Monoliths to microservices: App Transformation

Hands-on Technical Workshop

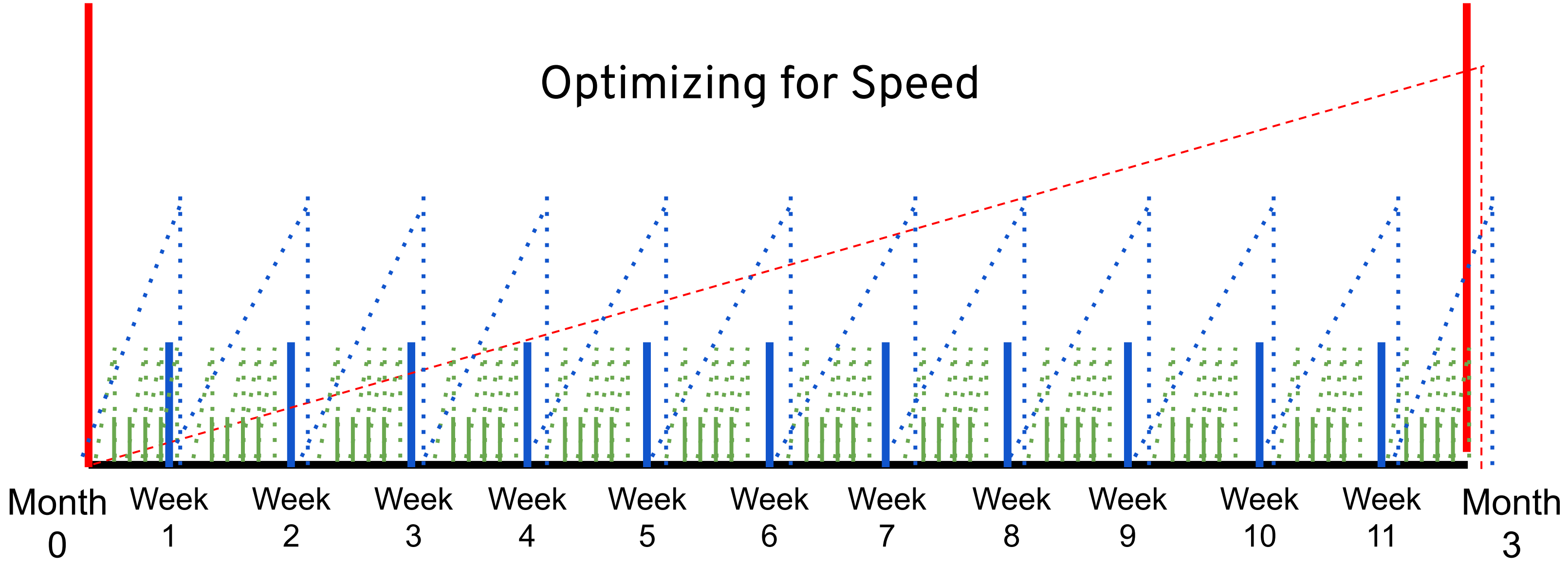
Part 3: Monoliths to microservices with MicroProfile & Spring Boot

Why monolith to microservices

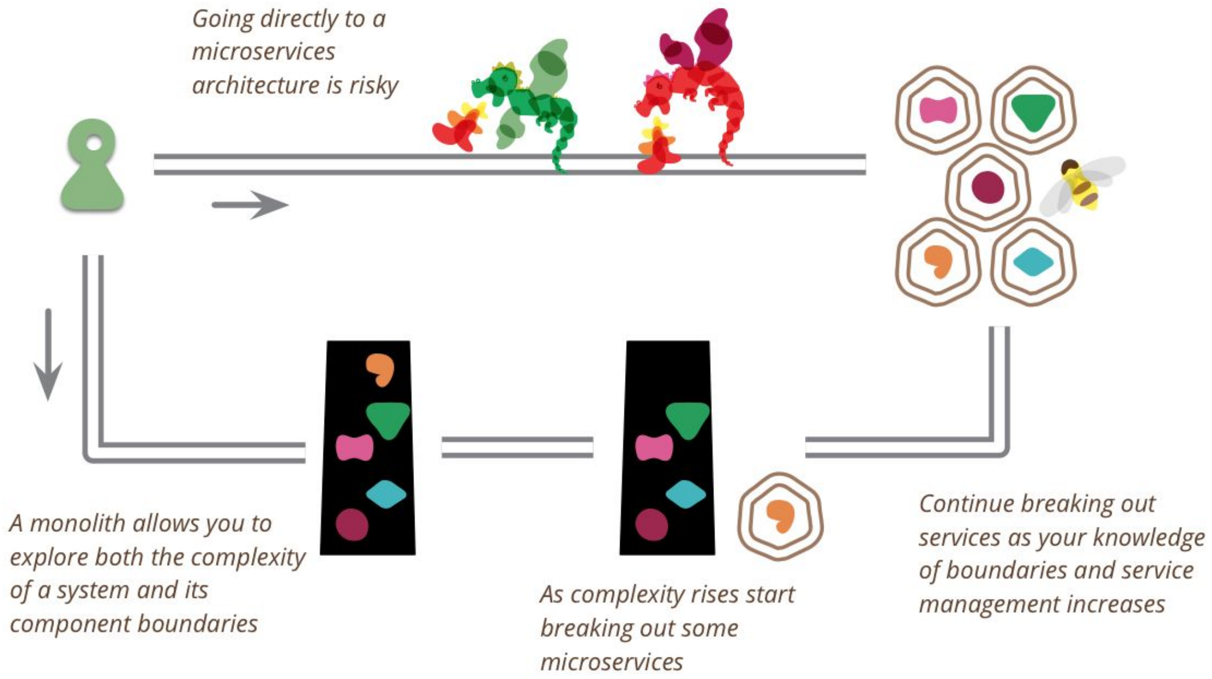
Break things down (organizations, teams, IT systems, etc) down into smaller pieces for greater parallelization and autonomy and focus on reducing time to value.

Reducing time to value

Monolith Lifecycle
Fast Moving Monolith
Microservices



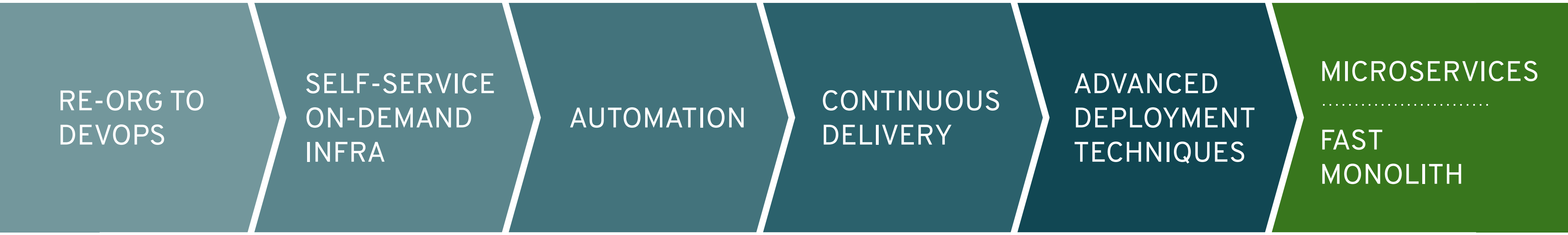
Monolith first?


Going directly to a microservices architecture is risky
A monolith allows you to explore both the complexity of a system and its component boundaries
As complexity rises start breaking out some microservices
Continue breaking out services as your knowledge of boundaries and service management increases

http://martinfowler.com/bliki/MonolithFirst.html

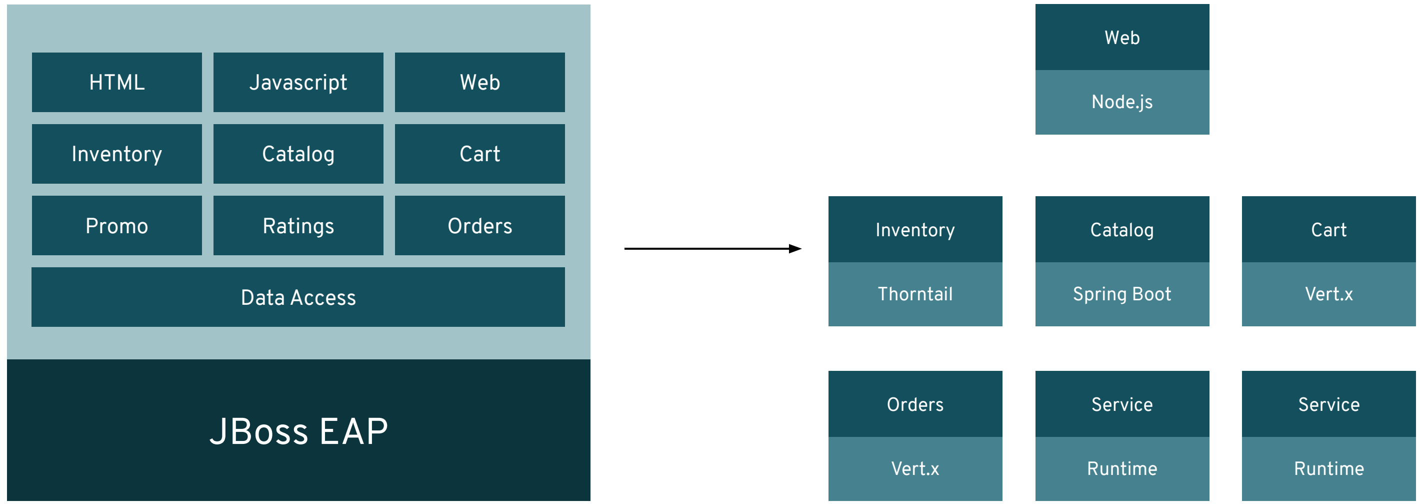
The bigger picture: the path to cloud-native apps

A DIGITAL DARWINISM



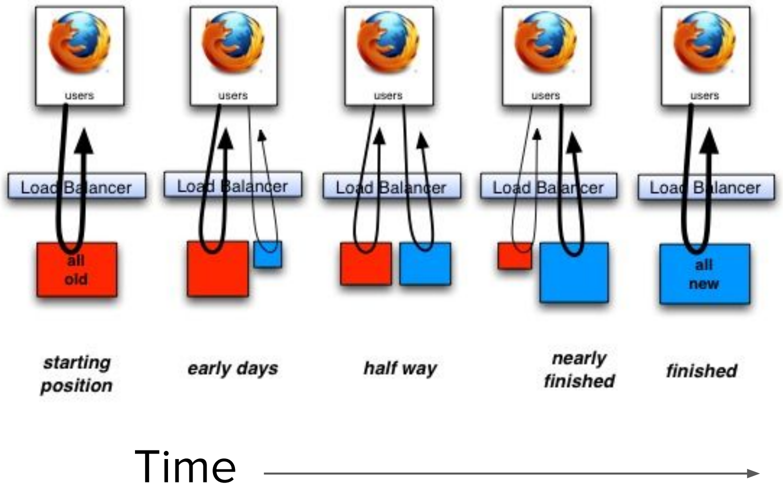
Strangling the monolith

- In this lab, you will begin to ‘strangle’ the coolstore monolith by implementing its services as external microservices, split along business boundaries
- Once implemented, traffic destined to the original monolith’s services will be redirected (via OpenShift software-defined routing) to the new services



Strangling the monolith

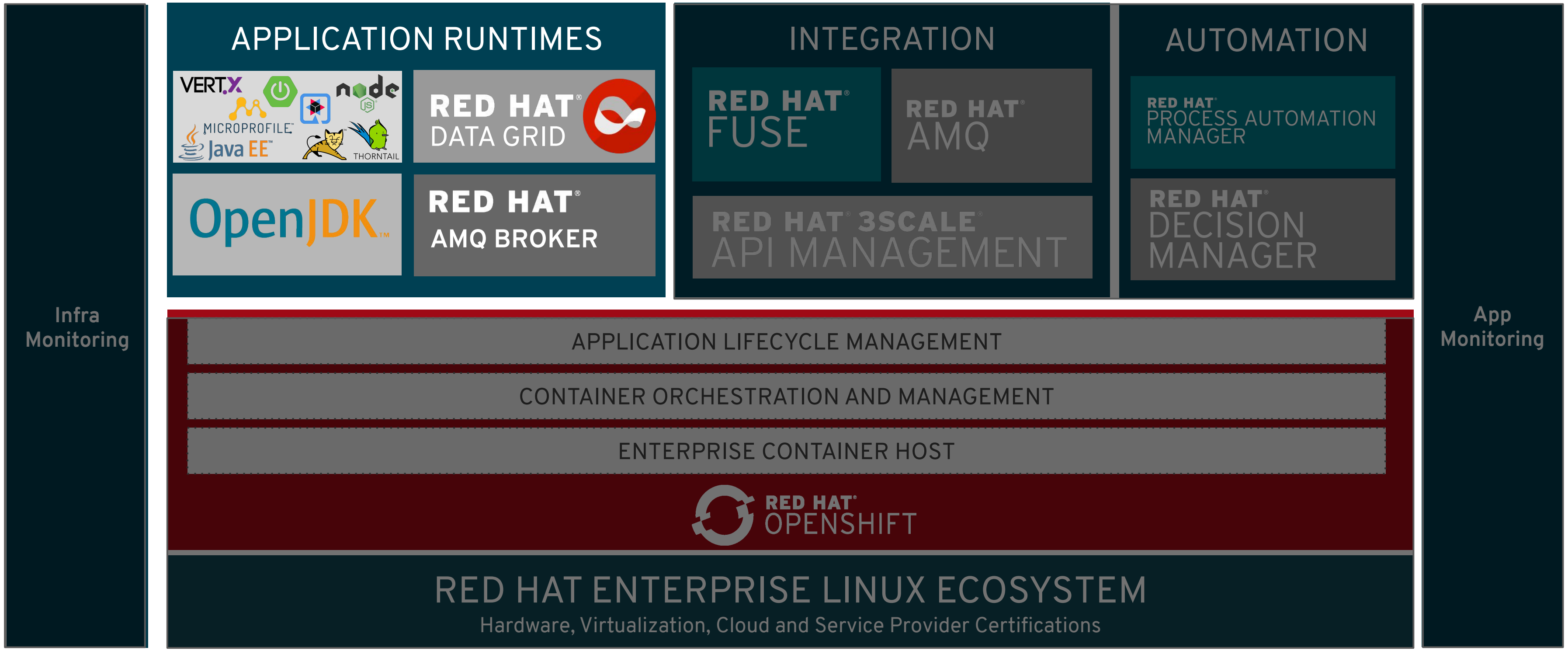
- Strangling - **incrementally** replacing functionality in app with something better (cheaper, faster, easier to maintain).
- As functionality is replaced, "dead" parts of monolith can be removed/retired.
- You can also wait for all functionality to be replaced before retiring anything!
- You can optionally include new functionality during strangulation to make it more attractive to business stakeholders.



Red Hat Runtimes

Red Hat platform for the hybrid cloud

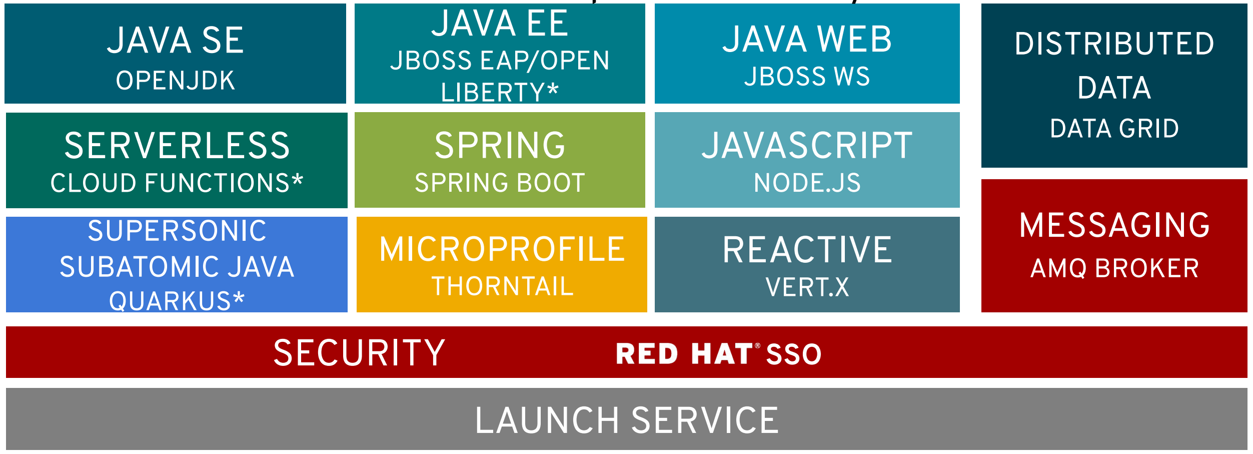
OpenShift and Middleware optimized for the cloud


Infra Monitoring
APPLICATION RUNTIMES
VERT.X
node JS
MICROPROFILE
Java EE
THORNTAIL
RED HAT DATA GRID
OpenJDK
RED HAT AMQ BROKER
INTEGRATION
RED HAT FUSE
RED HAT AMQ
RED HAT 3SCALE API MANAGEMENT
AUTOMATION
RED HAT PROCESS AUTOMATION MANAGER
RED HAT DECISION MANAGER
App Monitoring
APPLICATION LIFECYCLE MANAGEMENT
CONTAINER ORCHESTRATION AND MANAGEMENT
ENTERPRISE CONTAINER HOST
RED HAT OPENSHIFT
RED HAT ENTERPRISE LINUX ECOSYSTEM
Hardware, Virtualization, Cloud and Service Provider Certifications

Red Hat Runtimes

Red Hat Runtimes

Non-restrictive development for the hybrid cloud



Optimized for OpenShift / Kubernetes Services with pre-configured Missions and Boosters
Integration with RH Developer, CI/CD tools, Security Services
Available Application Migration Toolkit
Python, Go and .Net also supported by Red Hat (with a different SLA)

Facilitate cloud native app development ON THE HYBRID CLOUD:

- ✓ Faster getting started
- ✓ Simplify container dev
- ✓ Automate DevOps
- ✓ Standardize tools/processes
- ✓ Fully supported JDK

Spring

spring

- Microservices for Developers using Spring Framework
- An opinionated approach to building Spring applications
- Historical alternative to Java EE
- Getting started experience
- Spring MVC / DI / Boot most popular

Spring in Red Hat Runtimes

- **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 (Narayana), Messaging (Rabbit MQ -> AMQ), more

Cloud native support in Spring

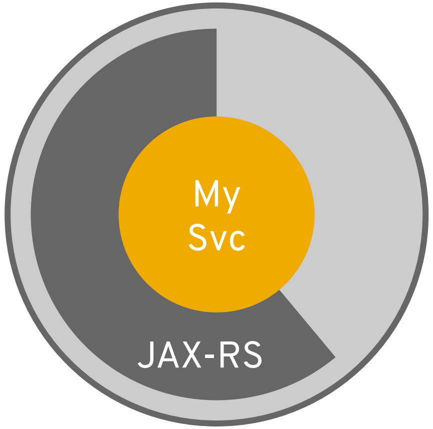
- Health Checks (actuator)
- Externalized Config (spring-cloud-kubernetes)
- Client-side discovery / load balancing (Eureka/Kubernetes)
- Circuit Breaking / Bulkheading (Hystrix)
- Logging / Monitoring / Tracing / Metrics
- Secure deployments with Keycloak
- API Documentation (Swagger)

Thorntail

THORNTAIL

Java EE microservices

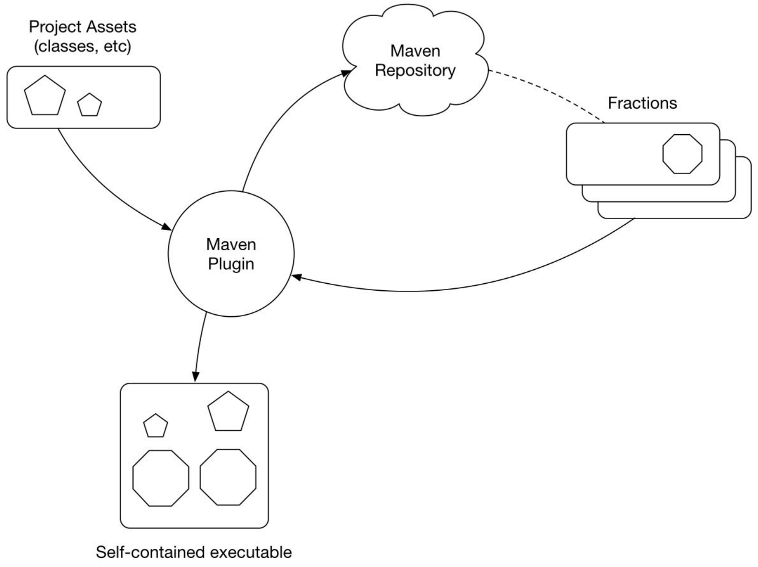
- Leverage Java EE expertise
- Open standard
- Microservices focus
- Optimized for OpenShift
- Super lightweight
- Tooling for Developers
- MicroProfile Implementation



```
$ java -jar my_microservice.jar
```

Thorntail “pieces” - Fractions

- A tangible unit providing a specific piece of functionality
- Embodied in a maven artifact
- To support the compositional aspect in Thorntail
- Provides the “runtime” capabilities
- Means to add API dependencies (e.g. JAX-RS)
- Means to configure the system
  - With reasonable defaults
- Means to discover other components (topology)
- Means to alter deployments (e.g. keycloak)
- Can be auto-detected or explicitly declared

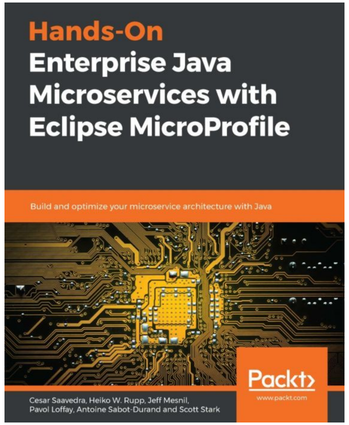


Cloud native support in Thorntail

- Health Checks
- Externalized Config
- Client-side discovery / load balancing
- Circuit Breaking / Bulkheading
- Logging / Monitoring / Tracing / Metrics
- Secure deployments with Keycloak
- MicroProfile
- API Documentation

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

eclipse


Hands-On
Enterprise Java Microservices with Eclipse MicroProfile
Build and optimize your microservice architecture with Java
Packt>
www.packt.com
Cesar Saavedra, Heiko W. Rupp, Jeff Mesnil, Pavol Loffay, Antoine Sabot-Durand and Scott Stark

- Defines open source Java microservices specifications
- Industry Collaboration - Red Hat, IBM, Payara, Tomitribe, London Java Community, SouJava, Oracle, Hazelcast, Fujitsu, Microsoft...
- Thorntail is Red Hat's implementation
- Minimum footprint for Enterprise Java cloud-native services (v3.1) :

JSON-P 1.1	JSON-B 1.0	Health 2.1	JWT Propagation 1.1	Config 1.3	OpenAPI 1.1
CDI 2.0	JAX-RS 2.1	Fault Tolerance 2.0	Metrics 2.1	Open Tracing 1.3	Rest Client 1.3

Red Hat

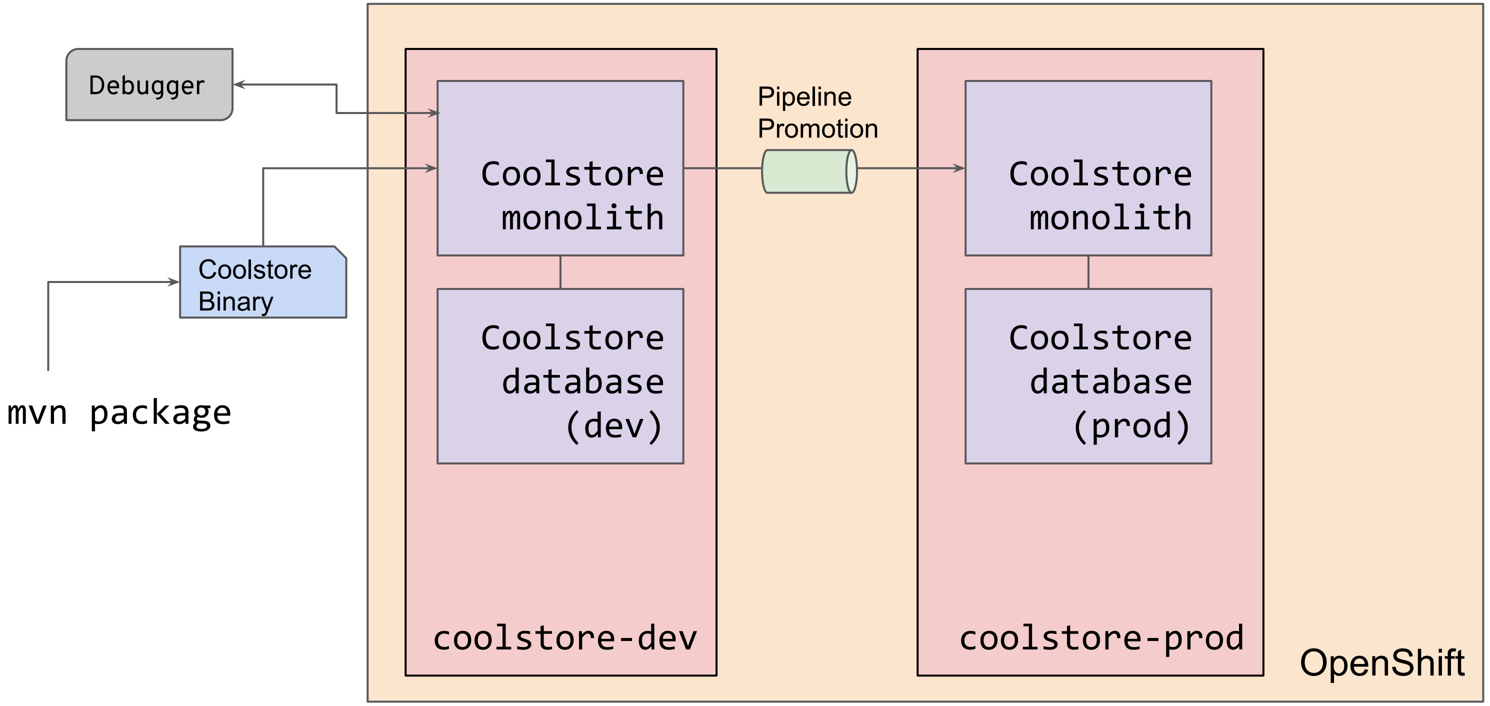
Lab: Monoliths to microservices with MicroProfile & Spring Boot

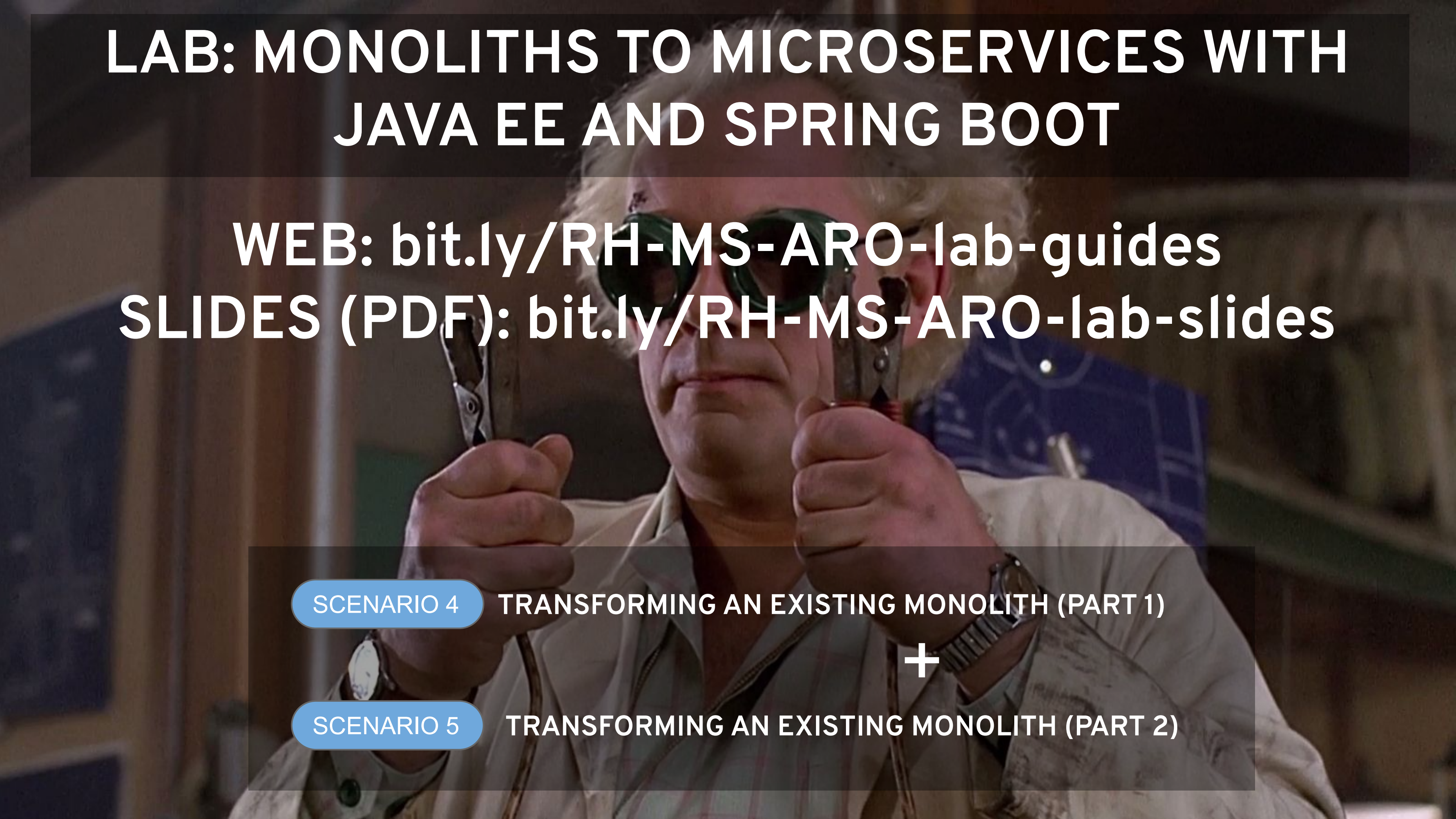
Goal for lab

In this lab you will learn:

- How Red Hat OpenShift and Red Hat Runtimes help jumpstart app modernization
- Benefits and challenges of microservices
- How to transform existing monolithic applications to microservices using strangler pattern and 12-factor app patterns.
- Use modern app dev frameworks like Thorntail and Spring Boot to implement microservice applications on OpenShift

Current state - the monolith





LAB: MONOLITHS TO MICROSERVICES WITH JAVA EE AND SPRING BOOT

WEB: bit.ly/RH-MS-ARO-lab-guides
SLIDES (PDF): bit.ly/RH-MS-ARO-lab-slides

SCENARIO 4 TRANSFORMING AN EXISTING MONOLITH (PART 1)

+

SCENARIO 5 TRANSFORMING AN EXISTING MONOLITH (PART 2)

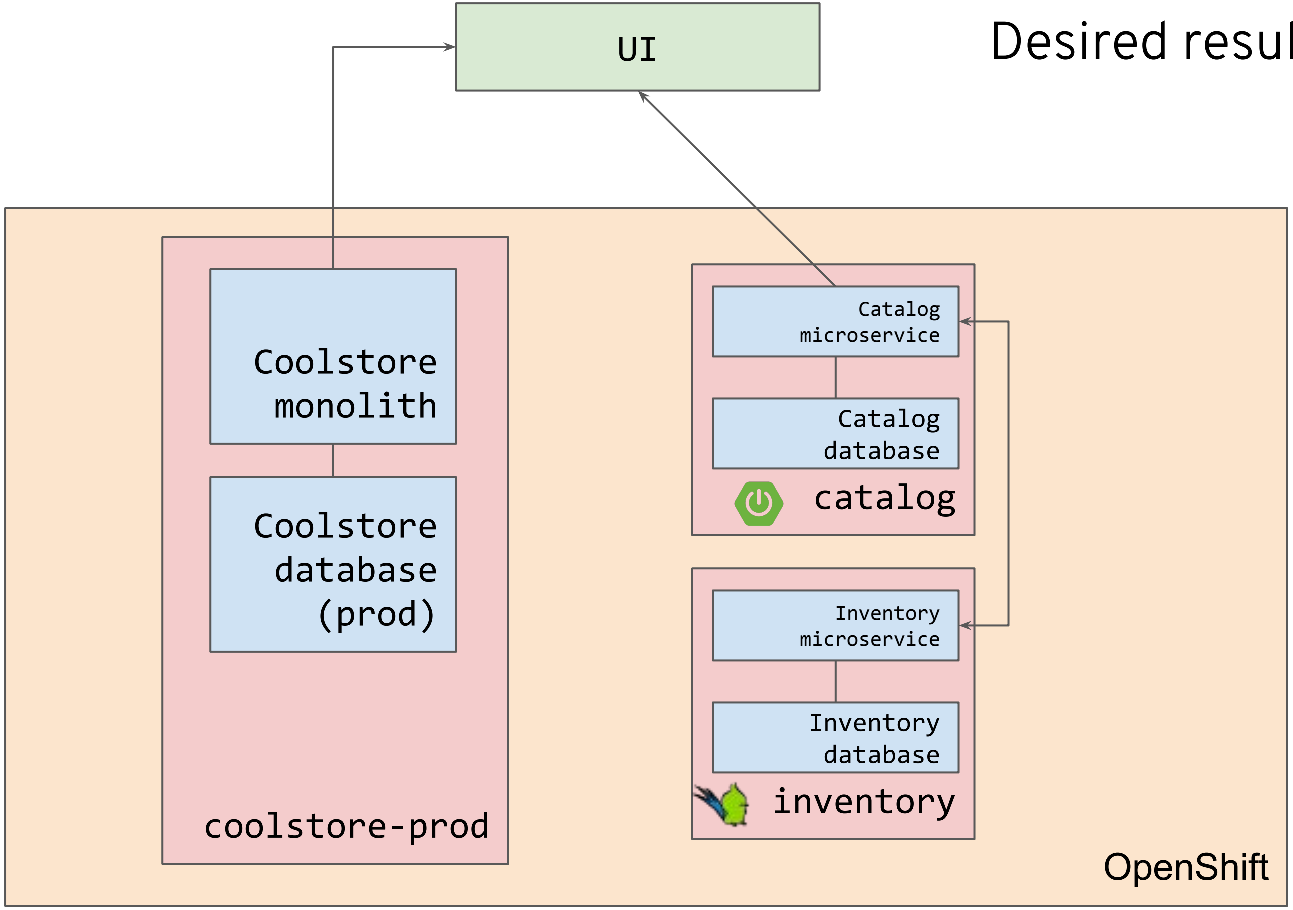
Wrap-up and discussion

Result of lab

In this lab you learned how to:

- Implement a Java EE microservice using Thorntail
- Implement a Java EE microservice using Spring Boot
- Develop container-based testing
- Add microservice concerns like Health checks, externalized configuration and circuit breaking
- Use the strangler pattern to slowly migrate functionality from monolith to microservices

Desired result of lab



Thank you

Red Hat

LinkedIn: linkedin.com/company/red-hat

YouTube: youtube.com/user/RedHatVideos

Facebook: facebook.com/redhatinc

Twitter: twitter.com/RedHatNews

Google+: plus.google.com/+RedHat

Microsoft Azure

LinkedIn: linkedin.com/company/microsoft/

YouTube: youtube.com/user/MSCloudOS

Facebook: facebook.com/microsoftazure/

Twitter: twitter.com/azure

Azure Friday: channel9.msdn.com/Shows/Azure-Friday

Azure | Channel 9: channel9.msdn.com/Blogs/Azure