Monoliths to microservices: App Transformation

Hands-on Technical Workshop

Part 5: Resilient distributed apps

Distributed services architectures

Benefits (when implemented correctly):

- Performance
- Reliability
- Resiliency
- Extensibility
- Availability
- Robustness

Distributed services architectures

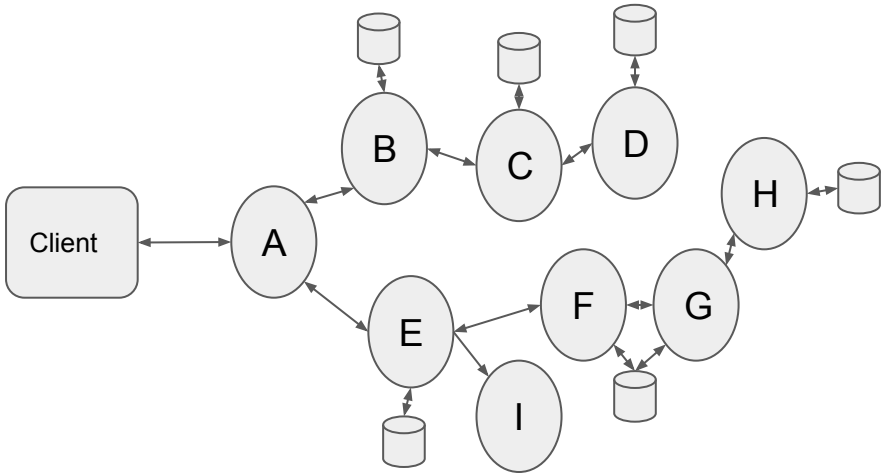
Fallacies of Distributed Computing

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

Distributed services architectures

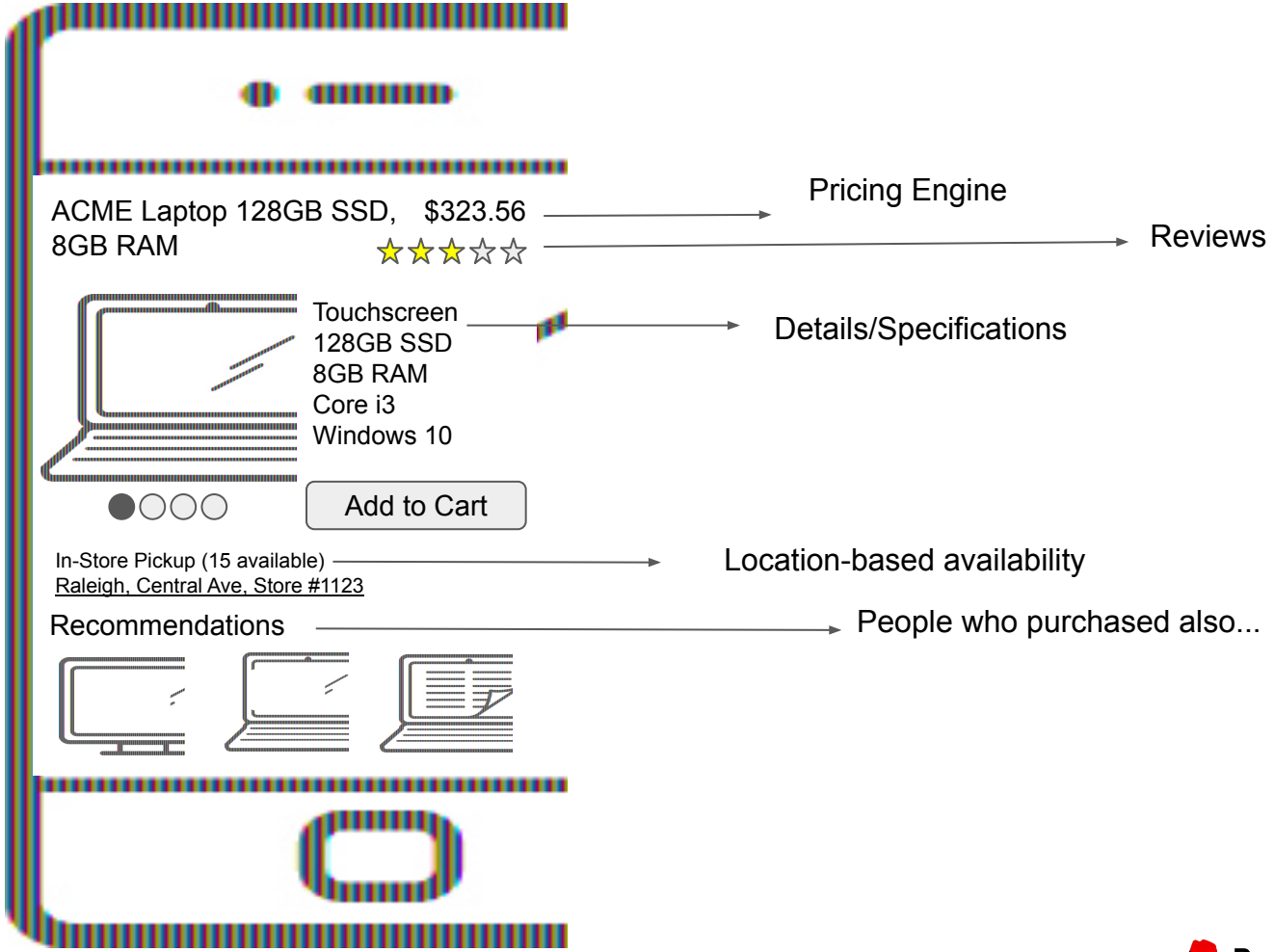
Applications must deal with

- Unpredictable failure modes
- End-to-end application correctness
- System degradation
- Topology changes
- Elastic/ephemeral/transient resources



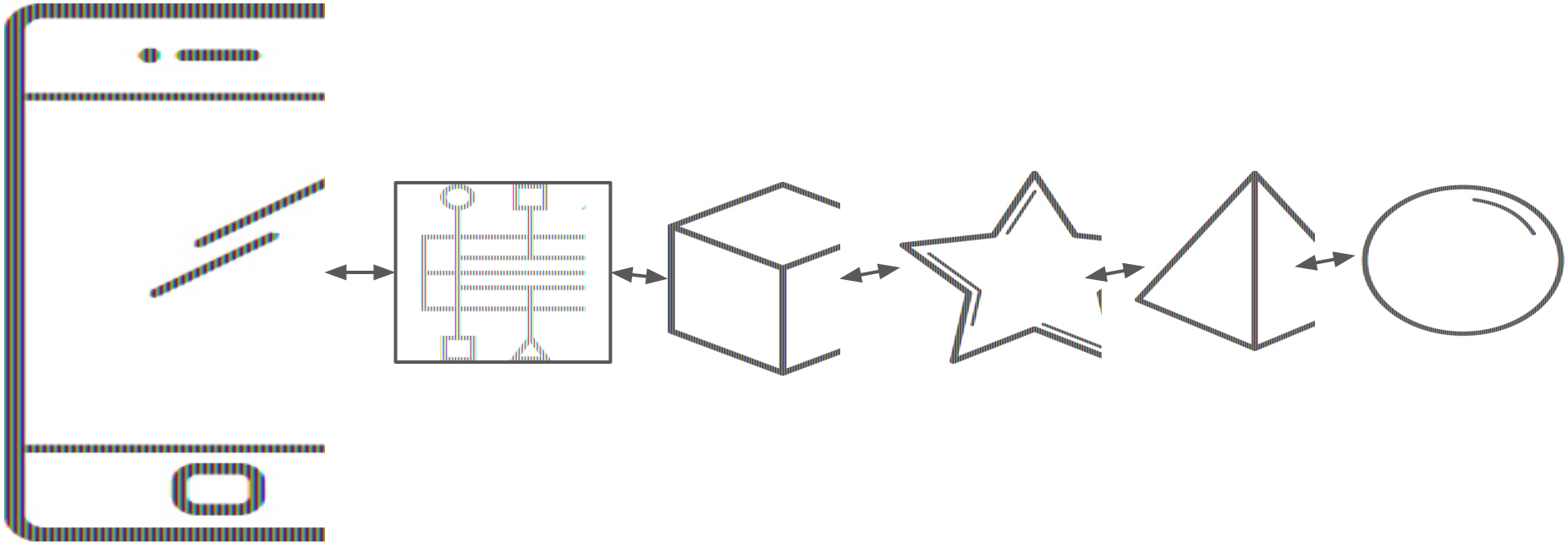
MICROSERVICES == DISTRIBUTED COMPUTING

Example

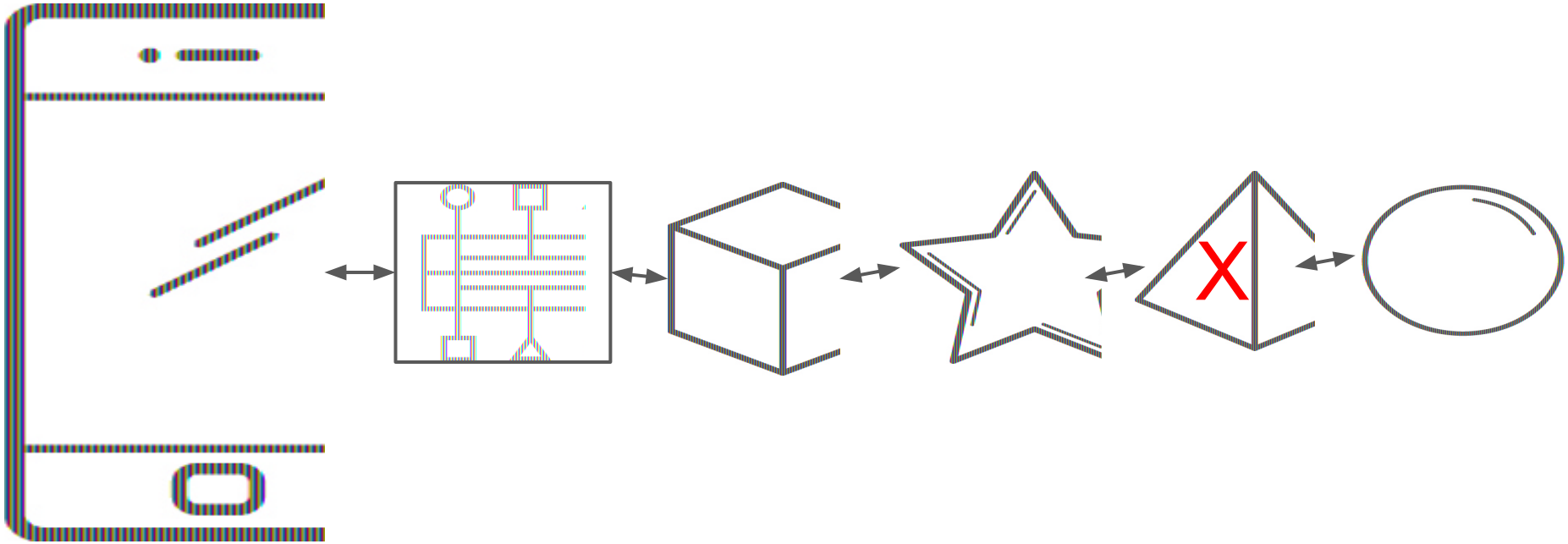


Red Hat

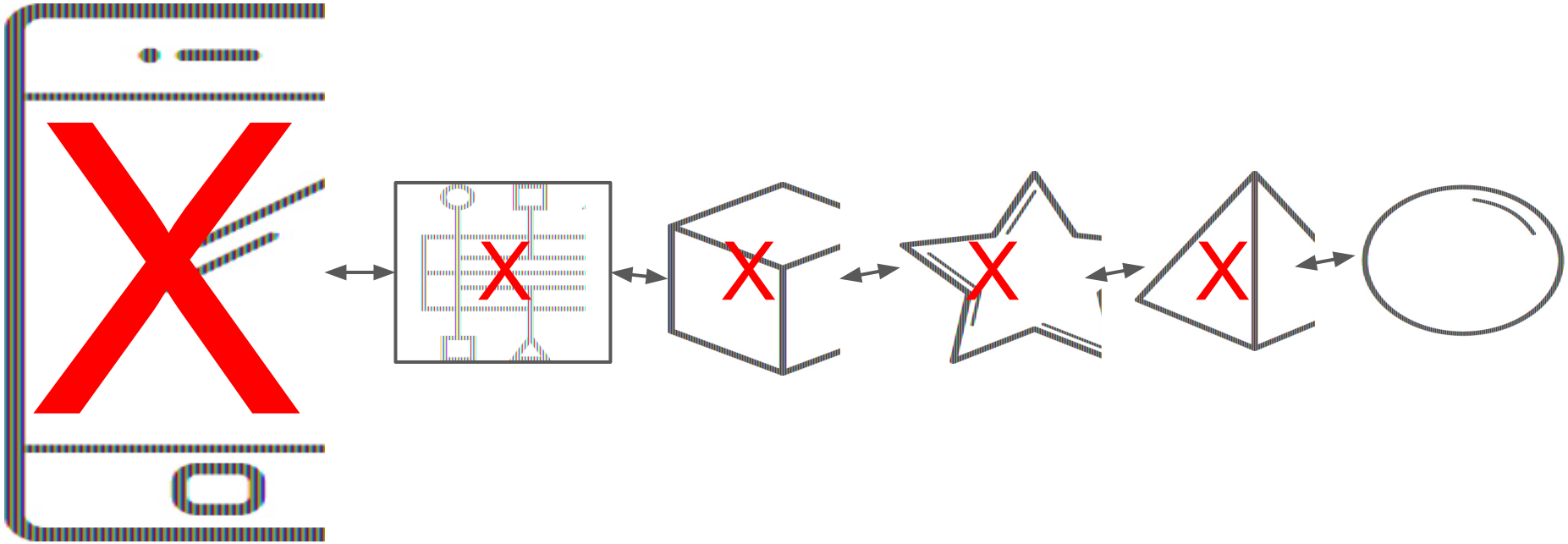
Chaining



Chaining (fail)



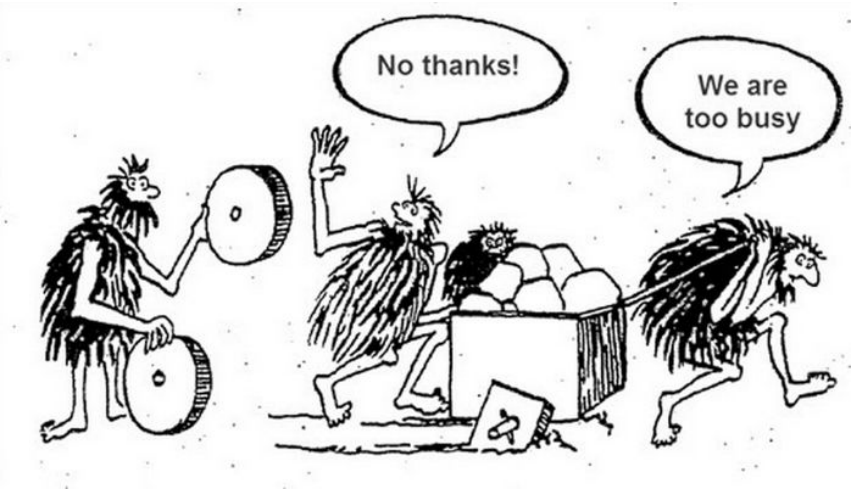
Chaining (cascading fail)



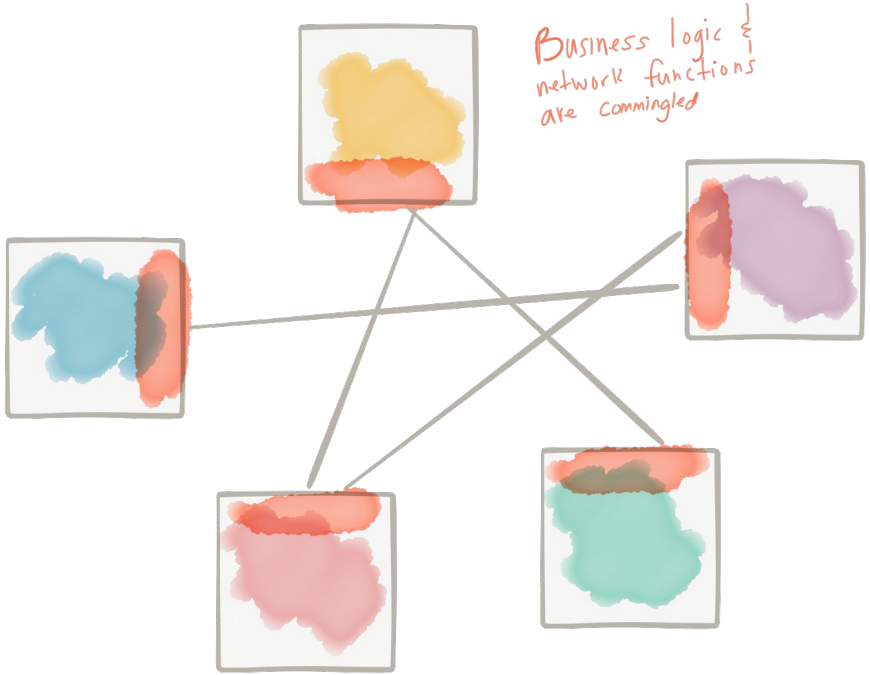
Possible solutions

Today, Developers do this:

- Circuit Breaking
- Bulkheading
- Timeouts/Retries
- Service Discovery
- Client-side Load Balancing



Too much infrastructure in business logic



But I'm using...


```
spring-cloud-netflix-hystrix
spring-cloud-netflix-zuul
spring-cloud-netflix-eureka-client
spring-cloud-netflix-ribbon
spring-cloud-netflix-atlas
spring-cloud-netflix-spectator
spring-cloud-netflix-hystrix-stream
...
@Enable....150MagicThings
```

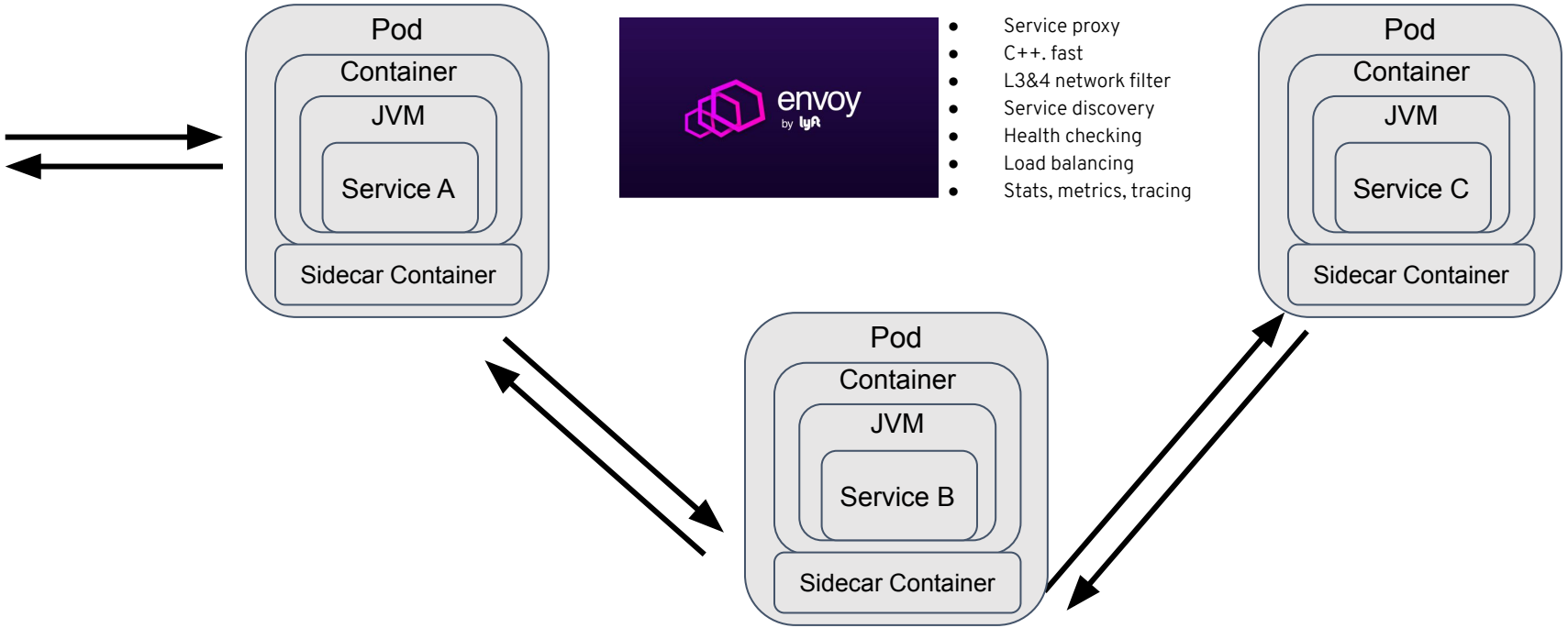
```
org.wildfly.swarm.hystrix
org.wildfly.swarm.ribbon
org.wildfly.swarm.topology
org.wildfly.swarm.camel-zookeeper
org.wildfly.swarm.hystrix
org.wildfly.swarm.hystrix
...
```

```
vertx-circuit-breaker
vertx-service-discovery
vertx-dropwizard-metrics
Vertx-zipkin
...
```

+ Node.js
+ Go
+ Python
+ Ruby
+ Perl
+ ....

Sidecars

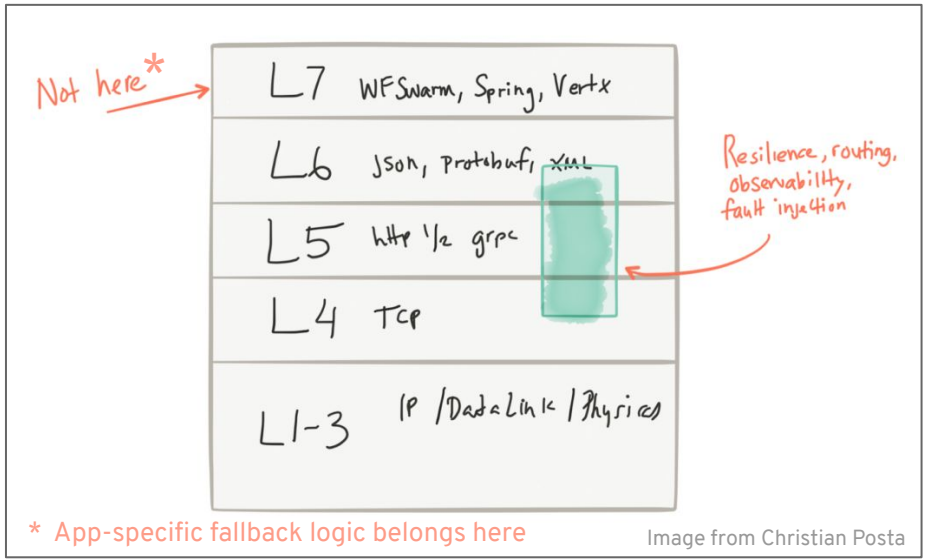
Pods with two containers



Istio - Sail

(Kubernetes - Helmsman or ship's pilot)

Istio - A robust service mesh for microservices



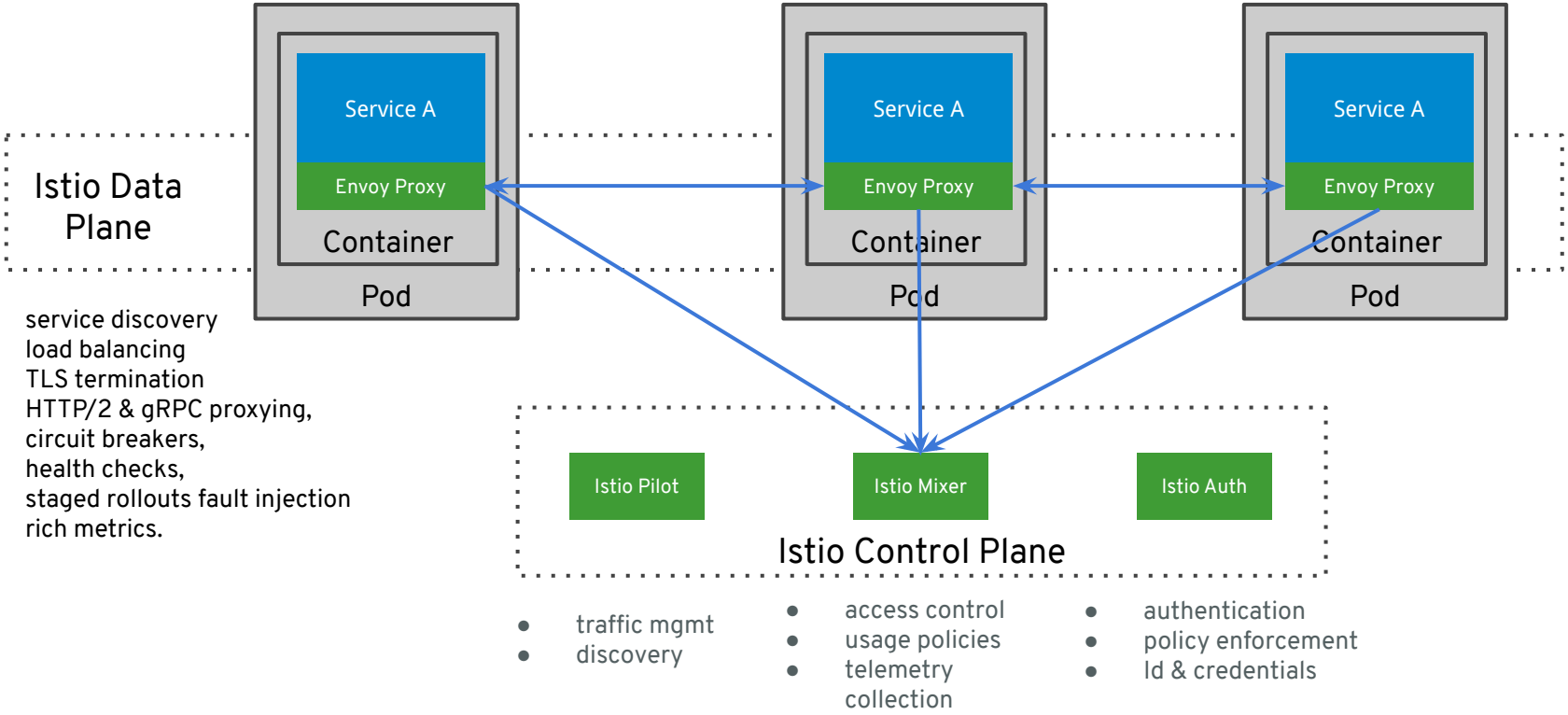
Key Features

- Intelligent routing and load balancing
- Fleet-wide, in-depth observability
- Resiliency across languages and platforms
- Fault injection
- Developer productivity
- Policy driven ops
- Circuit breaking, outlier detection
- Timeouts/retries
- Rate limiting
- Secure by default
- Incremental, unobtrusive adoption

Further Reading :

https://blog.openshift.com/red-hat-istio-launch/
https://istio.io/blog/istio-service-mesh-for-microservices.html
http://blog.christianposta.com/microservices/the-hardest-part-of-microservices-calling-your-services/

Istio - A robust service mesh for microservices



- service discovery
- load balancing
- TLS termination
- HTTP/2 & gRPC proxying,
- circuit breakers,
- health checks,
- staged rollouts fault injection
- rich metrics.

- traffic mgmt
- discovery

- access control
- usage policies
- telemetry collection

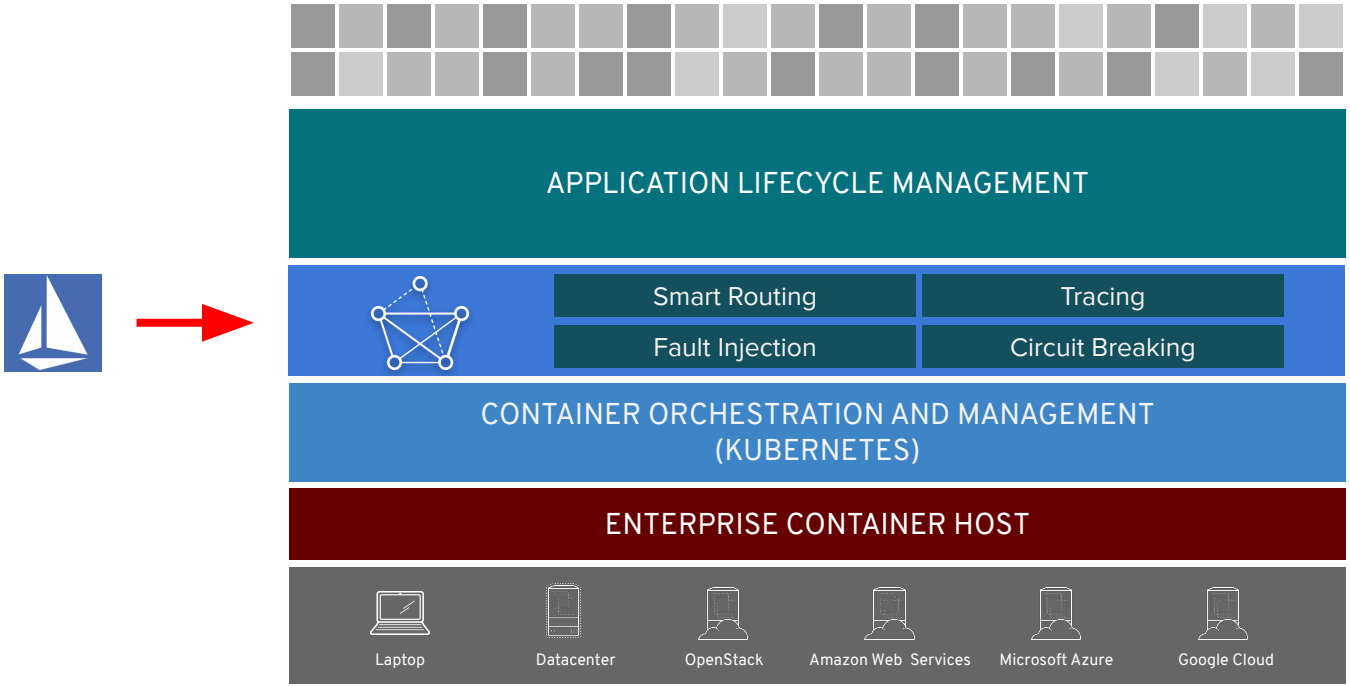
- authentication
- policy enforcement
- Id & credentials

Microservices 3.0 - Service mesh

Code Independent:

- Intelligent Routing and Load-Balancing
  - A/B Tests
  - Canary Releases
  - Dark Launches
- Distributed Tracing
- Circuit Breakers
- Fine grained Access Control
- Telemetry, metrics and Logs
- Fleet wide policy enforcement

App superpowers with istio



ANY
CONTAINER

Red Hat
OpenShift

ANY
INFRASTRUCTURE

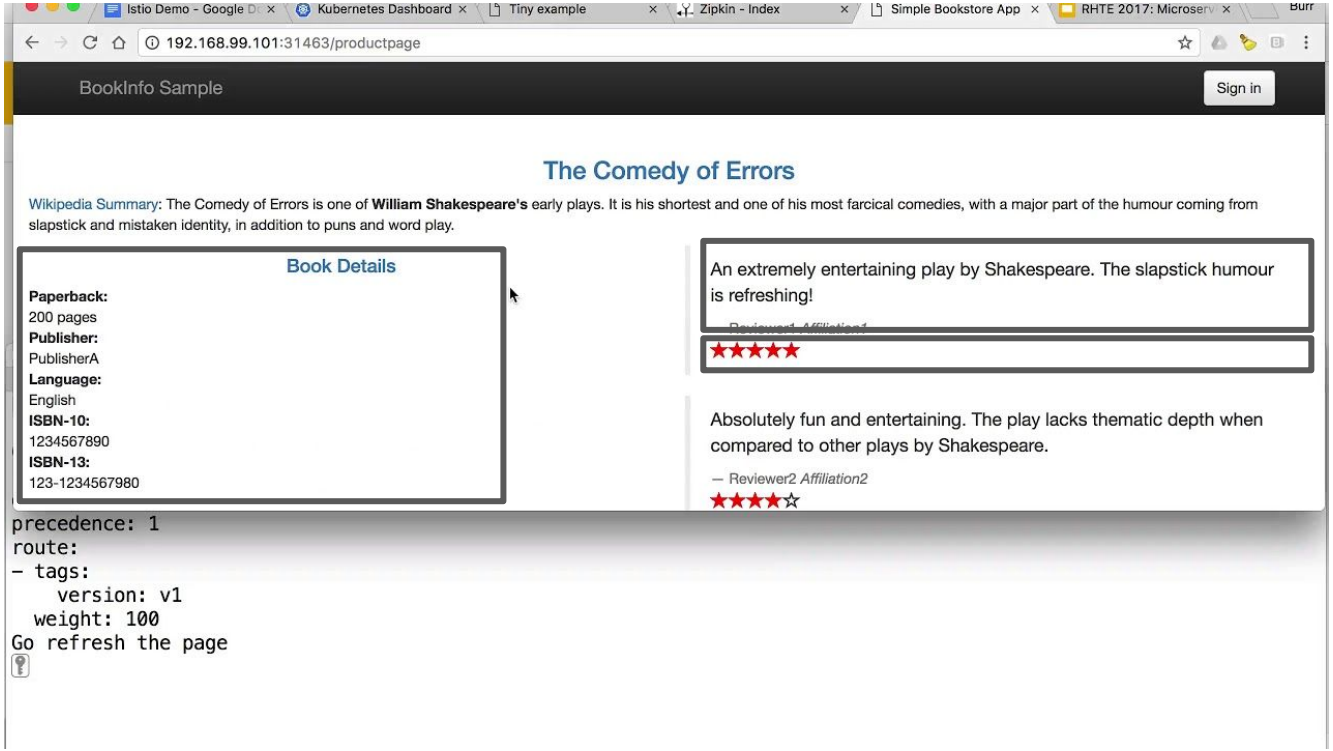
Lab: Detecting and preventing issues in distributed apps with istio

Goal for lab

In this lab you will learn:

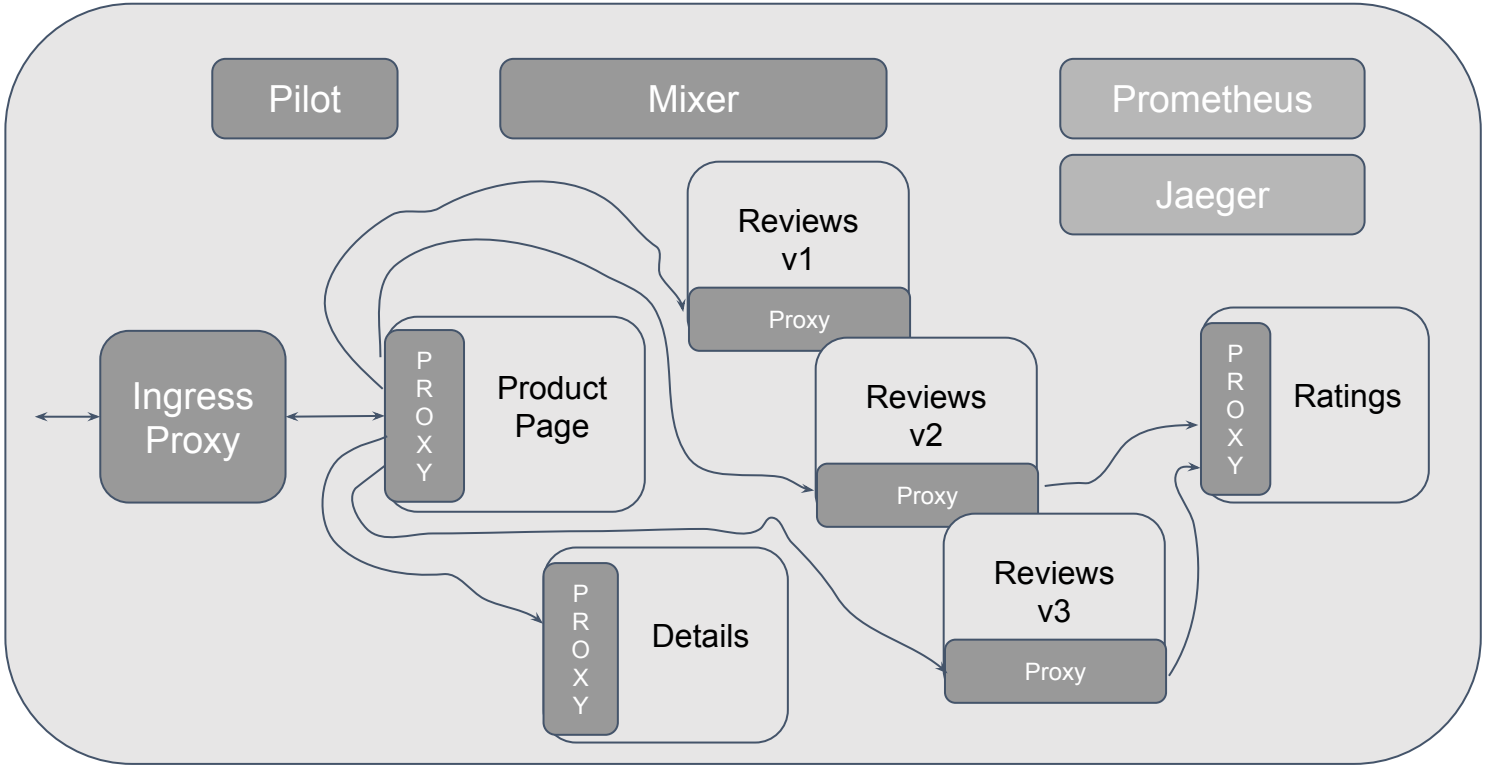
- How to **install Istio** onto **OpenShift Container Platform**
- How to deploy apps with **sidecar proxies**
- How to generate and visualize **deep metrics** for apps
- How to **alter routing** dynamically
- How to **inject faults** for testing
- How to do **rate limiting**
- How Istio implements **circuit breaking** and **distributed tracing**

Sample app: “BookInfo”



Red Hat

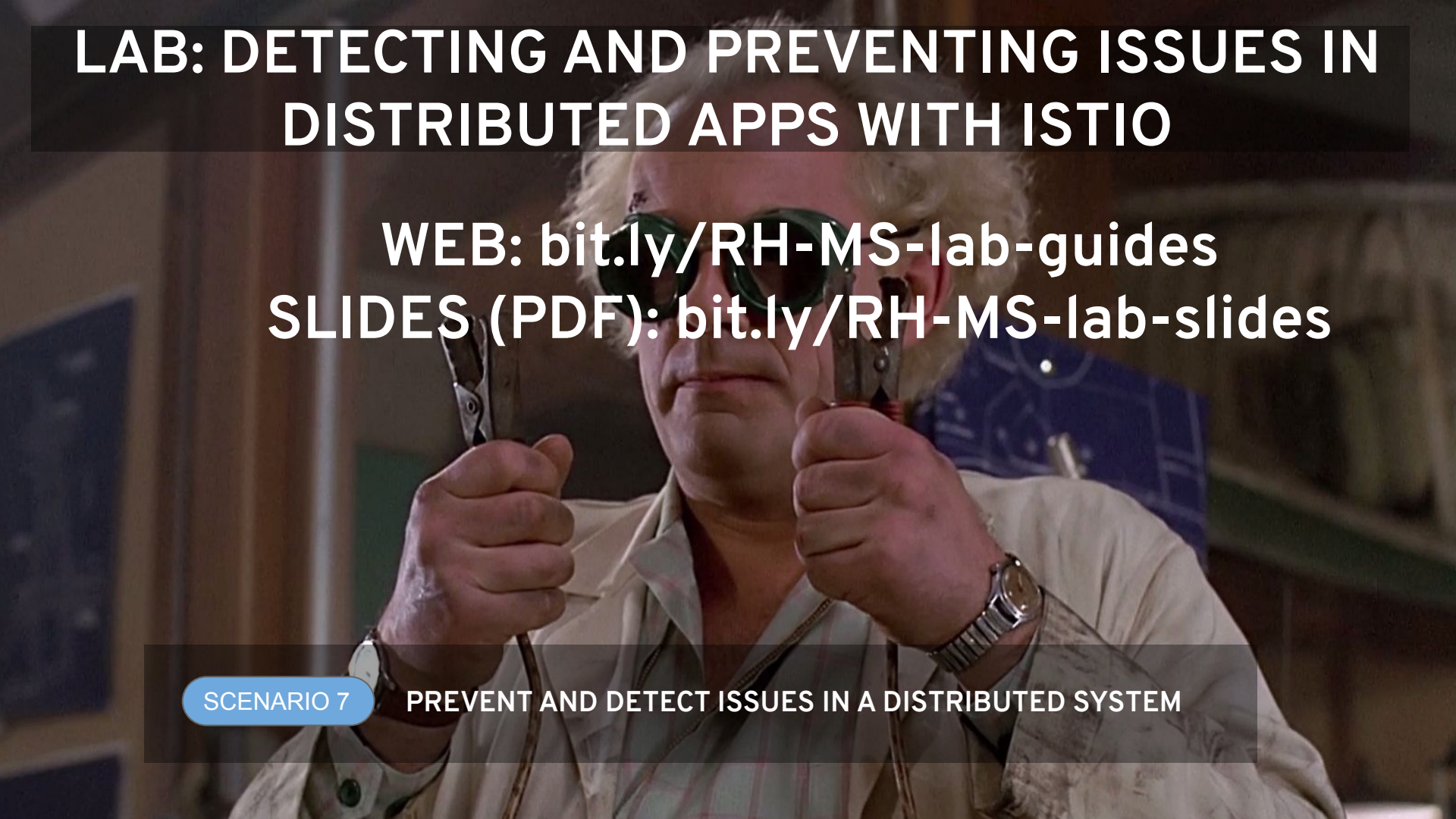
Sample app: “BookInfo”



Red Hat

LAB: DETECTING AND PREVENTING ISSUES IN DISTRIBUTED APPS WITH ISTIO

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



SCENARIO 7

PREVENT AND DETECT ISSUES IN A DISTRIBUTED SYSTEM

Wrap-up and discussion

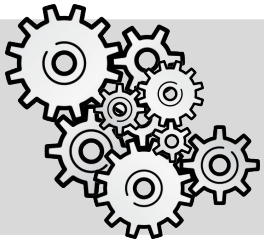
Result of lab

In this lab you learned:

- How to install Istio onto OpenShift Container Platform
- How to deploy apps with sidecar proxies
- How to generate and visualize deep metrics for apps
- How to alter routing dynamically
- How to inject faults for testing
- How to do rate limiting
- How Istio implements circuit breaking and distributed tracing
- Use cases for service mesh

Microservices 4.0?

Service



> Autonomous
> Loosely-coupled

Microservice

> Single Purpose
> Stateless
> Independently Scalable
> Automated

Function

> Single Action
> Event-sourced
> Ephemeral

Serverless projects/services

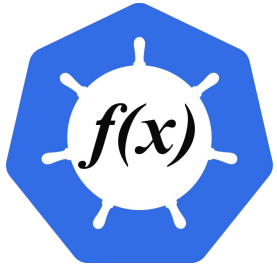
syncano

webtask

APACHE OpenWhisk™

APEX | SERVERLESS INFRASTRUCTURE

Microsoft Azure



Back&

http://funcatron.org

fission

SERVERLESS

CLOUD FUNCTIONS BETA

stdlib

serverless-docker

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