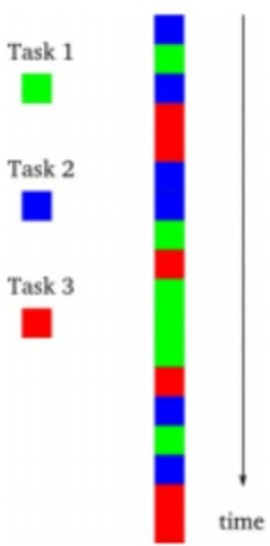
redhat.

Threaded Model

- CPU controls interleaving
- Developer must coordinate threads/processes
- “preemptive multitasking”

redhat.

Execution Models



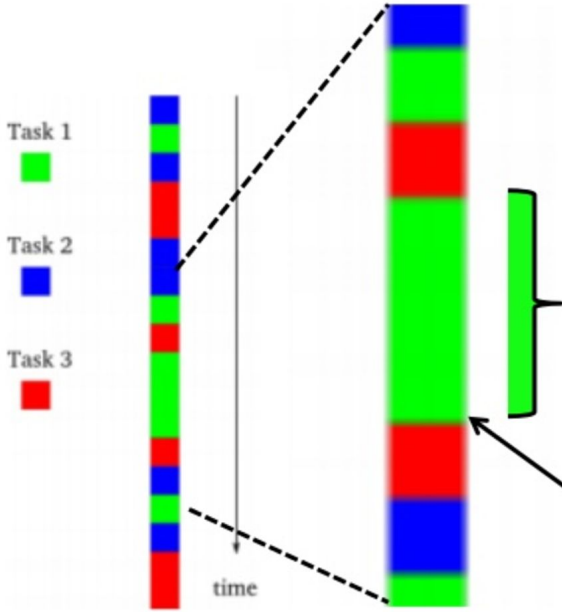
Asynchronous Model

- Developer controls interleaving
- “cooperative multitasking”
- Wave goodbye to race conditions, synchronized, and deadlocks!
- Windows 3.x, MacOS 9.x, Space Shuttle

also VERT.X node

redhat.

Execution Models



The green code (your code!) runs **uninterrupted** until **it (you!)** reaches a “good stopping point”

(I/O)

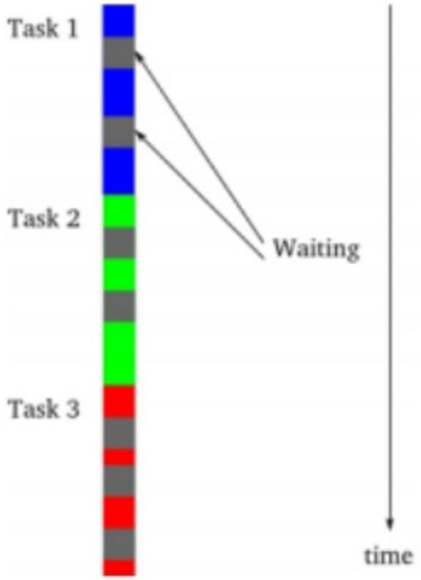
redhat.

What does this buy me?

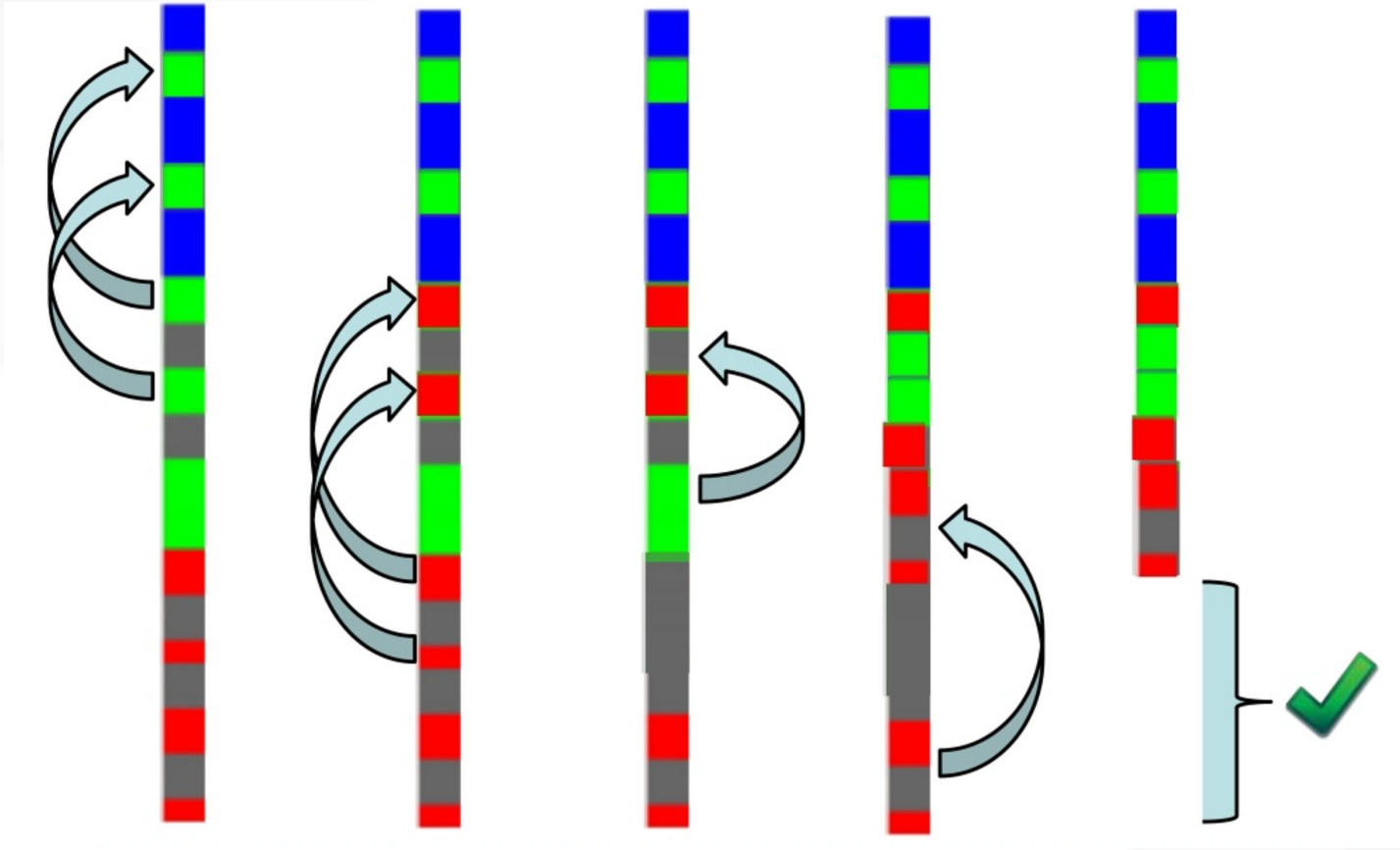
NOTHING! Except when

- Task Pool is large
- Task I/O >> Task CPU
- Tasks mostly independent

… Like web servers



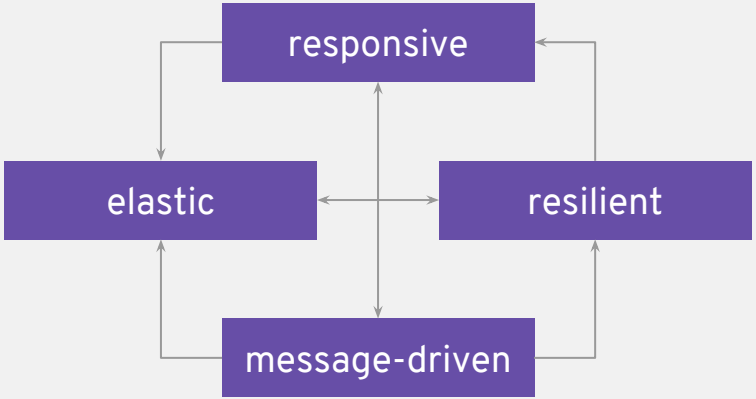
redhat.



redhat.

VERT.X

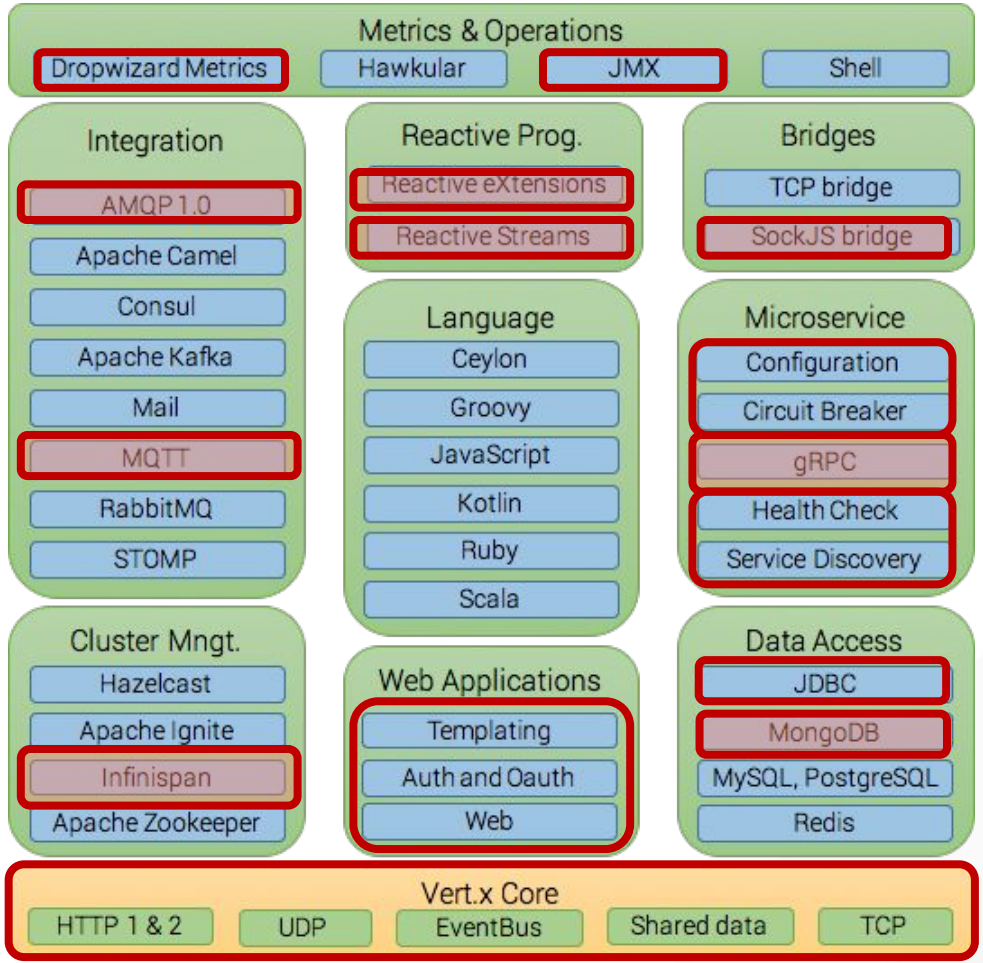
- Reactive Microservices for JVM
- Ideal for High Concurrency and Low Latency Services
- Lightweight Messaging
- Event Driven Non-Blocking I/O
- 2014 JAX Innovations Award Winner
- Polyglot: Java, JavaScript, Groovy, Ruby, Ceylon, Scala and Kotlin



Eclipse Vert.x in RHOAR

Build reactive systems

- Polyglot [Java supported]
- Integrable
- Embeddable
- Pragmatic
- Freedom



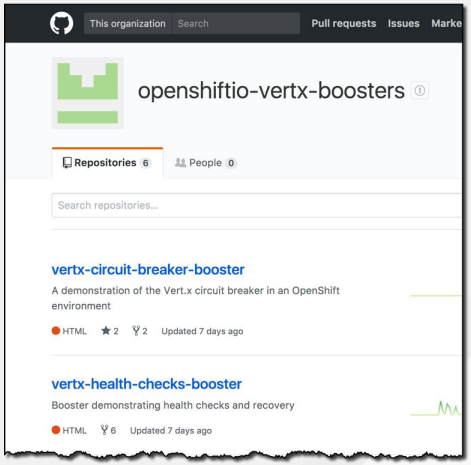
Productized

Technical Preview

redhat.

Vert.x and RHOAR

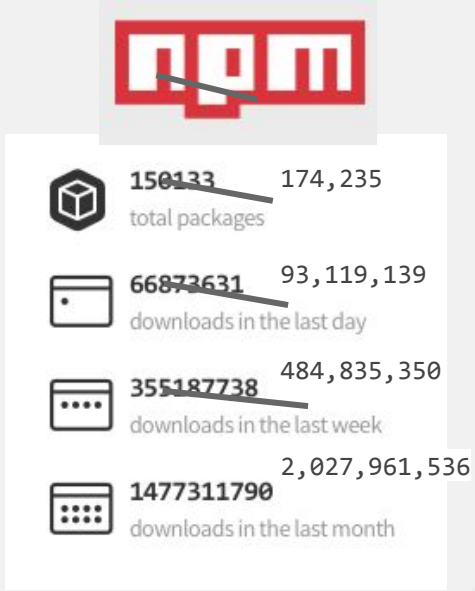
- RHOAR releases include the most recently released version of Vert.x
  - Currently 3.4.2
- Supported & Certified components imported using Maven BOMs
  - `<version>3.4.2.redhat-3</version>`
- Quick Starts are called **Boosters**
- Launch will create projects based on the Boosters
- Booster source code and samples available here
  - https://github.com/openshiftio-vertx-boosters



NODE.JS

(TECH PREVIEW IN FIRST RHOAR GA)

- Large, vibrant Community
- Benefits
  - Everyone knows JavaScript
  - Isomorphic JavaScript
  - Developer Productivity
  - Performance and scalability
- Open Source
  - moved from BDFL to Foundation
  - Enterprise collaboration



Node.js in RHOAR

- High priority for RHOAR with feature parity to other runtimes
- Node core distro to be delivered only through RHOAR, no stand alone sku
- Non-Distro efforts
  - Team working on tooling, boosters for RHOAR integration
- Booster coverage minimal atm
  - Focused on infrastructure/workflow
  - Boosters are fairly quick after that
- Consumption requirements
  - S2I images
  - Openshift Streams integration
- SCL conflict TBD

SUMMARY

- Application development teams are **evolving** their process, platform and architecture to meet modern business challenges
- There are **multiple** technical solutions for app modernization depending on resources, regulations and risk.
- Some organizations move **faster** than others
- Red Hat & RHOAR provides a trusted solution for **today's** business-critical apps and a supported path to **modern** application architectures

THANKS!

redhat.