redhat.

Microsoft Azure

MONOLITHS TO MICROSERVICES: APP TRANSFORMATION

Hands-on Technical Workshop

PART 3: MONOLITHS TO MICROSERVICES WITH MICROPROFILE AND SPRING BOOT

WHY MONOLITH TO MICROSERVICES

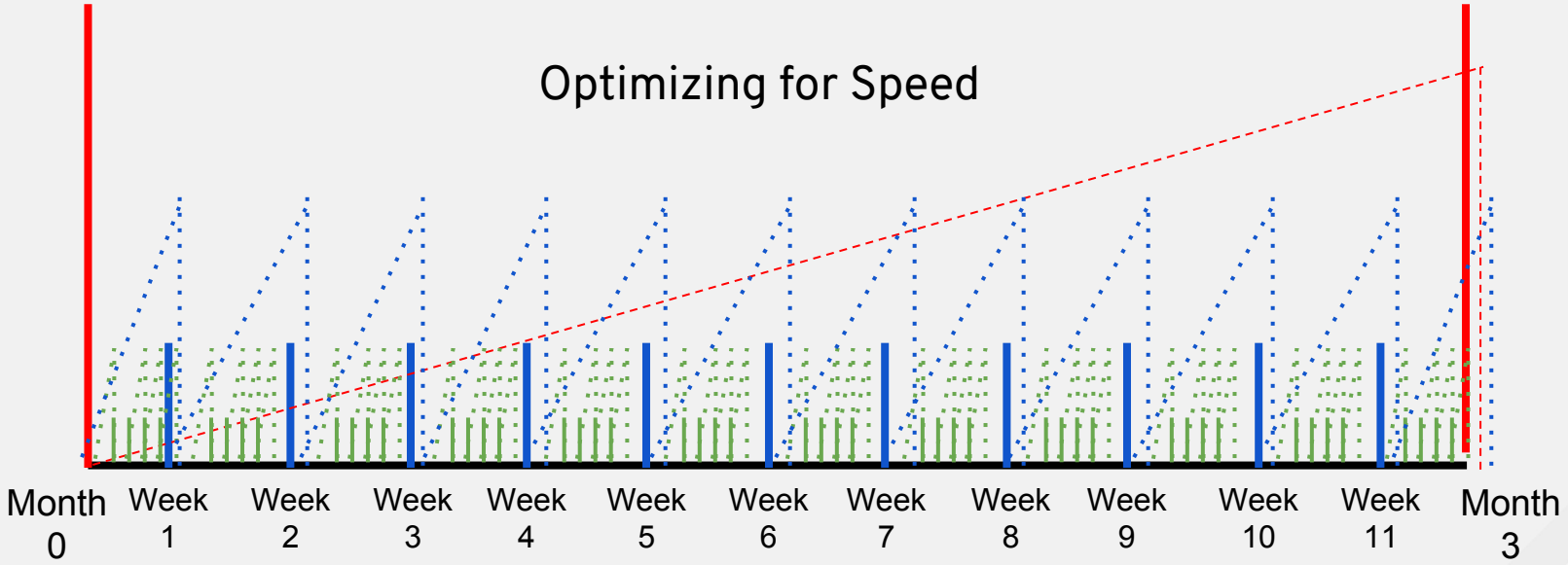
Break things down (organizations, teams, IT systems, etc) down into smaller pieces for greater parallelization and autonomy and focus on reducing time to value.

REDUCING TIME TO VALUE

Monolith Lifecycle

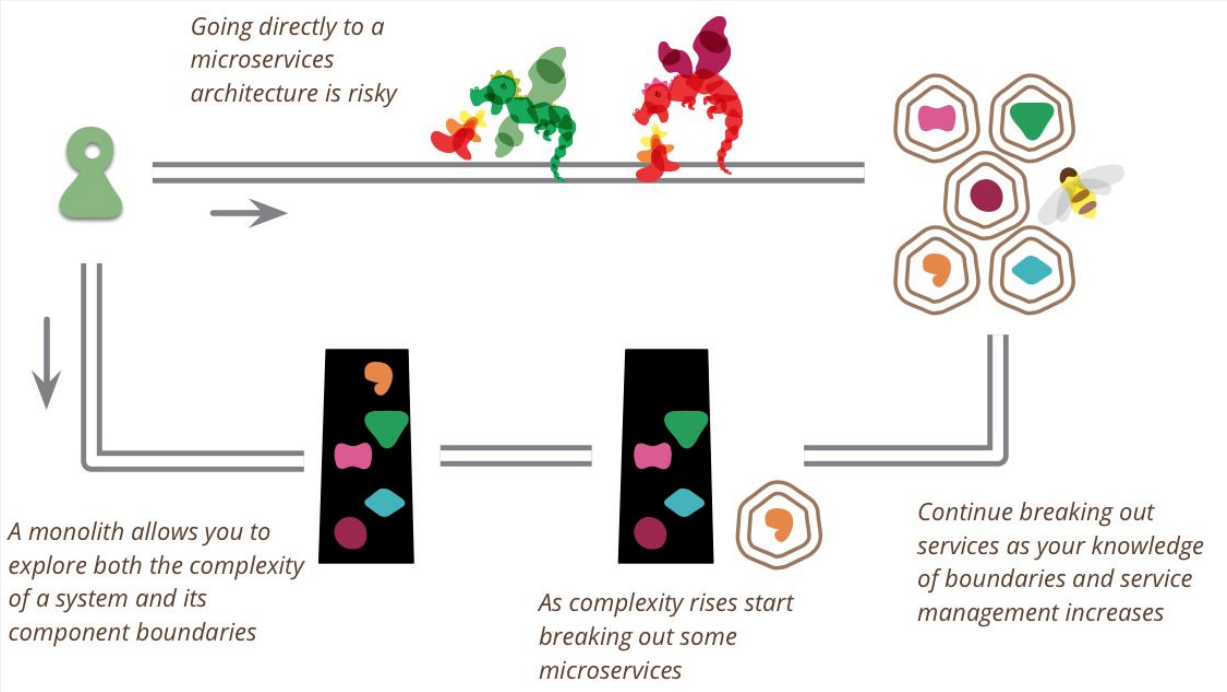
Fast Moving Monolith

Microservices



redhat.

Monolith First?



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

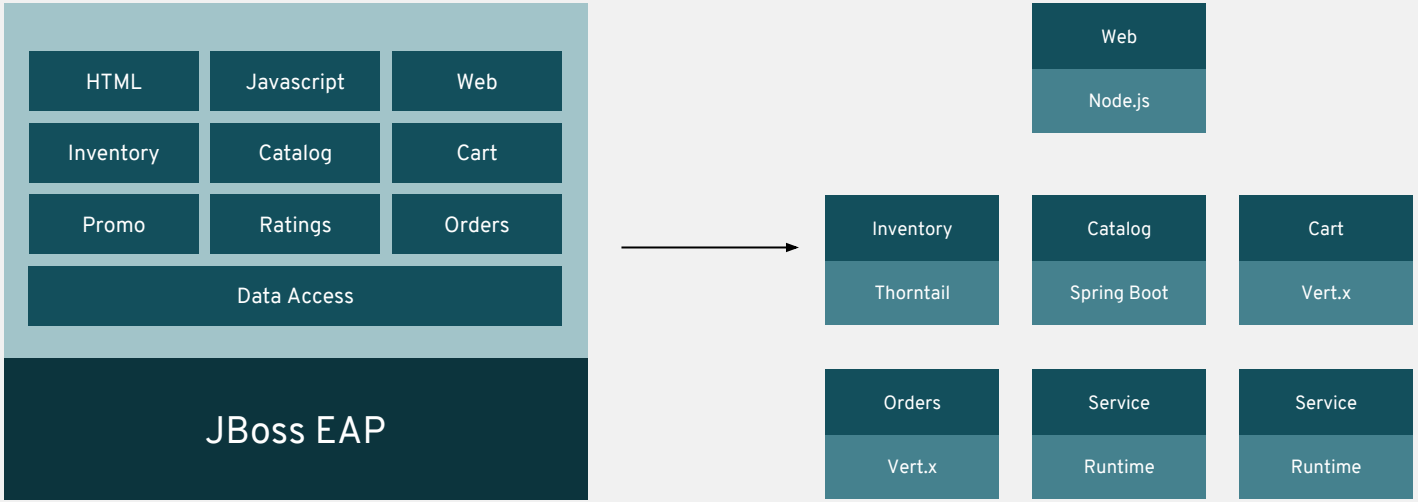
THE BIGGER PICTURE: THE PATH TO CLOUD-NATIVE APPS

A DIGITAL DARWINISM

redhat.

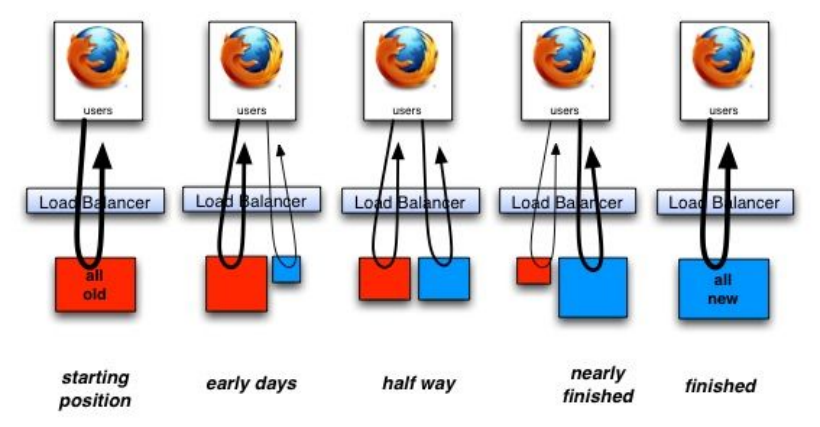
STRANGLING THE MONOLITH

- 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.



Time →

RED HAT®
APPLICATION RUNTIMES

RED HAT PLATFORM FOR THE HYBRID CLOUD

OPENSHIFT AND MIDDLEWARE OPTIMIZED FOR THE CLOUD

Infra Monitoring

APPLICATION RUNTIMES

- RED HAT OPENSHIFT Application Runtimes
- 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 APPLICATION RUNTIMES

NON-RESTRICTIVE DEVELOPMENT FOR THE HYBRID CLOUD

LAUNCH SERVICE

RED HAT OPENSHIFT Application Runtimes

JAVA SE OPENJDK	JAVA EE JBOSS EAP	JAVA WEB JBOSS WS
SERVERLESS CLOUD FUNCTIONS*	SPRING SPRING BOOT	JAVASCRIPT NODE.JS
MICROPROFILE THORNTAIL	REACTIVE VERT.X	

DISTRIBUTED DATA
DATA GRID

MESSAGING
AMQ BROKER

SECURITY RED HAT SSO

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

*Coming Soon

ENTERPRISE JAVA

RED HAT JBOSS
ENTERPRISE
APPLICATION PLATFORM

SERVLET APPS

JAVA MICROSERVICES

JAVASCRIPT FLEXIBILITY

REACTIVE SYSTEMS

TOMCAT SIMPLICITY

RED HAT JBOSS
WEB SERVER

SPRING

redhat

- 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 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 (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)

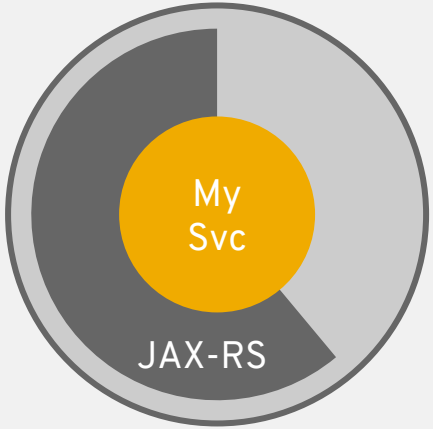
THORNTAIL (née WILDFLY SWARM)

redhat

THORNTAIL

Java EE microservices

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

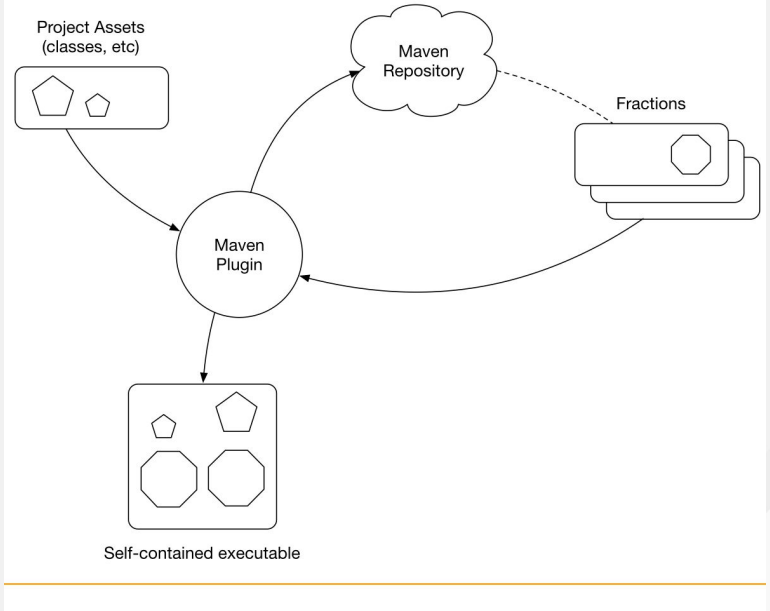



```
$ java -jar my_microservice.jar
```

MICROPROFILE™

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 MICROPROFILE™
- API Documentation

THORNTAIL

Build microservices

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

MICROPROFILE™

RED HAT® OPENSHIFT Application Runtimes

Thorntail and RHOAR

Upstream (Unsupported)

- Flyway
- JMS
- Jolokia
- Logstash
- Vert.x Integration
- Infinispan
- Fluentd
- Consul
- jGroups
- Swagger
- Spring

Tested and Verified

- Hystrix
- Ribbon
- MySQL
- Oracle DB

Additional Supported Fractions

- Metrics
- Health
- Configuration
- Monitor
- Keycloak
- Topology

Supported Specifications

- Java EE 7 Web Profile
- MicroProfile 1.2

MICROPROFILE™

OPTIMIZING ENTERPRISE JAVA

- Defines **open source** Java **microservices** specifications
- Industry Collaboration - Red Hat, IBM, Payara, Tomitribe, London Java Community, SouJava, Oracle, Hazelcast, Fujitsu, SmartBear...
- **Thorntail** is **Red Hat's** implementation
- Minimum footprint for Enterprise Java cloud-native services (v2.0) :

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

LAB: MONOLITHS TO MICROSERVICES WITH MICROPROFILE AND SPRING BOOT

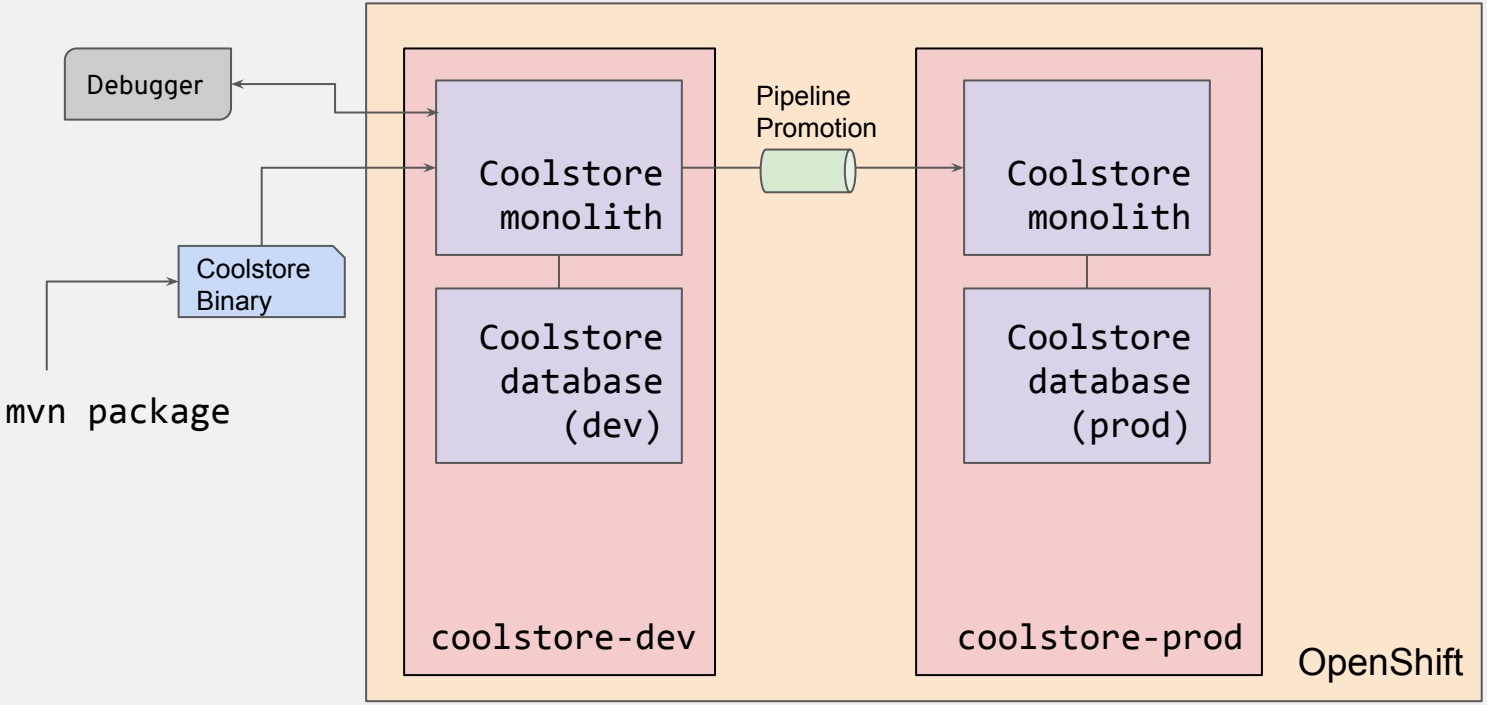
redhat.

GOAL FOR LAB

In this lab you will learn:

- How Red Hat OpenShift and Red Hat OpenShift Application Runtimes (RHOAR) 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-lab-guides
SLIDES (PDF): bit.ly/RH-MS-lab-slides

SCENARIO 4 TRANSFORMING AN EXISTING MONOLITH (PART 1)

+

SCENARIO 5 TRANSFORMING AN EXISTING MONOLITH (PART 2)

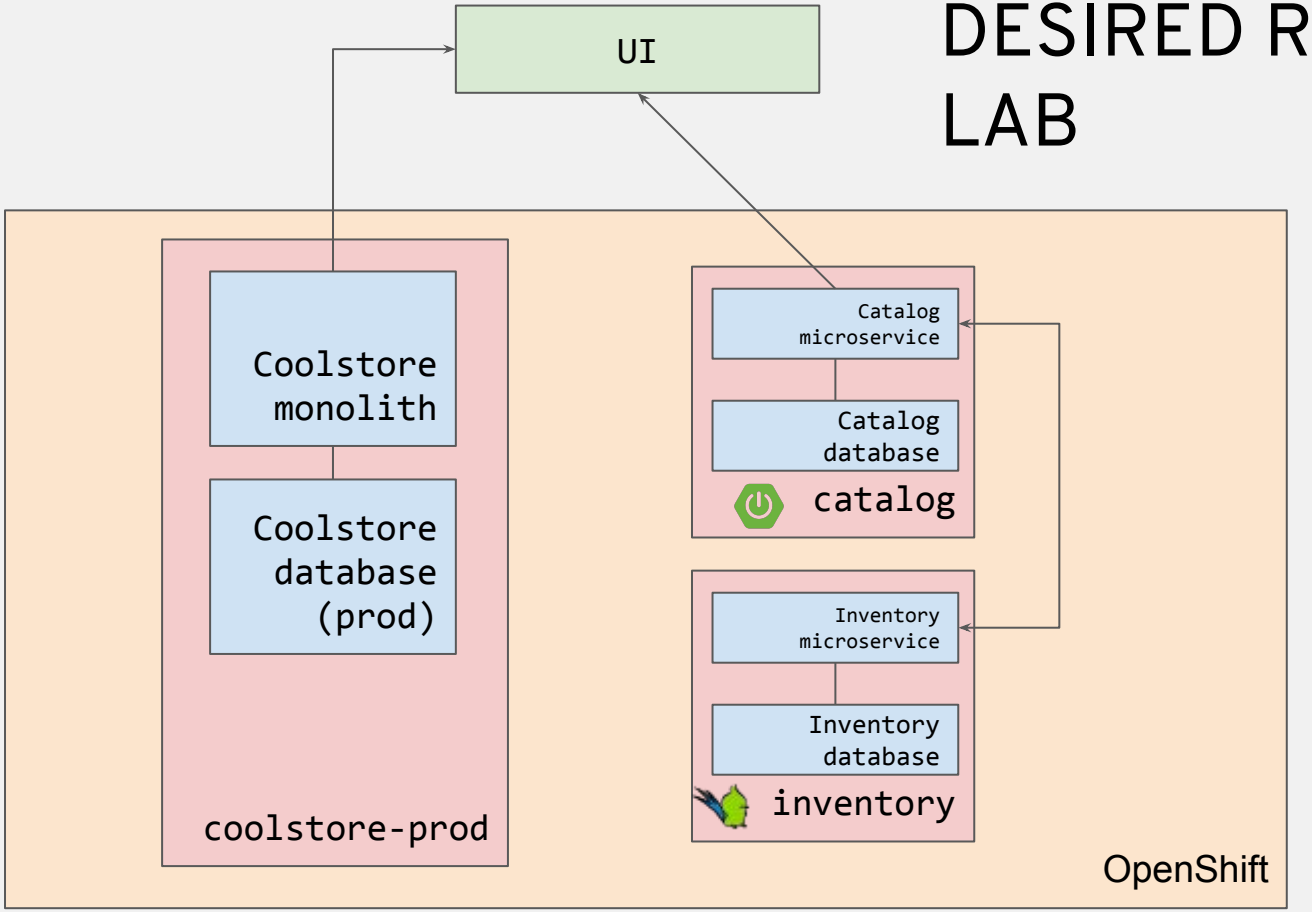
WRAP-UP AND DISCUSSION

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

LinkedIn: linkedin.com/company/red-hat

YouTube: youtube.com/user/RedHatVideos

Facebook: facebook.com/redhatinc

Twitter: twitter.com/RedHatNews

Google+: plus.google.com/+RedHat

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