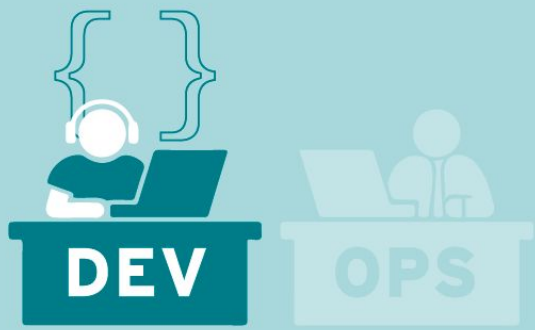
Red Hat



The Containers and Cloud-Native Roadshow Developer Track Lab Guide

A hands-on experience for Ops and Dev professionals

- $ MAN WHOAMI?
- WHAT I WANT TO LEARN
- FUN FACT (OPTIONAL)

Laine Vyvyan

Josh Smith

Red Hat

DEVELOPER TRACK MODULES

1 OPTIMIZING EXISTING APPLICATIONS

Migrate an existing monolithic Java application from a legacy platform to Red Hat.

Modernize by incrementally refactoring to microservices architecture and modern Java platform

2 DEBUGGING, MONITORING AND CONTINUOUS DELIVERY

Debug, instrument and monitor a modern microservice application.

Deploy continuously using Pipelines

3 CONTROL CLOUD NATIVE APPS WITH SERVICE MESH

Gain a deep understanding of app behavior through service mesh instrumentation and visualization

4 ADVANCED CLOUD NATIVE WITH EVENT-DRIVEN SERVERLESS

Dynamically respond to events and scale applications using powerful Kubernetes constructs

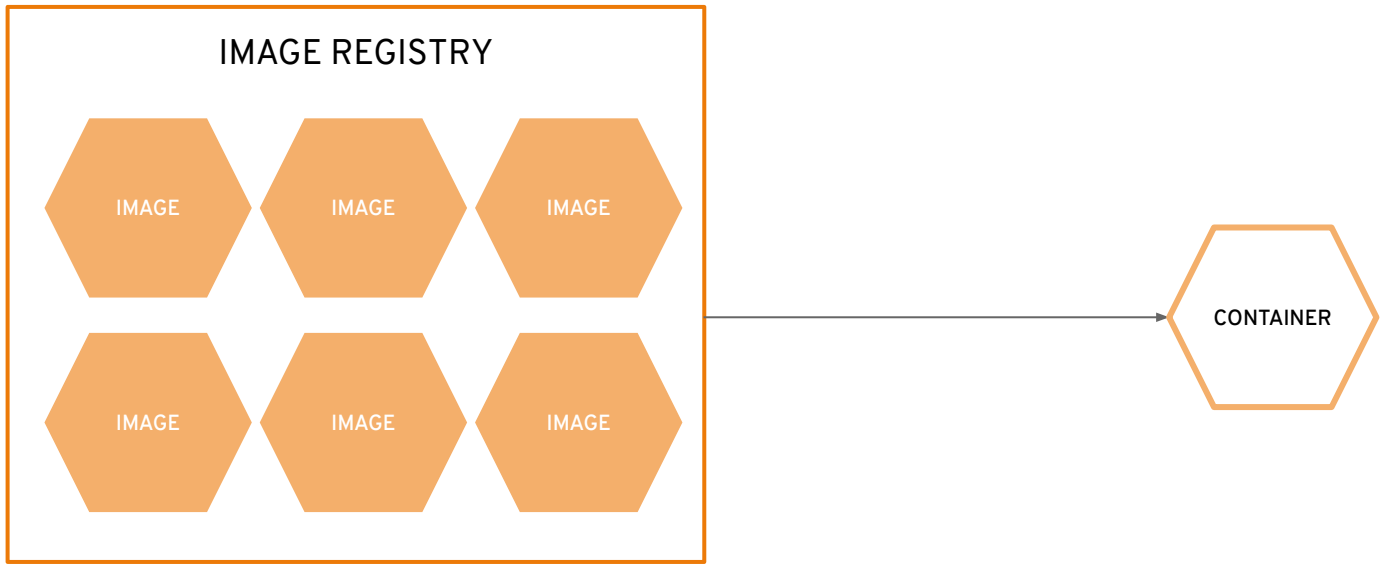
Red Hat

OpenShift Concepts

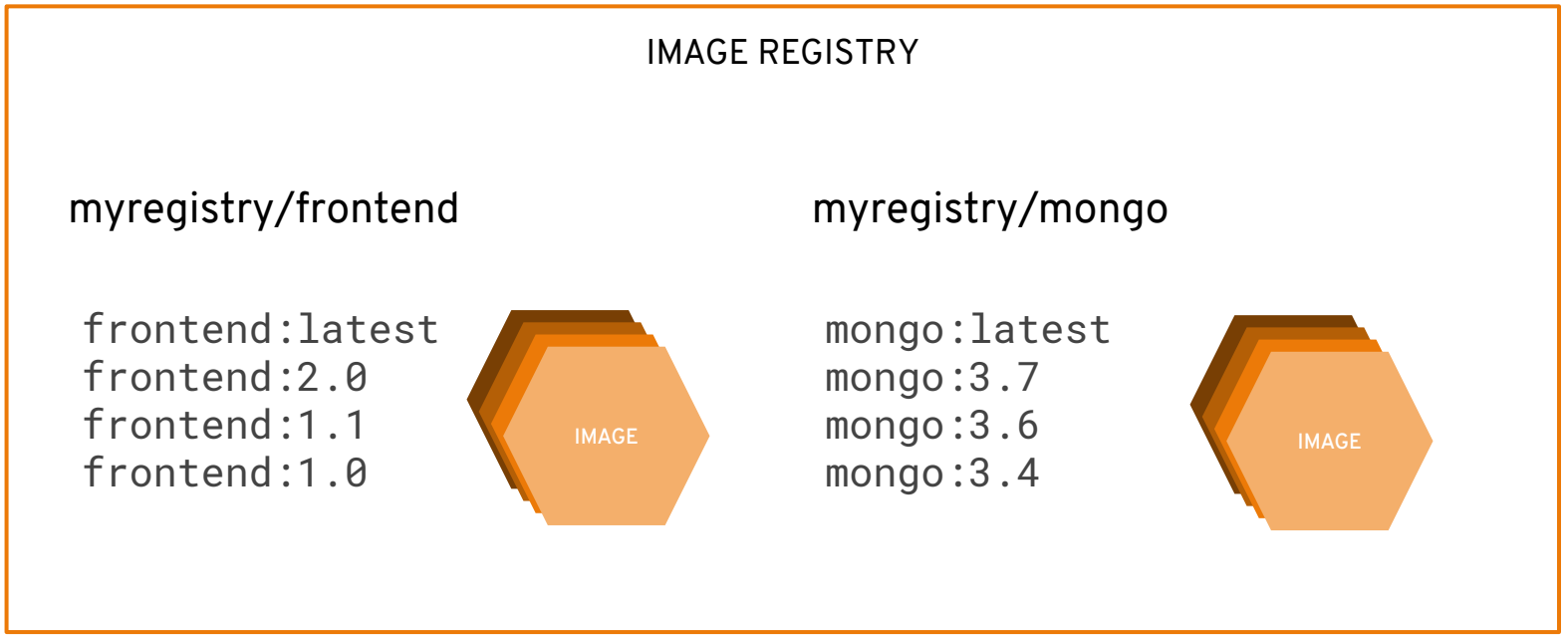
a container is the smallest compute unit

containers are created from container images

container images are stored in an image registry

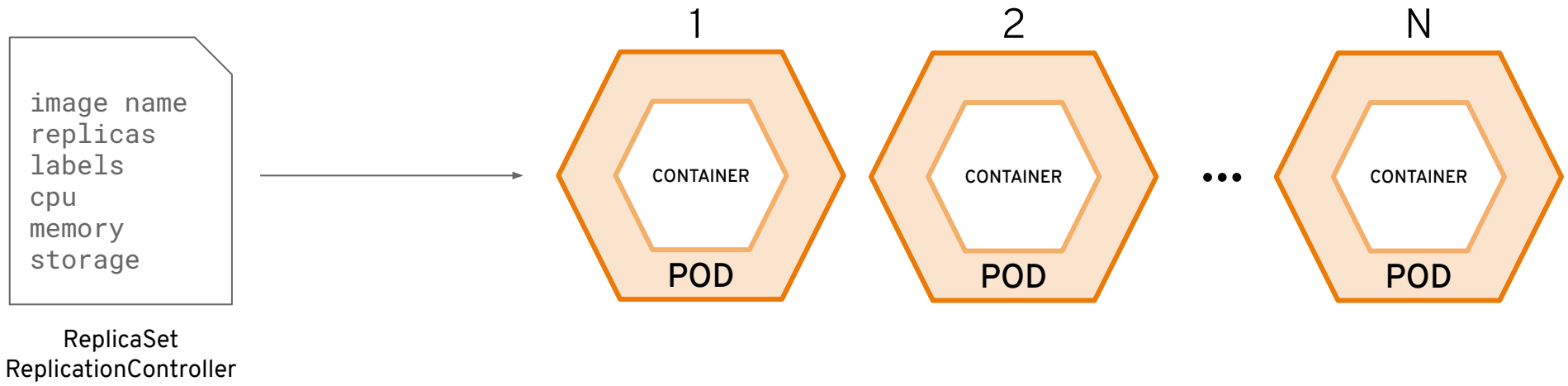


an image repository contains all versions of an image in the image registry

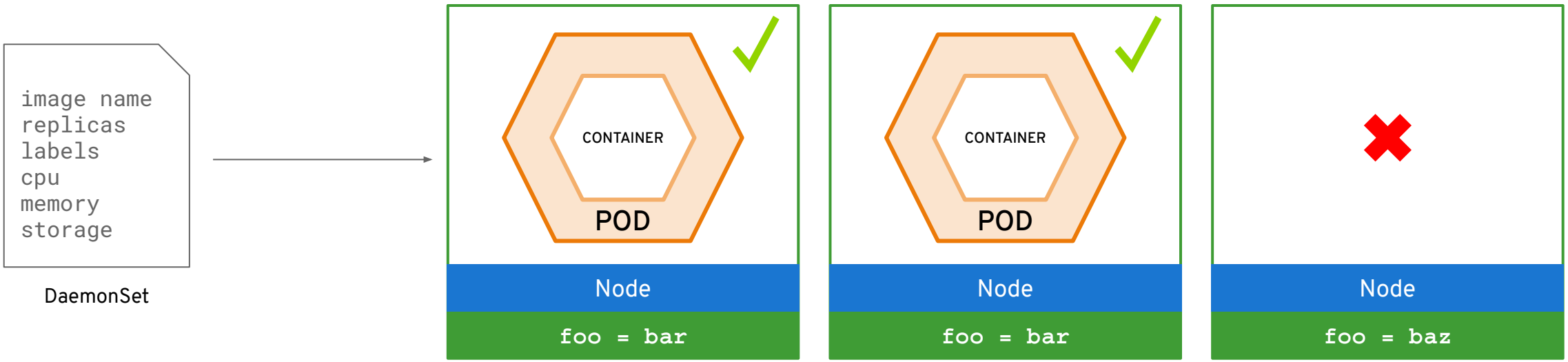


containers are wrapped in pods which are units of deployment and management

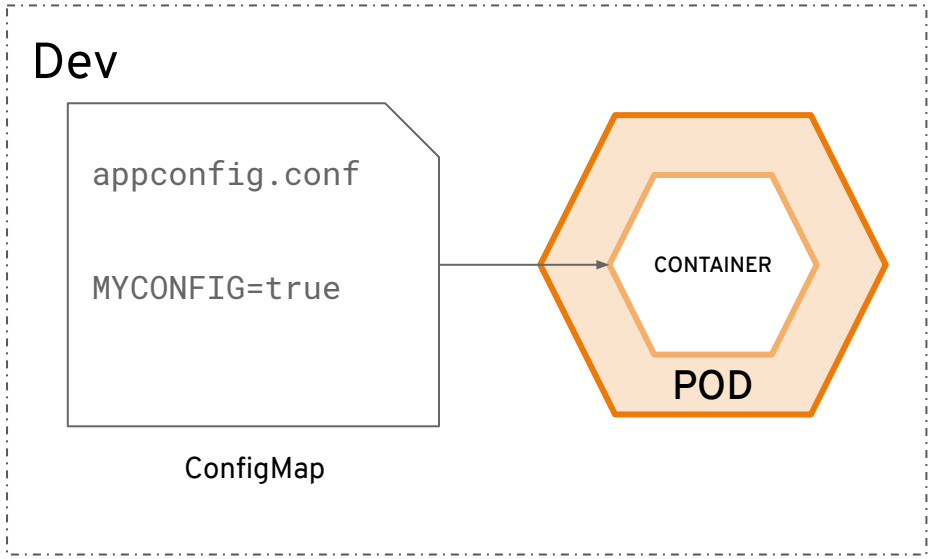
`ReplicationControllers` & `ReplicaSets` ensure a specified number of pods are running at any given time

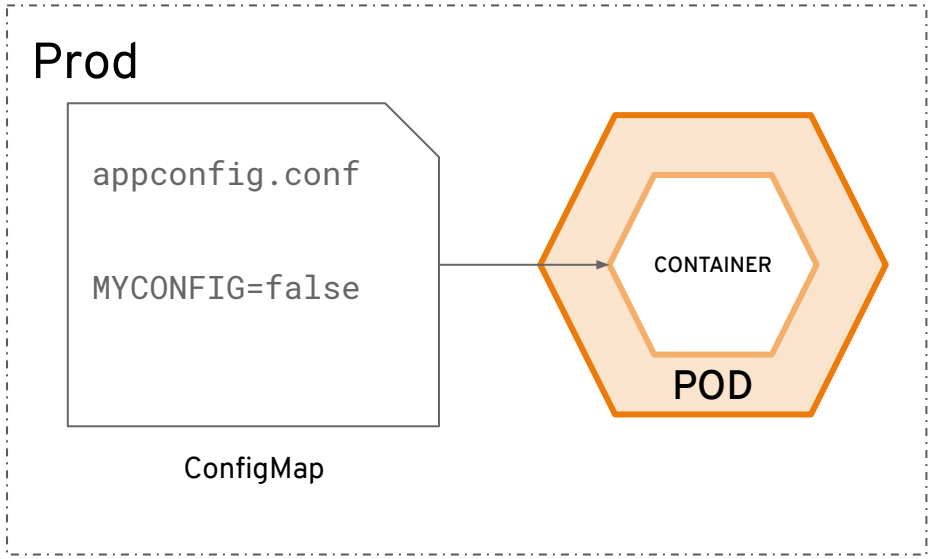


a daemonset ensures that all (or some) nodes run a copy of a pod

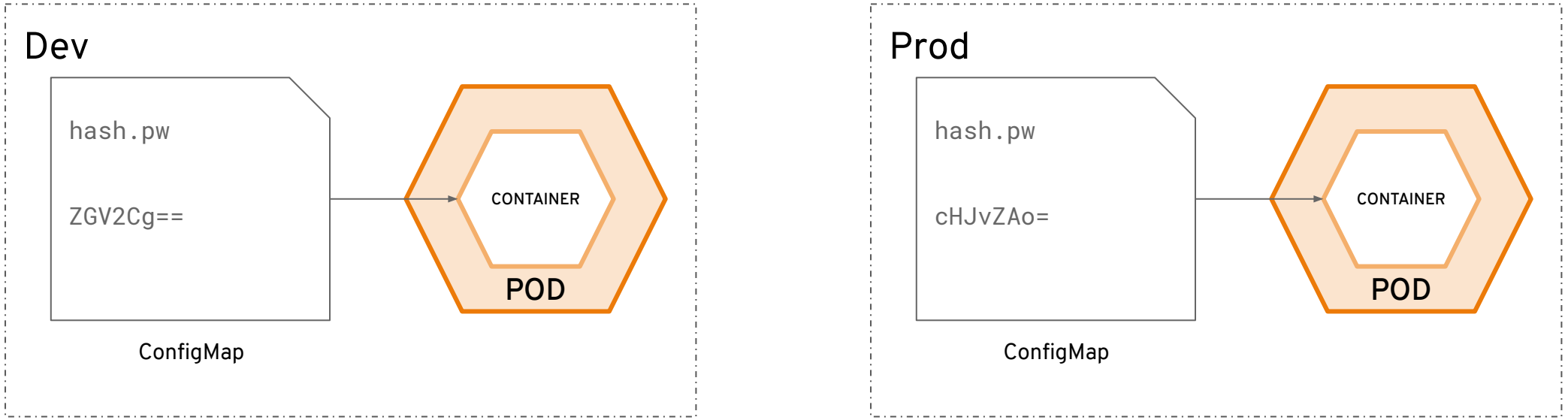


`configmaps` allow you to decouple configuration artifacts from image content

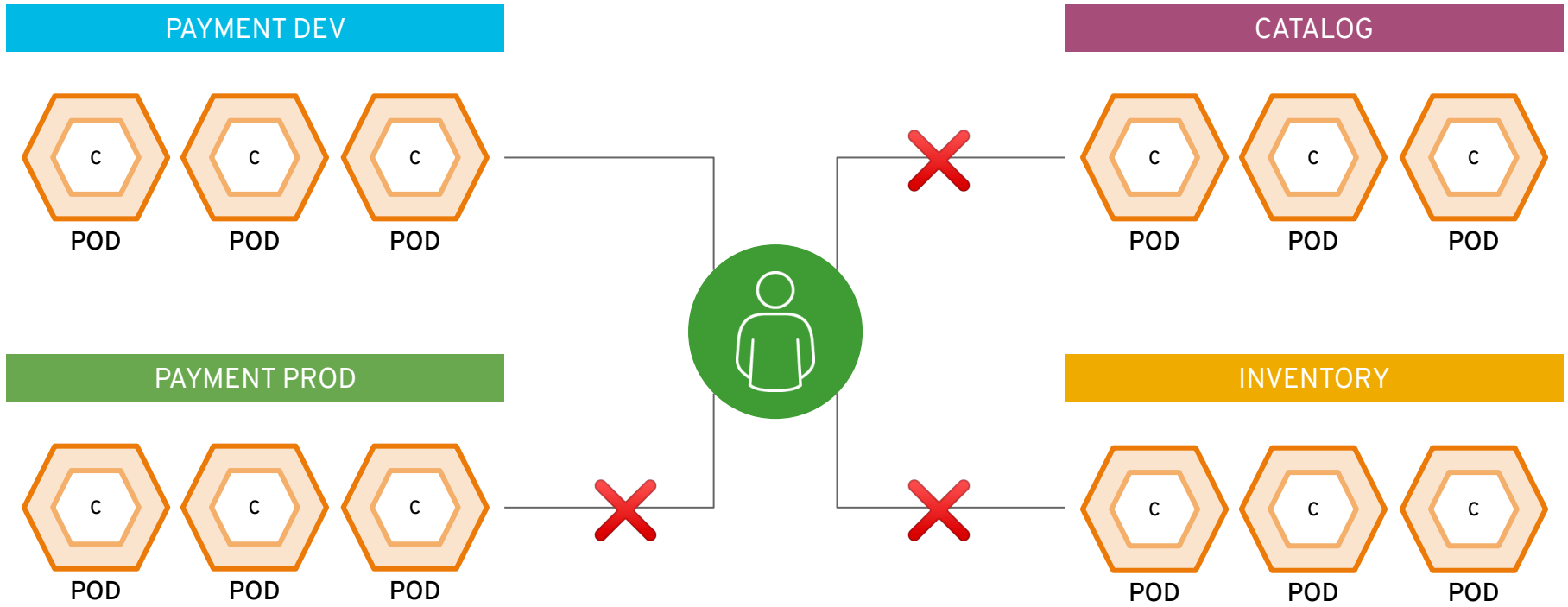




`secrets` provide a mechanism to hold sensitive information such as passwords

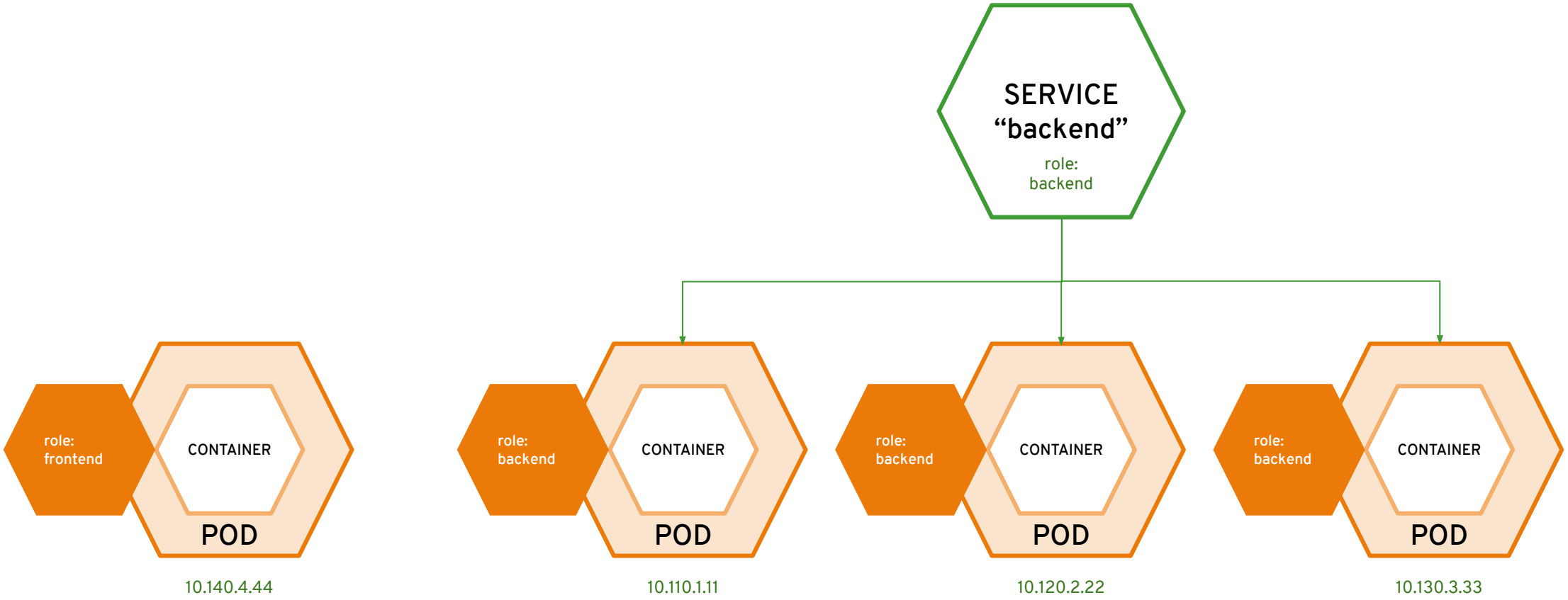


projects isolate apps across environments, teams, groups and departments



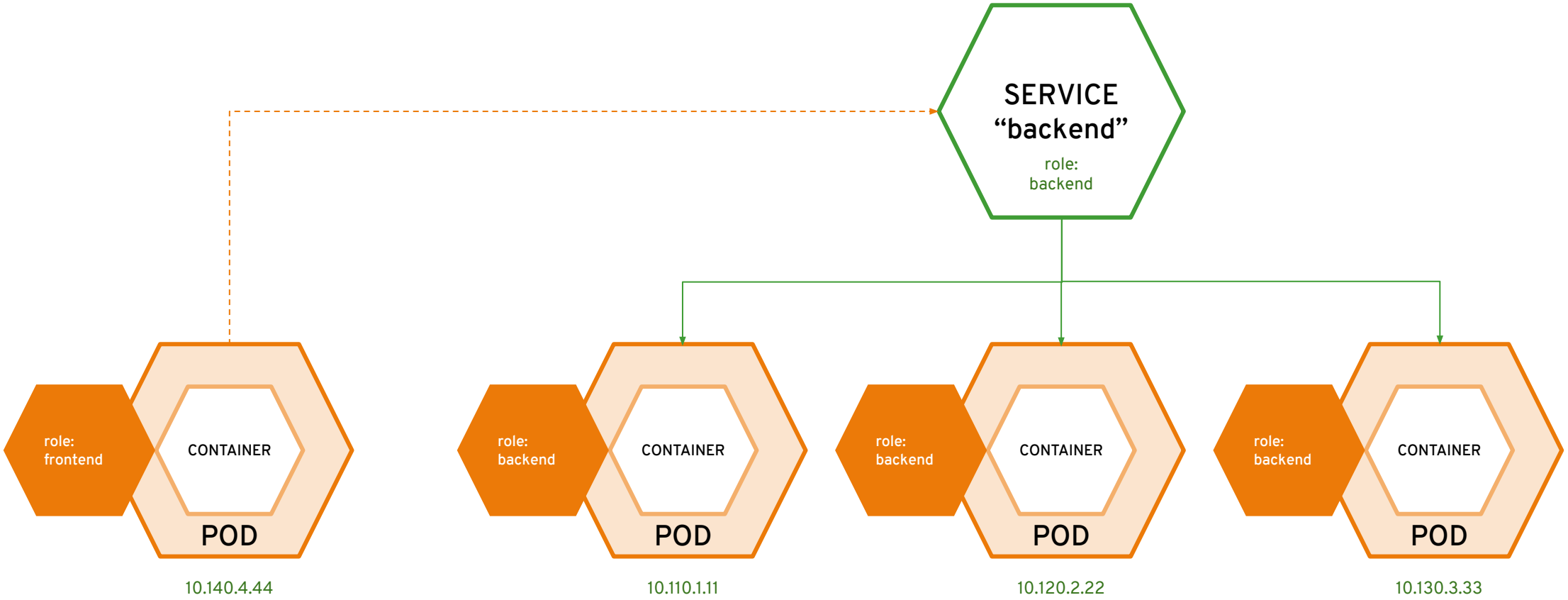
Red Hat

services provide internal load-balancing and service discovery across pods



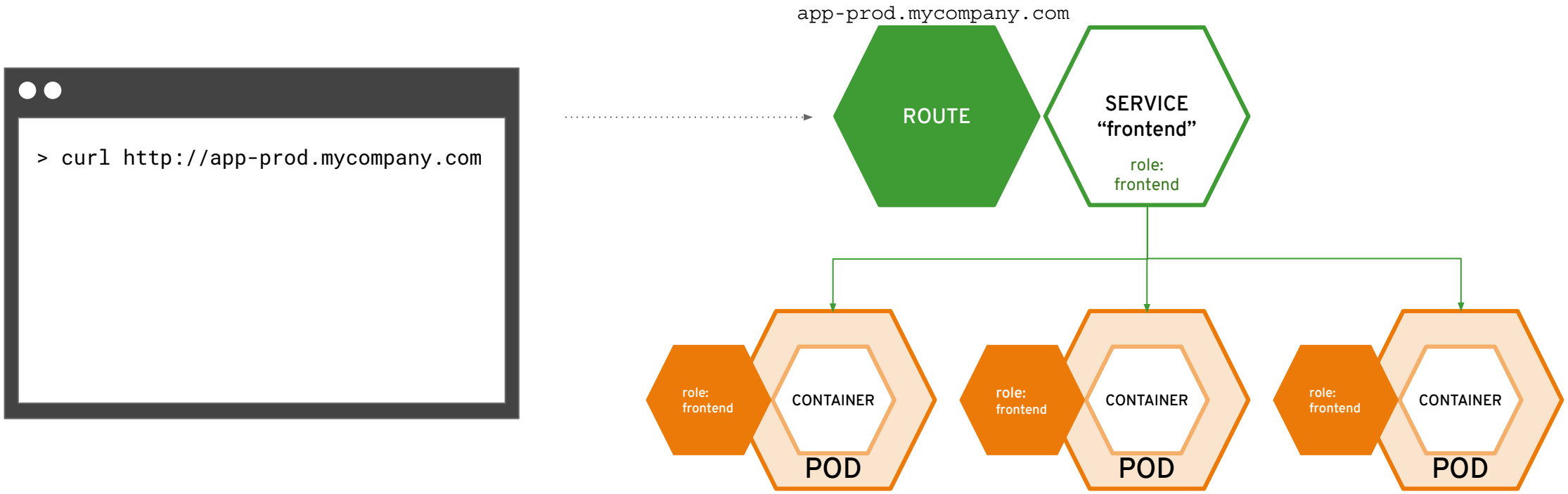
Red Hat

apps can talk to each other via services

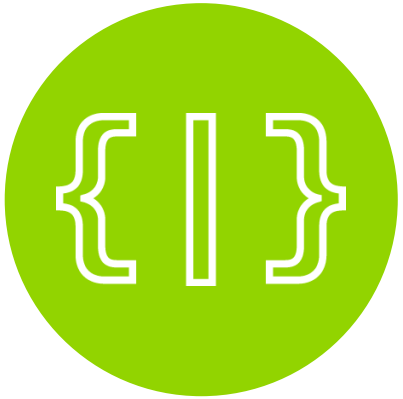


Red Hat

`routes` make services accessible to clients outside the environment via real-world urls



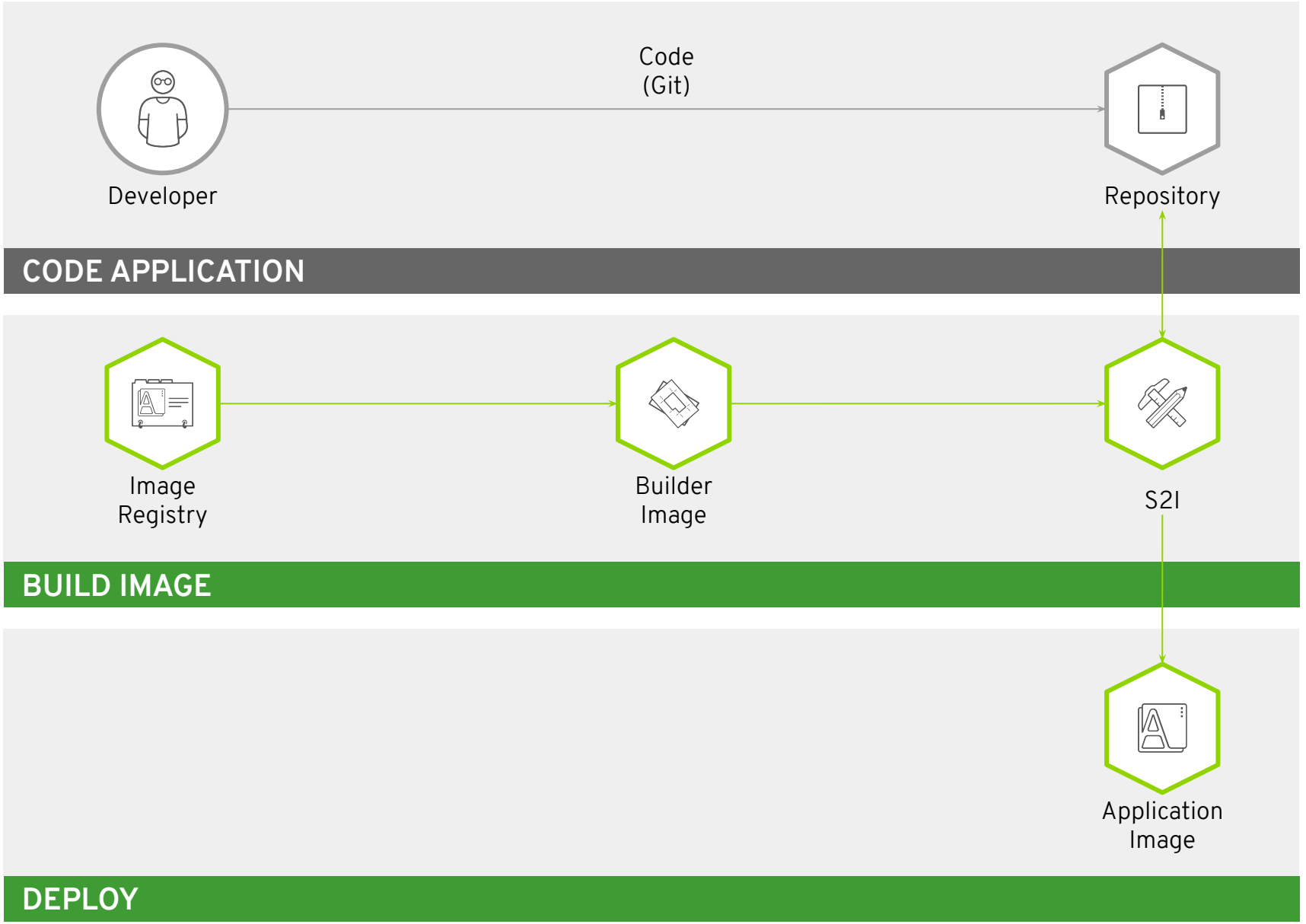
Red Hat



DEPLOY YOUR SOURCE CODE

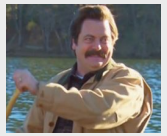
DEPLOY YOUR APP BINARY

DEPLOY YOUR CONTAINER IMAGE

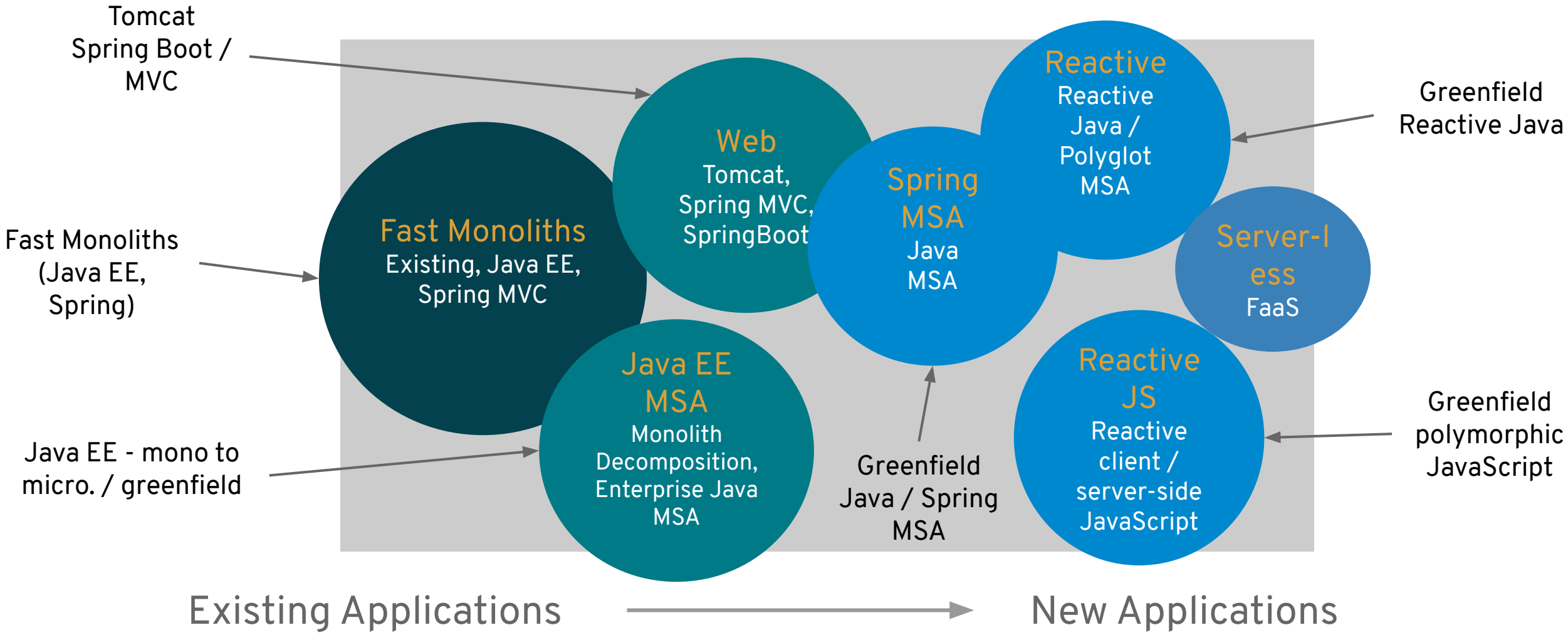

Code
(Git)
Developer
Repository
CODE APPLICATION
Image
Registry
Builder
Image
S2I
BUILD IMAGE
Application
Image
DEPLOY

Module 1: Optimizing Existing Applications

1. Optimizing Existing Applications
2. Debugging, Monitoring, and Continuous Delivery
3. Control Cloud Native Apps with Service Mesh
4. Advanced Cloud Native with Event-Driven Serverless



THE SPECTRUM OF ENTERPRISE APPS



OPTIONS FOR CONTAINERIZATION

Existing Application

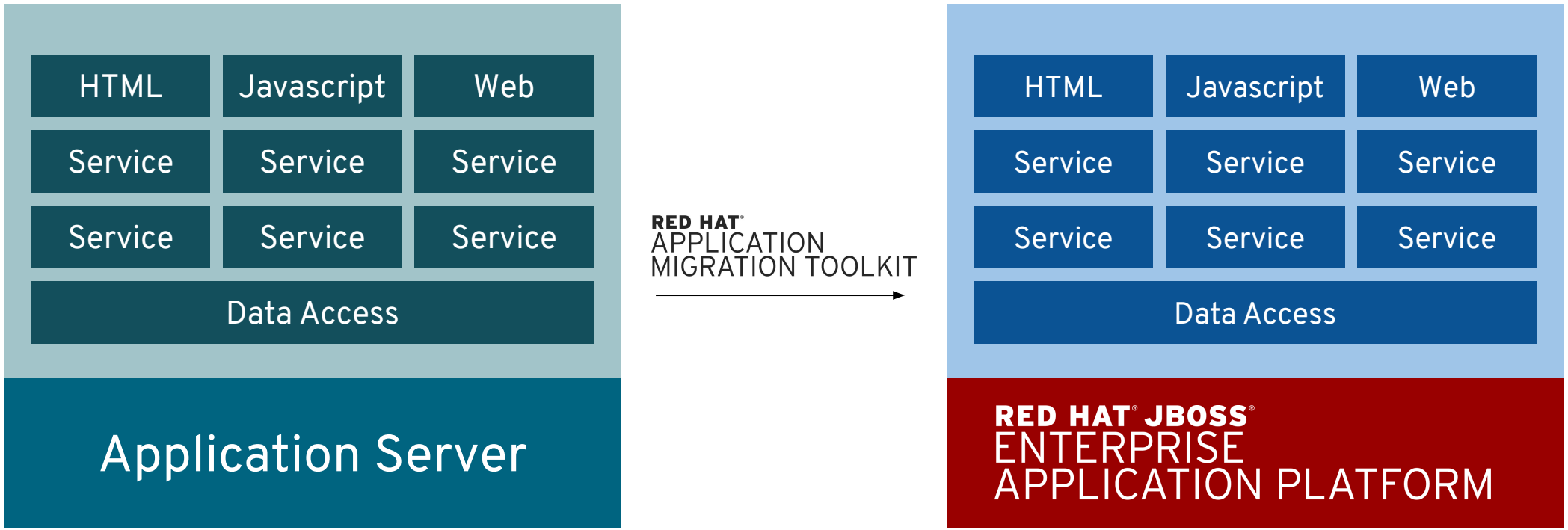
RETAIN	Keep and don't touch for now.
RETIRE	Decommission end-of-life application.
REHOST (Lift & Shift)	Repackage and move existing applications with as few changes as possible.
REPLATFORM (Lift & Reshape)	Change the underlying platform (runtime, framework, middleware, operating system).
REFACTOR (Extend / Rewrite)	Redesign code to take advantage of the new platform (extend, strangle, rewrite).
REPURCHASE (Drop & Shop)	Replace by Commercial off-the-shelf (COTS) or Software-as-as-Service (SaaS).

Low

Transformation Efforts / Complexity

High

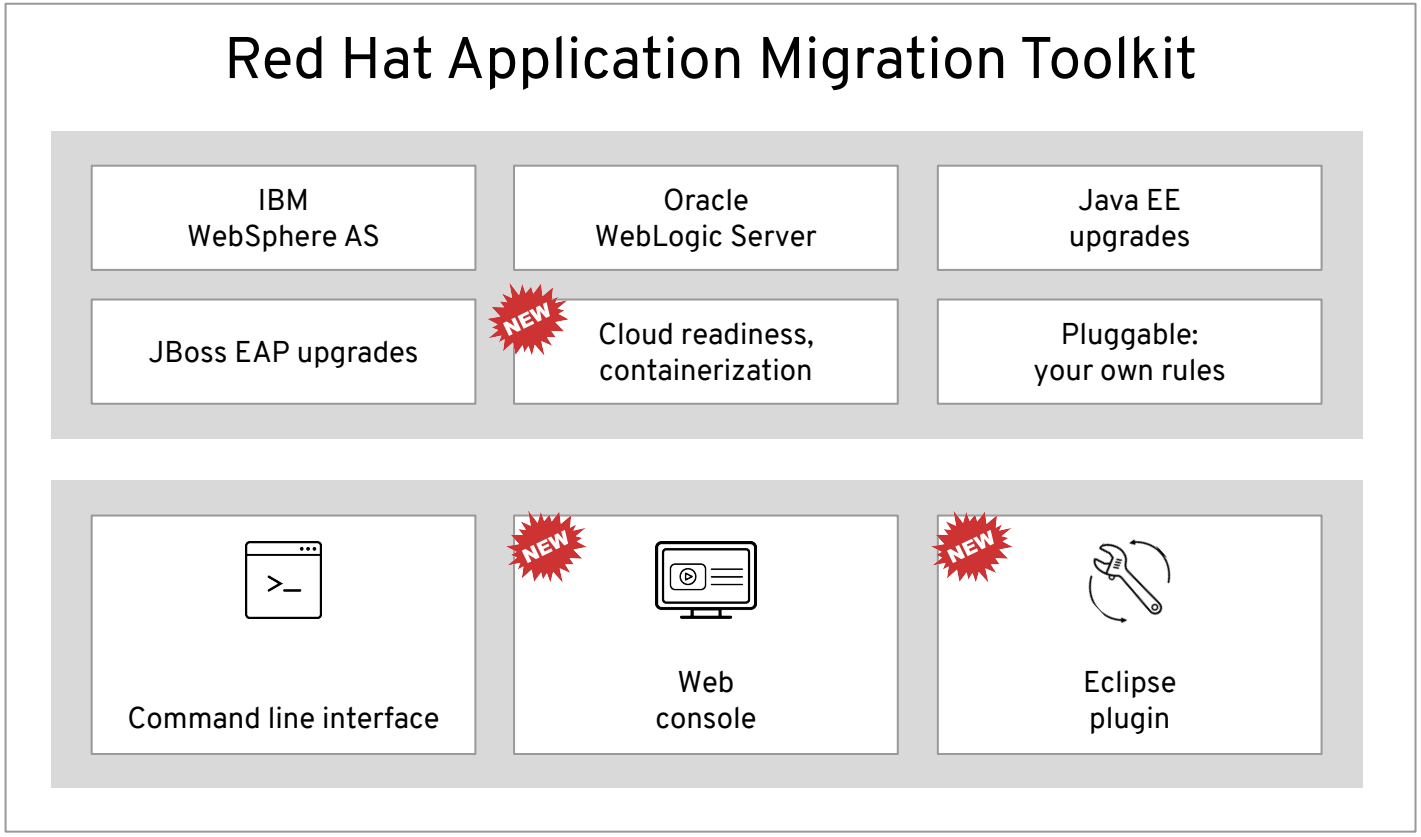
LIFT-AND-SHIFT MONOLITH TO CLOUD



RED HAT®
APPLICATION MIGRATION TOOLKIT

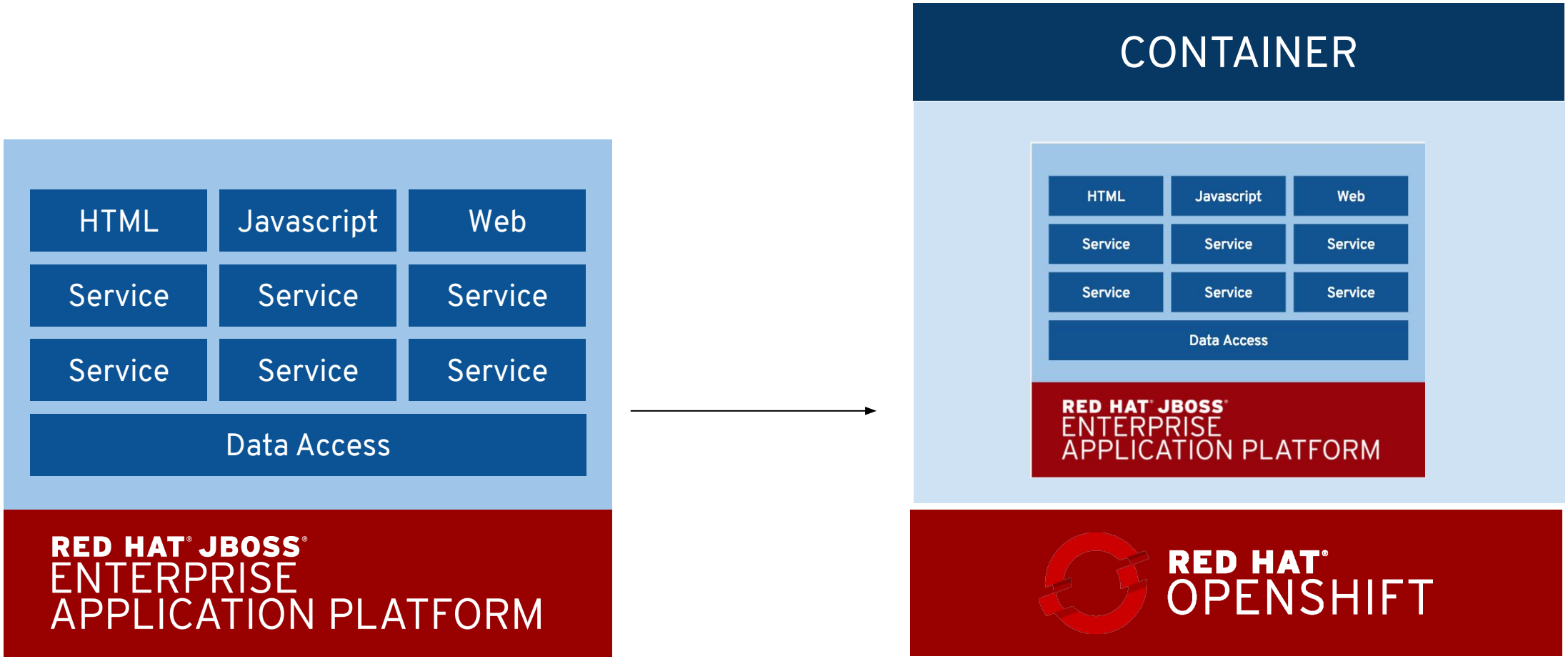
Catalyze large scale application modernizations and migrations

- Automate analysis
- Support effort estimation
- Accelerate code migration
- Free & Open Source



Red Hat

LIFT-AND-SHIFT MONOLITH TO CLOUD

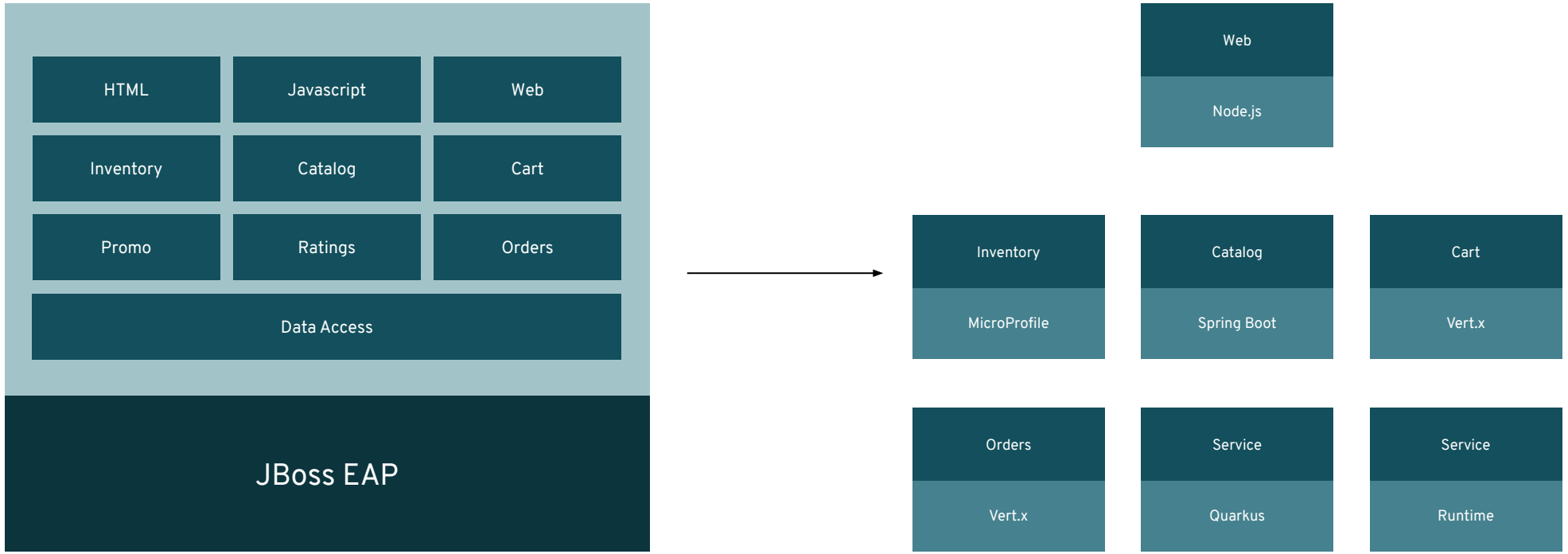


FAST-MOVING MONOLITHS

- Large organizations have a tremendous amount of resources **invested in existing** monolith applications
- Looking for a **sane way** to capture the benefits of containers and orchestration **without having to complete rewrite**
- **OpenShift** provides the platform for their existing investment with the benefit of a **path forward** for microservice based apps in the future

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
- As functionality is replaced, “dead” parts of monolith can be removed/retired.



CLOUD-NATIVE RUNTIMES

VERT.X

node JS

QUARKUS

“MODERN” JAVA APP STACK

MONOLITHS, MICROSERVICES
SINGLE APP
DAYS OF LIFE
100s MB RAM
SECONDS TO START

App

Dynamic Application Frameworks

Application Server

Java Virtual Machine (Hotspot)


QUARKUS

MICROSERVICES, **SERVERLESS**
SINGLE APP
SECONDS TO DAYS OF LIFE
10s MB RAM
MILLISECONDS TO START

App

Optimized Application Frameworks

Java Virtual Machine (Hotspot)

Optional


Red Hat

- Microservices for Developers using Spring Framework
  - Spring Boot (2.1.6), Spring Core, Spring Data, Spring Web, Spring Security, etc
- An opinionated approach to building Spring applications
- Red Hat Certified with
  - OpenShift Java Runtime
  - JBoss Web Server (Tomcat) embedded web container
- **Can also use Spring APIs in Quarkus**

GOAL FOR LAB

In this lab you will learn:

- How to use lab environment for today
- How to migrate an existing legacy Java EE application (CoolStore) from Weblogic to **JBoss EAP** using **Red Hat Application Migration Toolkit**
- How to deploy the result to **OpenShift container platform** to create a *Fast Moving Monolith*
- Begin modernization journey by breaking subset of monolith into microservices using **Spring Boot** and **Quarkus**

LAB INSTRUCTIONS

- **Everything is done in browser** - no local commands or installs needed on your laptop
- Tested with **Chrome 75.0.3770.142**, **Firefox 60.8.0esr**. → **Safari 12.x does not work!**
- If things get weird, just reload browser page
- **Turn off VPN** (we use websockets extensively), **pause AdBlock** for the lab domain (there are no ads)
- To recreate the lab locally, visit

  `github.com/RedHat-Middleware-Workshops/cloud-native-workshop-v2-infra`
- Everyone should have their own **unique logins,** e.g.: `user45` / `r3dh4t1!`

http://bit.ly/spectrum-dec2019-lab1

Credentials: `userXX` / `r3dh4t1!`

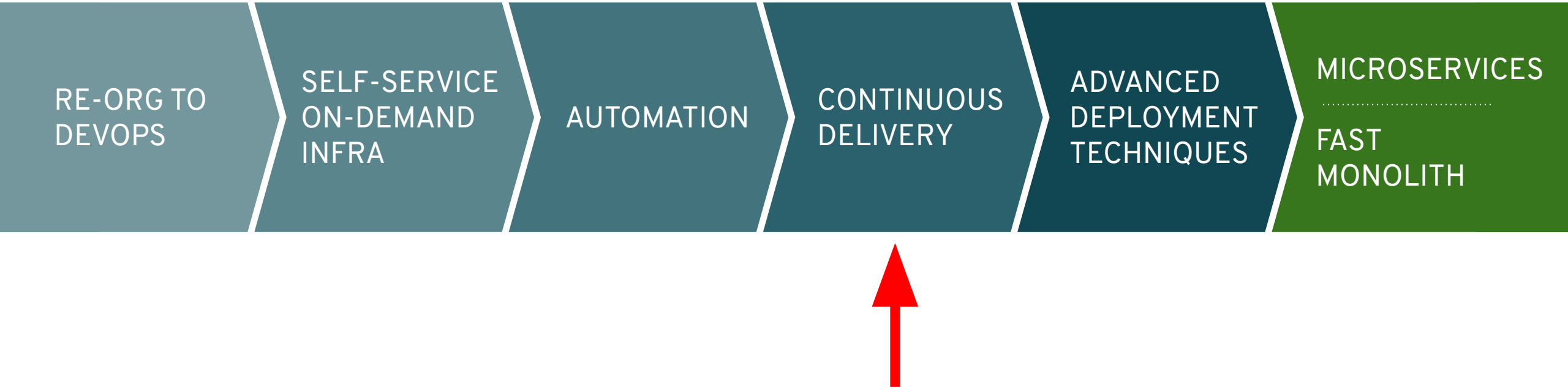
If you get stuck, raise a hand!

Module 2: Debugging, Monitoring and Continuous Delivery

1. Optimizing Existing Applications
2. Debugging, Monitoring, and Continuous Delivery
3. Control Cloud Native Apps with Service Mesh
4. Advanced Cloud Native with Event-Driven Serverless

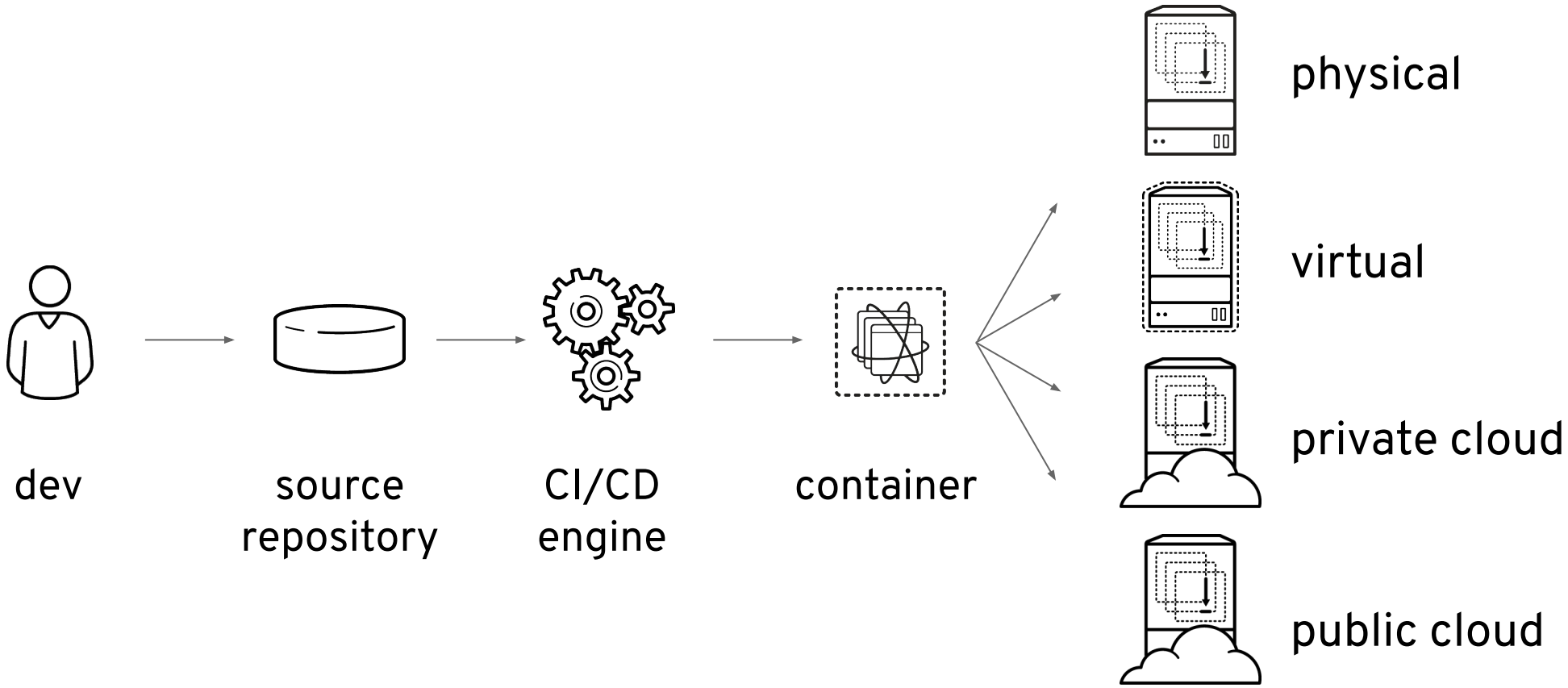
THE PATH TO CLOUD-NATIVE APPS

A DIGITAL DARWINISM



Red Hat

DEPLOYMENT PIPELINES

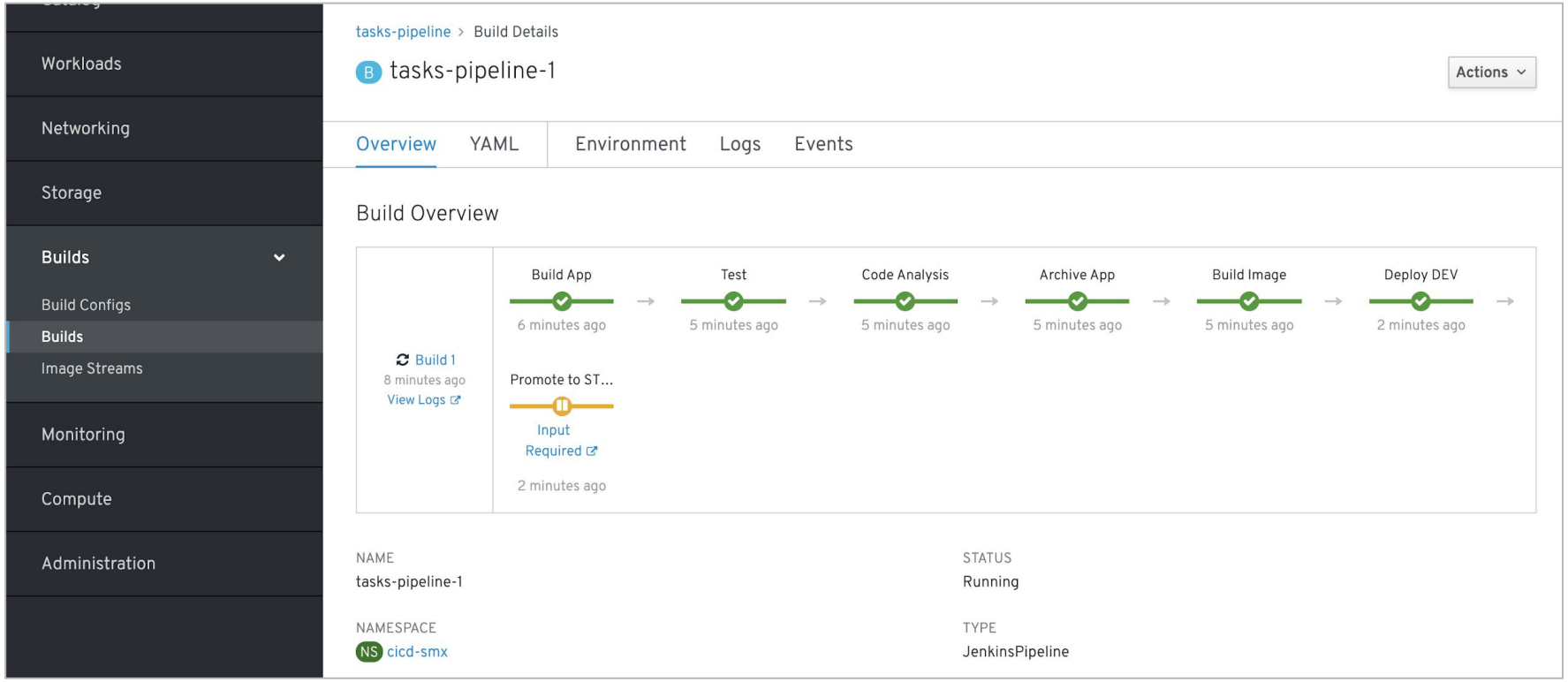


OPENSHIFT PIPELINES

Jenkins is still the most used CI/CD platform in enterprises and can be used from inside OpenShift.

An intuitive pipeline visualization makes it simple for users to see how builds are progressing.

The full Jenkins UI is also available.



Red Hat

OPENSHIFT PIPELINES

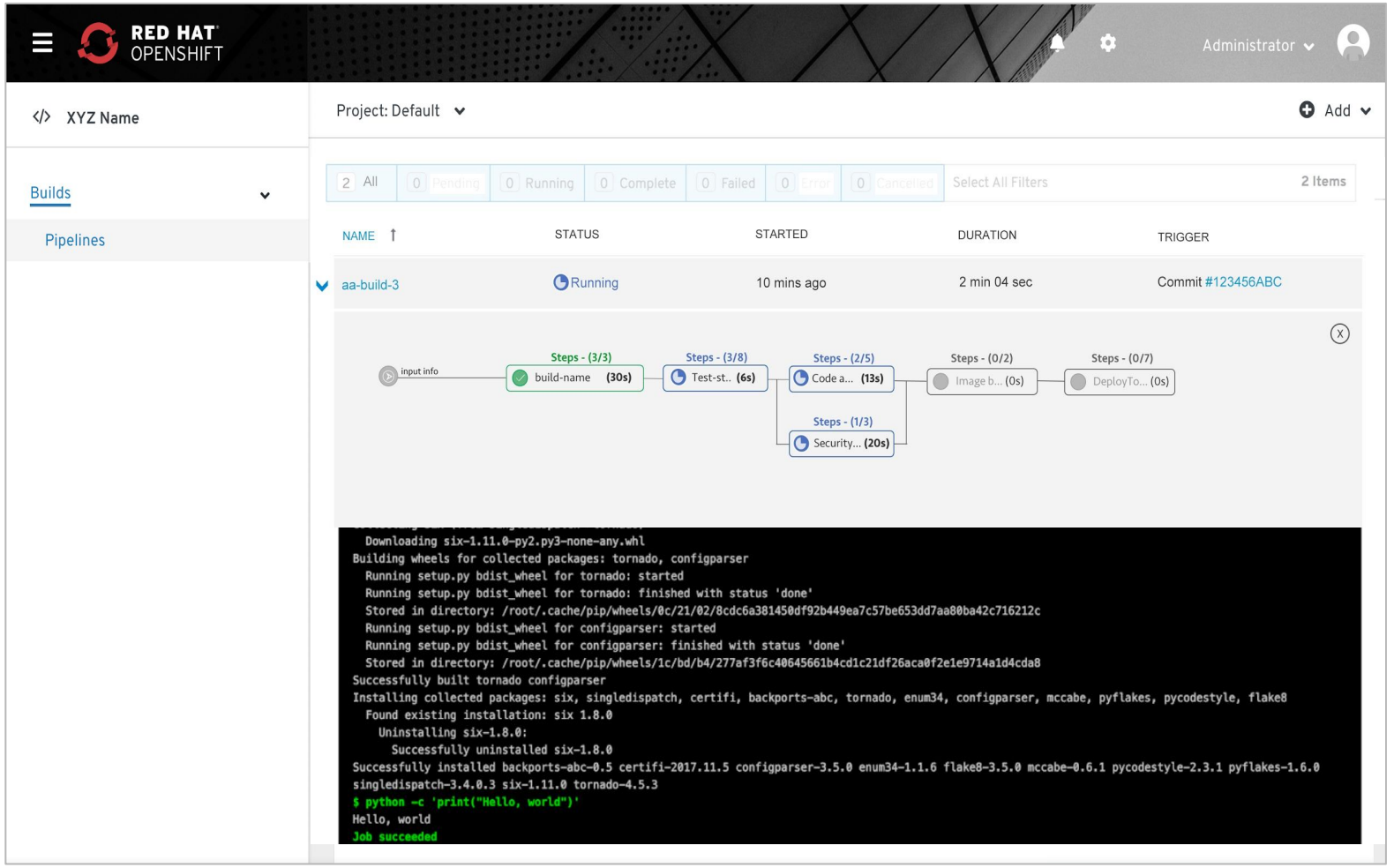
- CI/CD workflow via **Jenkins**
- Pipelines are started, monitored, and managed similar to other builds
- Auto-provisioning of Jenkins server
- On-demand Jenkins slaves
- Embedded Jenkinsfile or in Git repo

```
pipeline  {
  agent  {
    label  'maven'
  }
  stages  {
    stage('build  app')  {
      steps  {
        git  url:
'https://git/app.git'
        sh  "mvn  package"
      }
    }
    stage('build  image')  {
      steps  {
        script  {
          openshift.withCluster()  {
            openshift.startBuild("...")
          }
        }
      }
    }
  }
}
```

OPENSHIFT PIPELINES CI/CD PLATFORM

Next-gen Kubernetes CI/CD pipeline that works for containers (including serverless).

Based on the **Tekton** project (which was spun out of the Knative Pipelines project) started by Google, Red Hat and others.



Red Hat

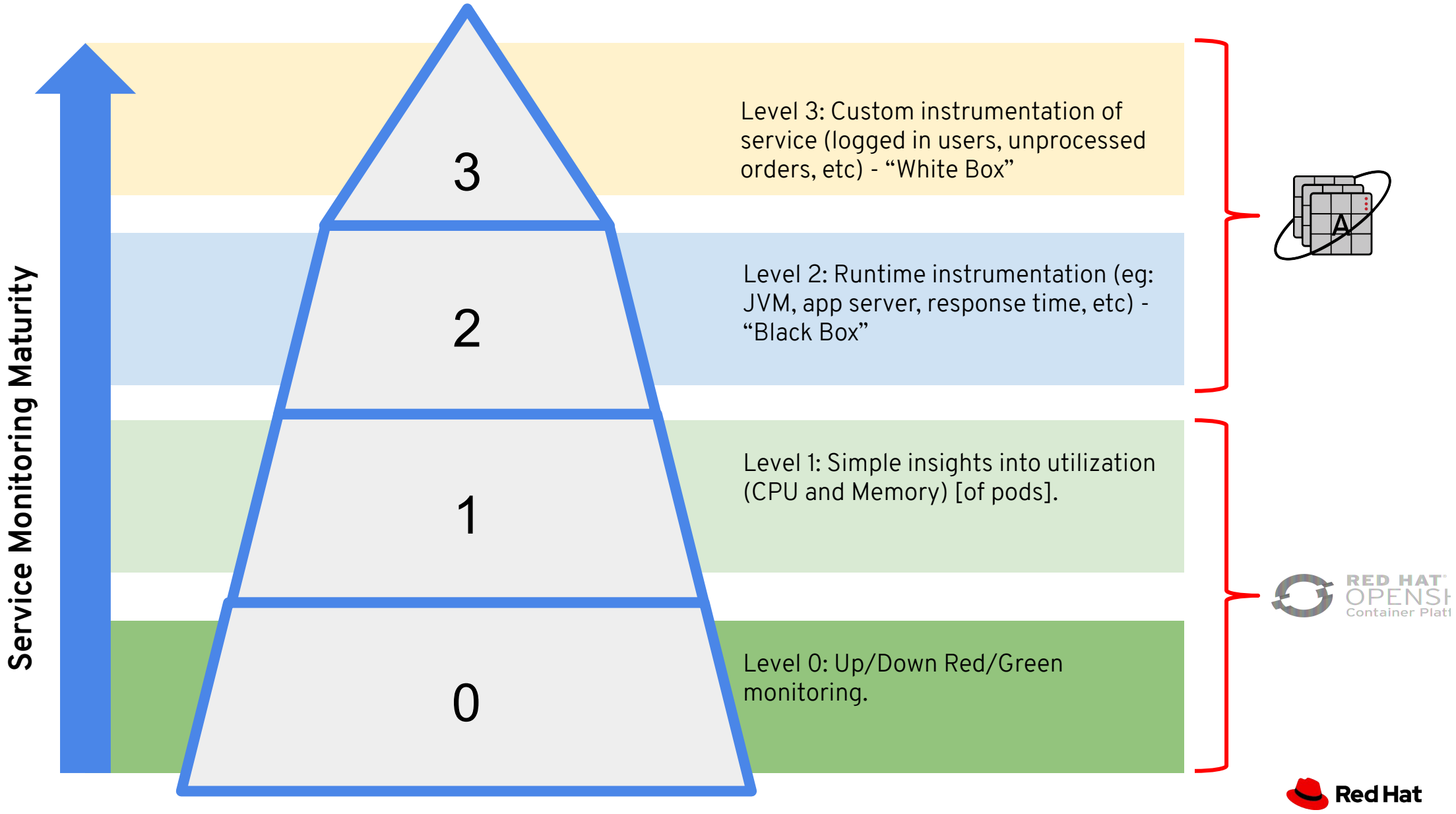
OpenShift Application Monitoring

Metrics collection and storage via Prometheus, an open-source monitoring system time series database.

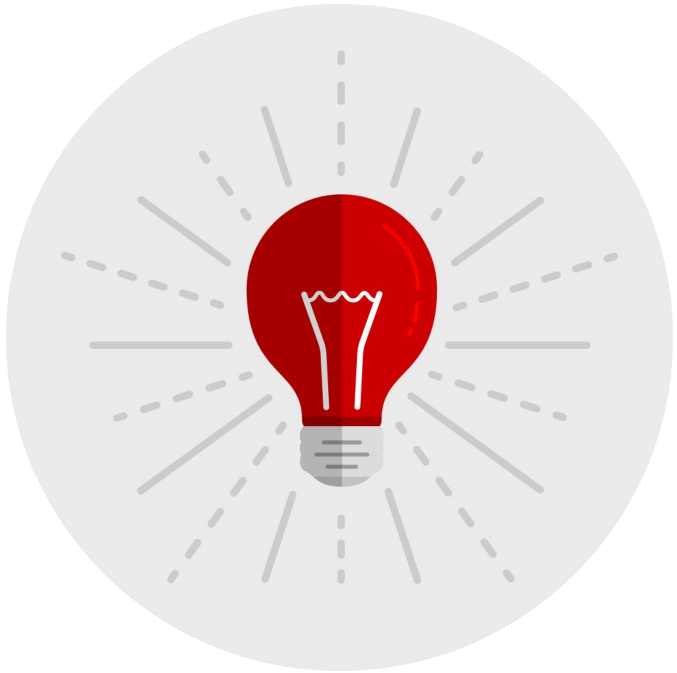
Alerting/notification via Prometheus' Alertmanager, an open-source tool that handles alerts sent by Prometheus.

Metrics visualization via Grafana, the leading metrics visualization technology.

Service Monitoring Maturity Model



Middleware OOTB Instrumentation



Prometheus Endpoint
Runtimes (Thorntail, Node.js, vert.x, Quarkus, Spring Boot), Decision & Process Servers.
Coming in EAP 7.3

Health Endpoint
EAP, Runtimes (Thorntail, Node.js, vert.x, Spring Boot, Quarkus), Decision & Process Servers.

Distributed Tracing Client
EAP, Runtimes (Thorntail, Node.js, vert.x, Spring Boot, Quarkus), Fuse.

JSON logging format
EAP, Spring Boot, Decision & Process Servers

GOAL FOR MODULE

In this module you will learn:

- How to create a CI/CD pipeline using Jenkins to automate the deployment of updates to microservice applications
- How to use CodeReady Workspaces to debug Java apps
- How to create health probes to allow the platform to act on unhealthy apps
- How to use distributed tracing and Jaeger to gain insight into application behavior
- How to instrument and monitor application performance using Prometheus (metrics ingestion and alerting) and Grafana (visualization)

LAB INSTRUCTIONS

- **Everything is done in browser** - no local commands or installs needed on your laptop
- Tested with **Chrome 75.0.3770.142**, **Firefox 60.8.0esr**. → **Safari 12.x does not work!**
- If things get weird, just reload browser page
- **Turn off VPN** (we use websockets extensively), **pause AdBlock** for the lab domain (there are no ads)
- To recreate the lab locally, visit

  `github.com/RedHat-Middleware-Workshops/cloud-native-workshop-v2-infra`
- Everyone should have their own **unique logins,** e.g.: `user45` / `r3dh4t1!`

Get Started at `[LINK]`

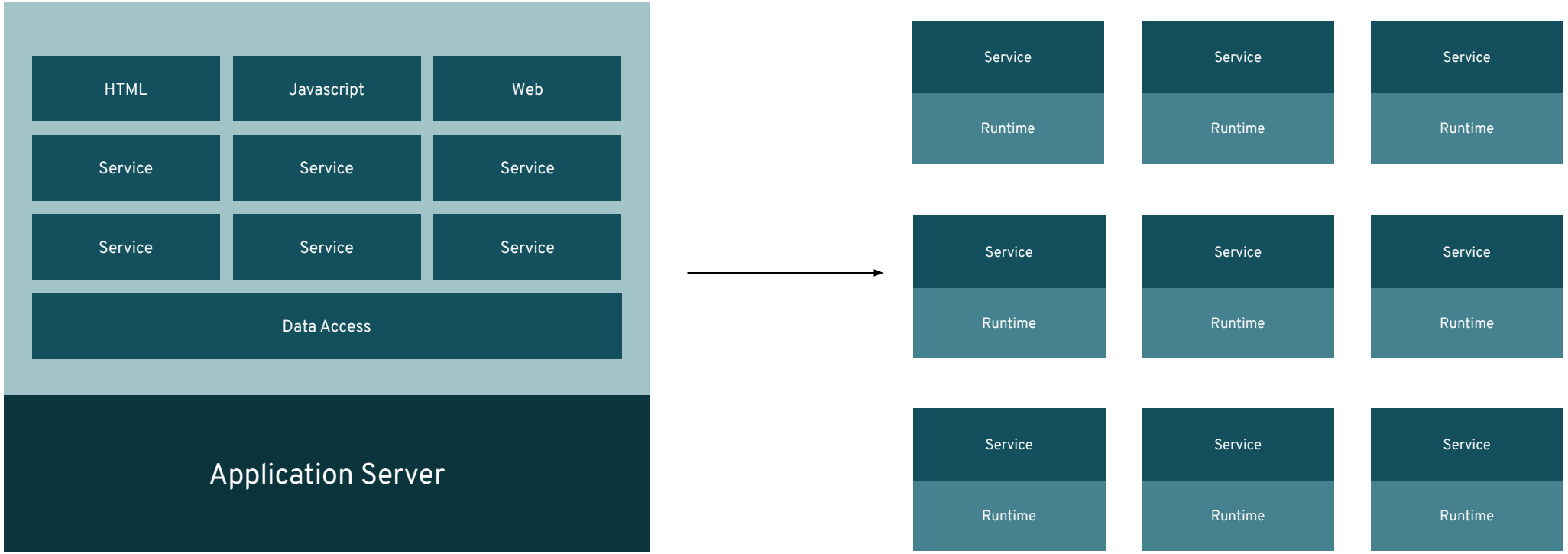
Credentials: `userXX` / `r3dh4t1!`

If you get stuck, raise hand

Module 3: Control Cloud Native Apps With Service Mesh

1. Optimizing Existing Applications
2. Debugging, Monitoring, and Continuous Delivery
3. Control Cloud Native Apps with Service Mesh
4. Advanced Cloud Native with Event-Driven Serverless

~~MICROSERVICES~~ ARCHITECTURE
DISTRIBUTED



DISTRIBUTED COMPUTING CHALLENGES

Fallacies of Distributed Computing

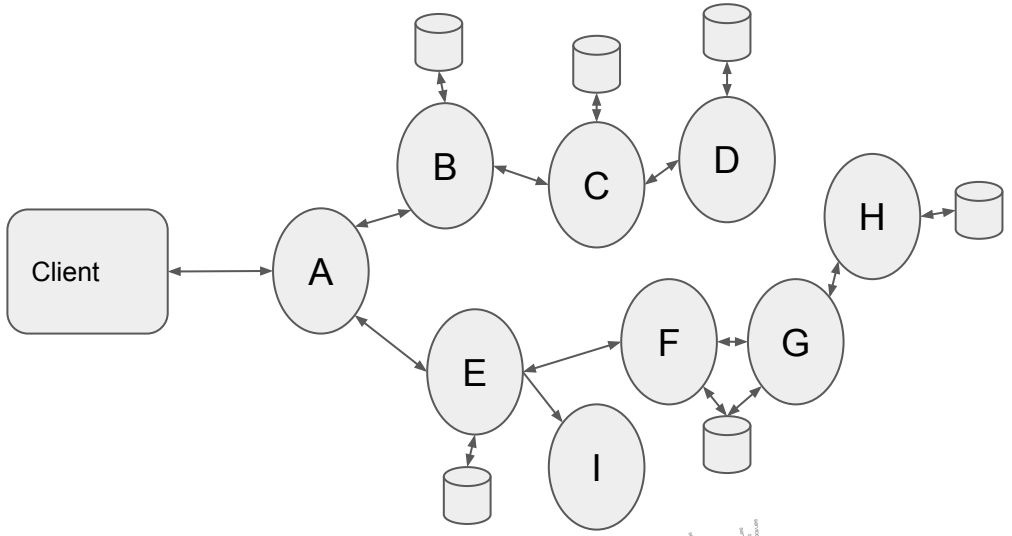
- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

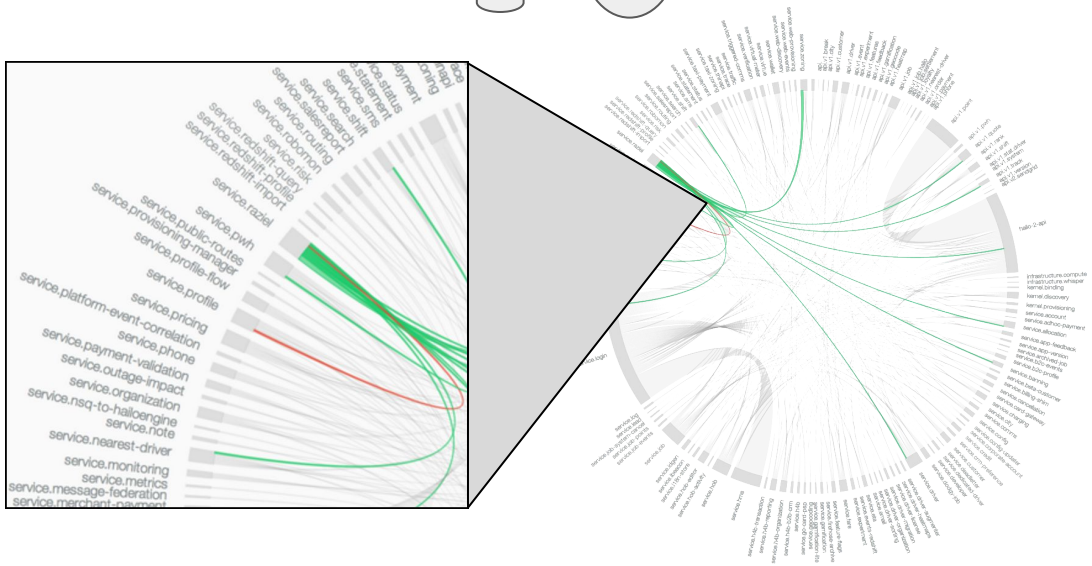
wikipedia.org/wiki/Fallacies_of_distributed_computing

MICROSERVICES ARE HARD

Because applications must deal with

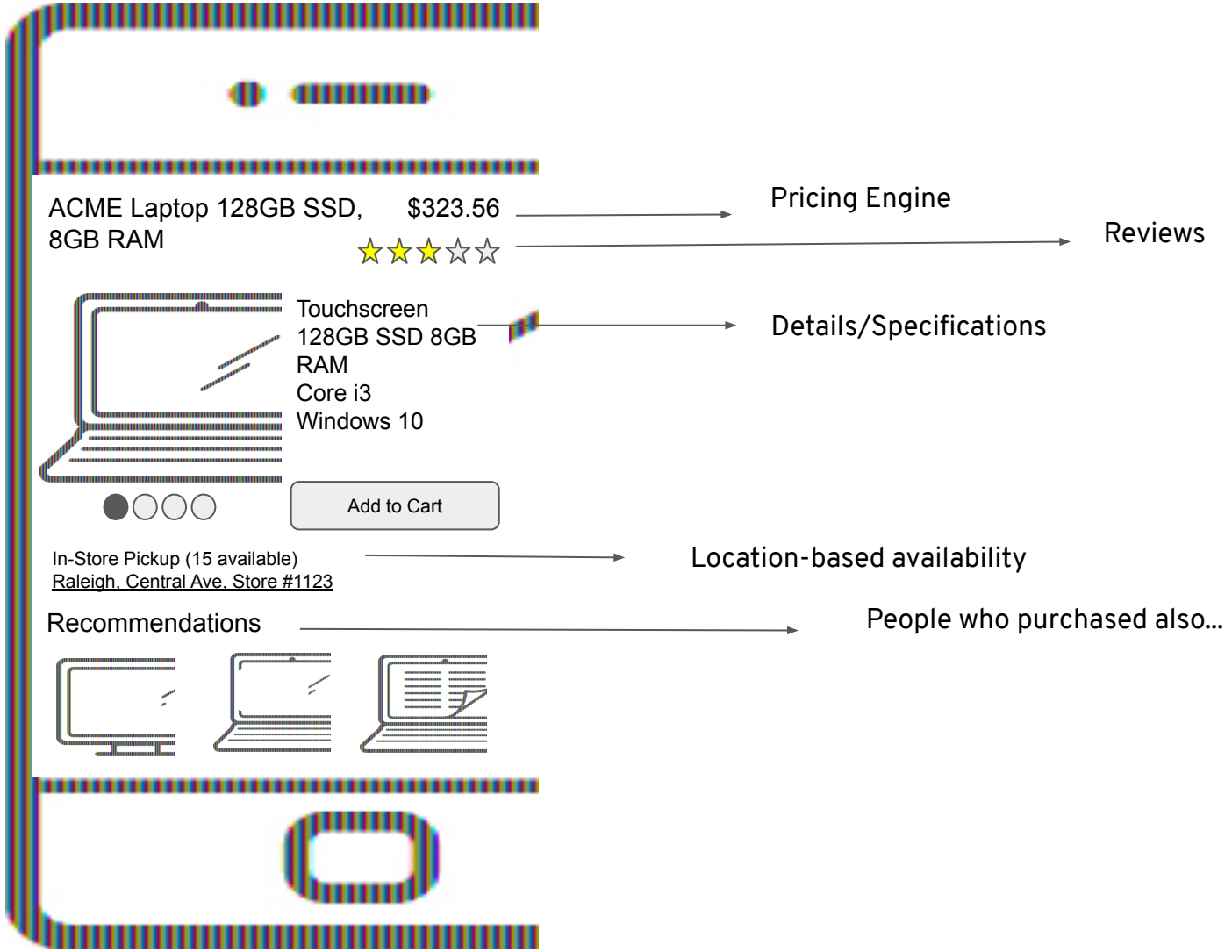
- Unpredictable failures
- End-to-end application correctness
- System degradation
- Topology changes
- Elastic/ephemeral/transient resources
- Distributed logs
- The fallacies of distributed computing



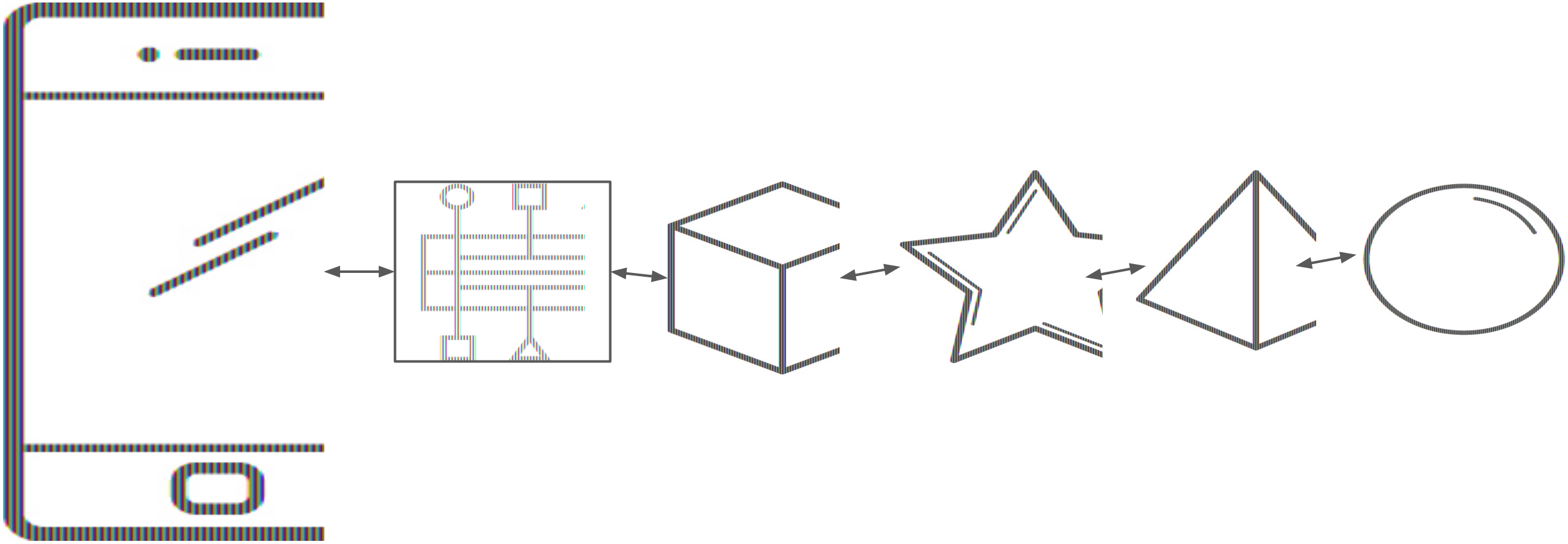


Red Hat

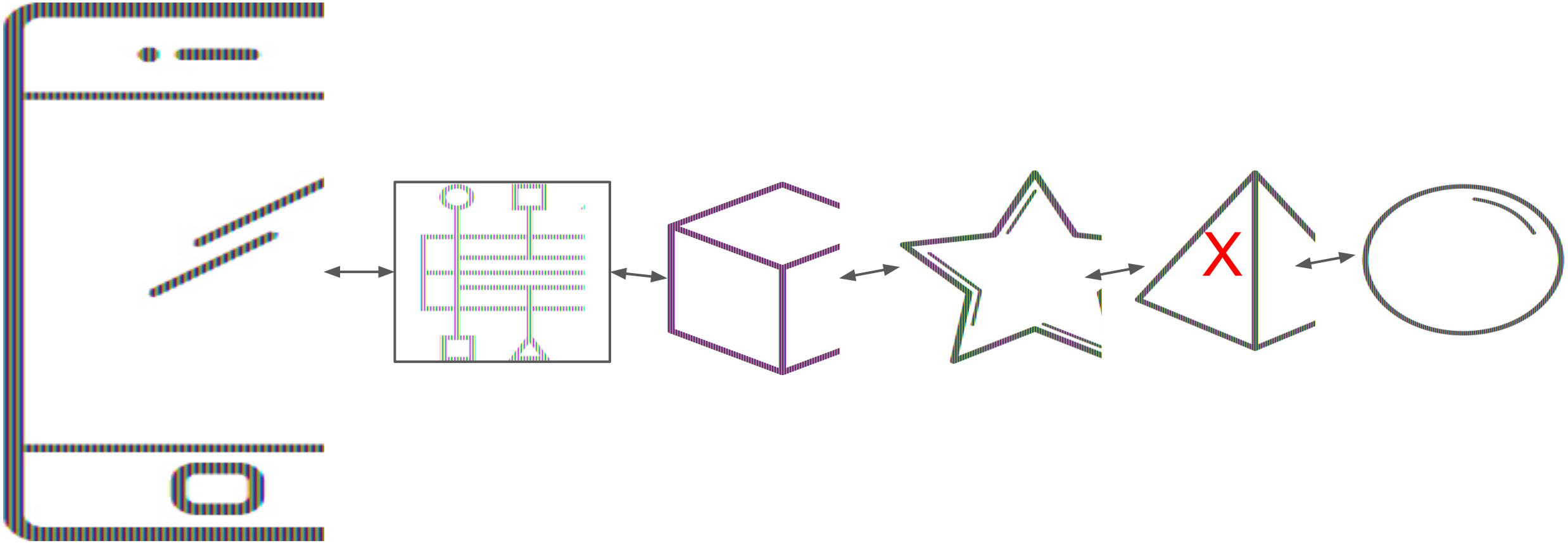
AN EXAMPLE


ACME Laptop 128GB SSD, 8GB RAM
$323.56
Pricing Engine
Reviews
Touchscreen
128GB SSD 8GB RAM
Core i3
Windows 10
Details/Specifications
Add to Cart
In-Store Pickup (15 available)
Raleigh, Central Ave, Store #1123
Location-based availability
Recommendations
People who purchased also...

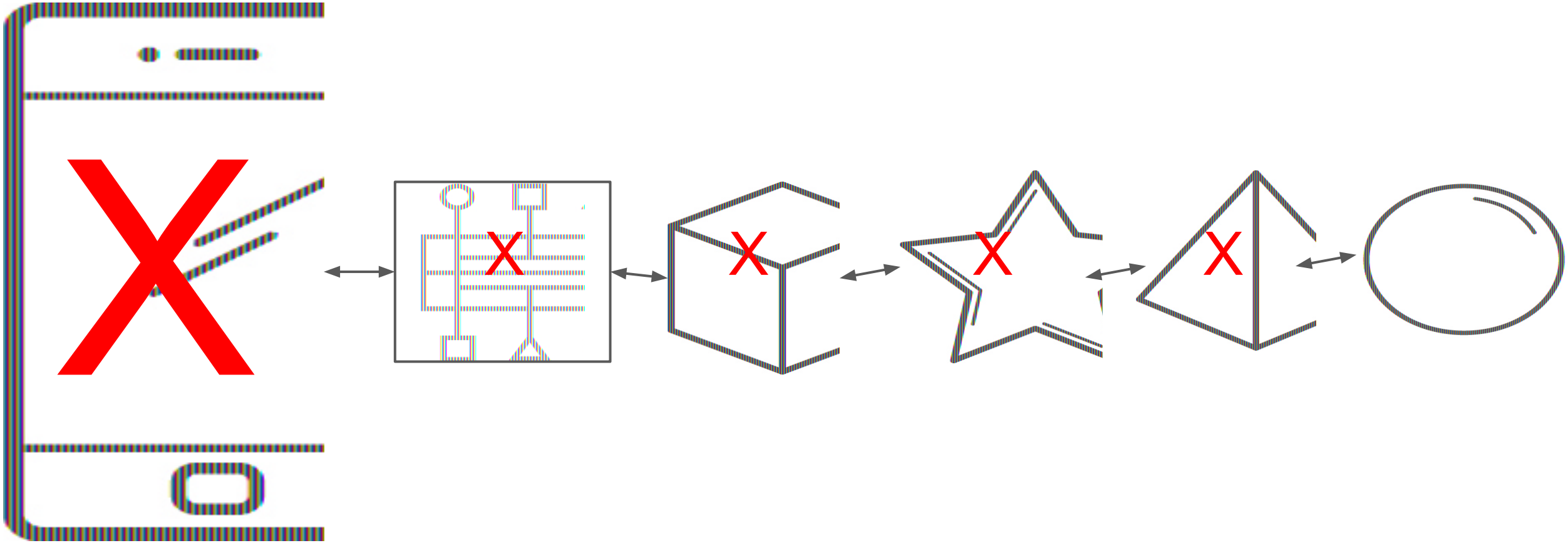
CHAINING



CHAINING (FAILURE)



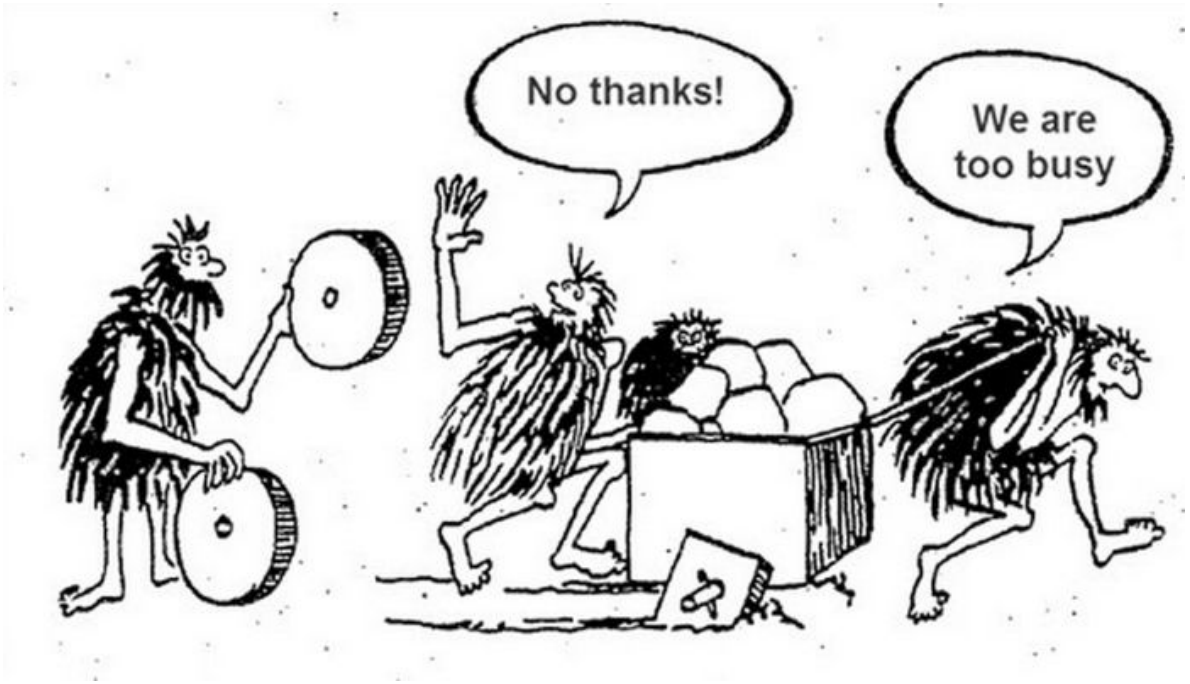
CHAINING (CASCADING FAILURE)

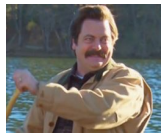


POSSIBLE SOLUTIONS

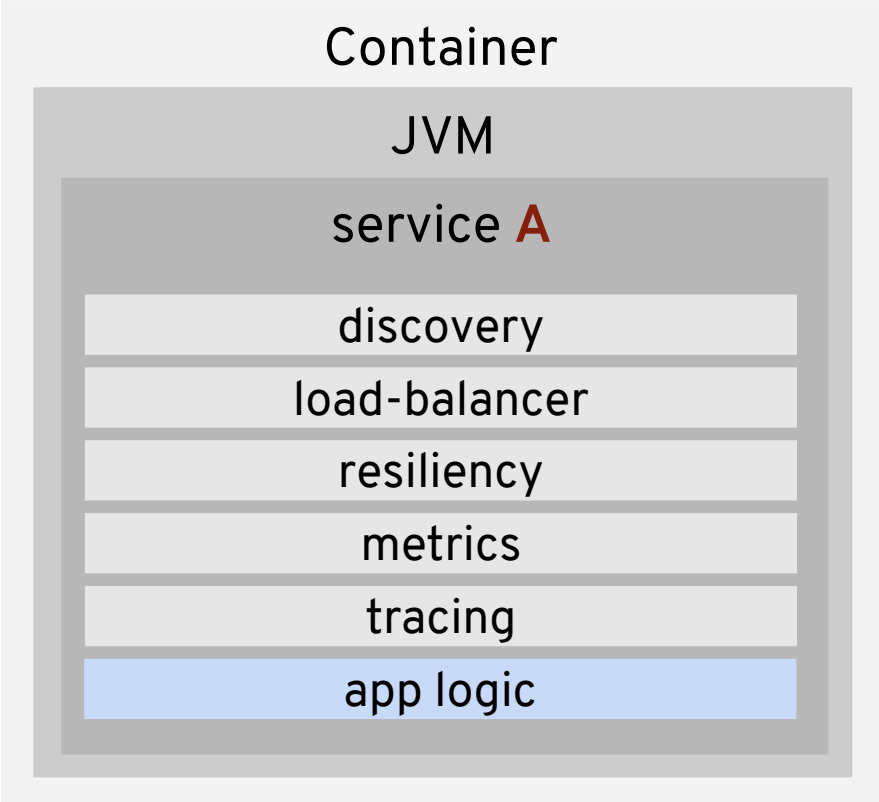
Have your developers do this:

- Circuit Breaking
- Bulkheading
- Timeouts/Retries
- Service Discovery
- Load Balancing
- Traffic Control





NETFLIX OSS

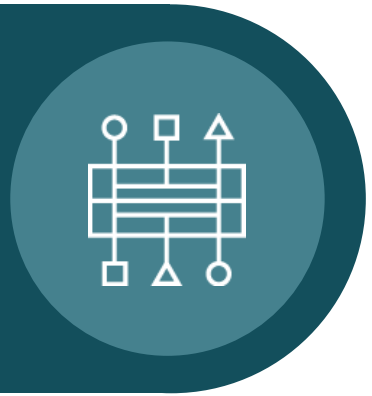


- Hystrix
- Zuul
- Ribbon
- Eureka
- Archaius

Need a library to support each language/framework combination

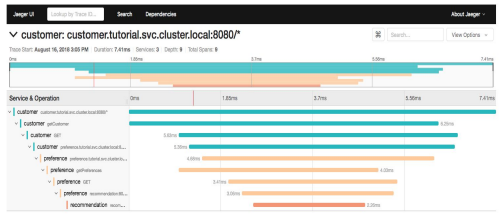
WHAT ABOUT…?

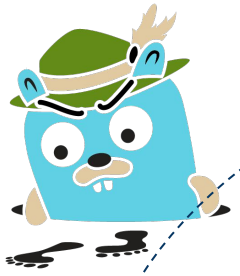
POLYGLOT APPS



EXISTING APPS

Service Mesh Ecosystem





Jaeger

Secure

Istio

Observe

Connect

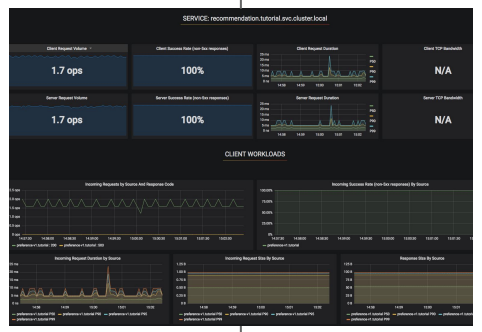
Control

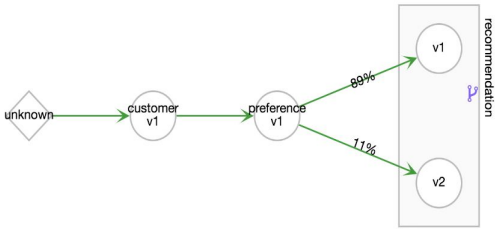
kiali

Kiali

Prometheus

Observe





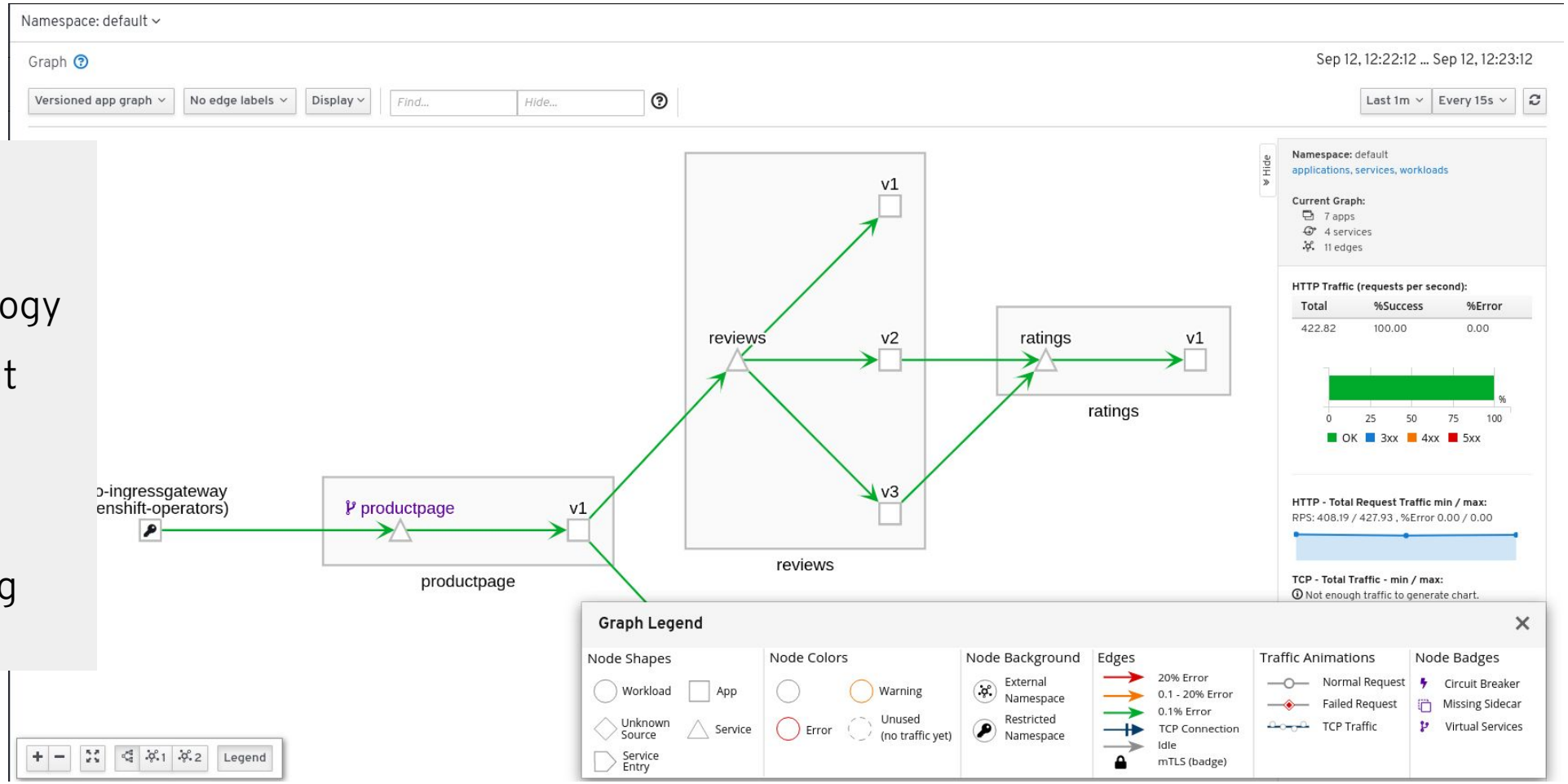
Grafana

Red Hat

Enhanced Visualization of Cluster Traffic With Kiali

Visualization of what Matters most:

- Application Topology
- Traffic throughput
- Error Rates
- Service Latency
- Service Versioning



GOAL FOR LAB

In this lab you will learn:

- How to deploy apps into the **OpenShift Service Mesh**
- How to generate and visualize **deep metrics** for apps with **Kiali** console
- How to **alter routing** dynamically
- How to **inject faults** for testing
- How to do **rate limiting**
- How the mesh implements **circuit breaking** and **distributed tracing**

LAB INSTRUCTIONS

- **Everything is done in browser** - no local commands or installs needed on your laptop
- Tested with **Chrome 75.0.3770.142**, **Firefox 60.8.0esr**. → **Safari 12.x does not work!**
- If things get weird, just reload browser page
- **Turn off VPN** (we use websockets extensively), **pause AdBlock** for the lab domain (there are no ads)
- To recreate the lab locally, visit

  `github.com/RedHat-Middleware-Workshops/cloud-native-workshop-v2-infra`
- Everyone should have their own **unique logins,** e.g.: `user45` / `r3dh4t1!`

http://bit.ly/spectrum-dec2019-lab3

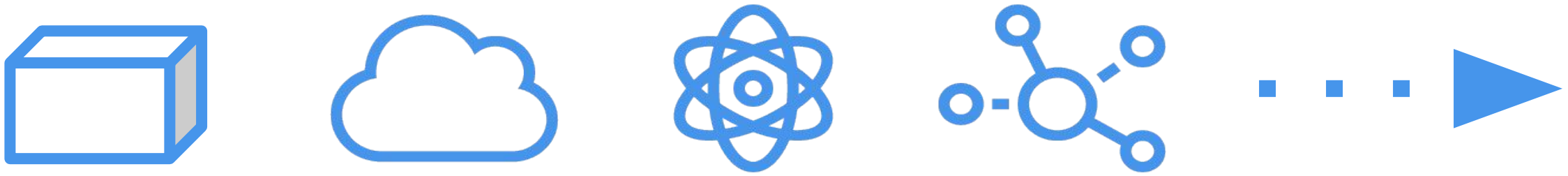
Credentials: `userXX` / `r3dh4t1!`

If you get stuck, raise a hand!

Module 4: Advanced Cloud Native Architectures With Event-driven Serverless

1. Optimizing Existing Applications
2. Debugging, Monitoring, and Continuous Delivery
3. Control Cloud Native Apps with Service Mesh
4. Advanced Cloud Native with Event-Driven Serverless

Growth in Application Architecture Choices



Monolith

Cloud Native

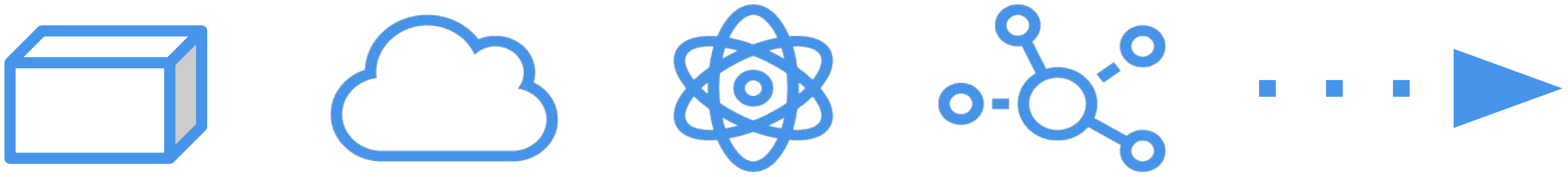
Microservices

Serverless

Event-Driven Architecture

Red Hat

Common Deployment Platform



Monolith | Cloud Native | Microservices | **Serverless** | **Event-Driven Architecture**

Red Hat

RISE IN EVENTS DRIVEN BY MODERN APP-DEV

1

MULTI-CLOUD

Deploying applications across on-premise and public cloud drives need to sync state and notify dependent applications anywhere

2

NEAR-REALTIME

End users expect a near realtime experience from modern applications

3

AVAILABILITY ISOLATION

Deployments must be resilient by being operationally isolated for availability

4

AGILITY

Applications must stay highly decoupled for agility, continuous improvement, and variation

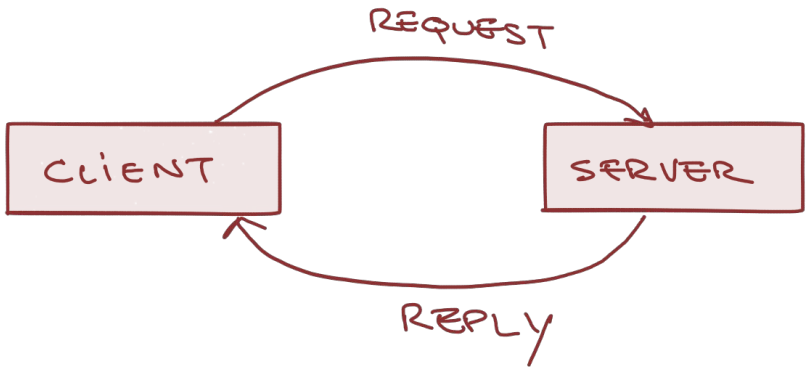
EVENTS

WHAT IS AN EVENT?

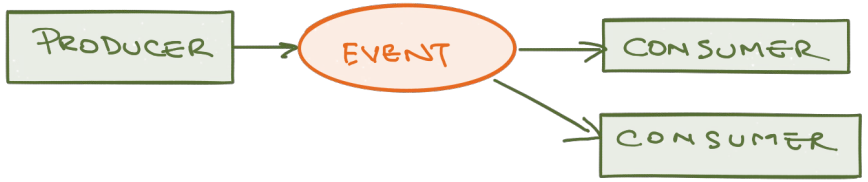
- Action or occurrence, something that happened in the past
  - ‘Order created’, ‘user logged in’
- Event characteristics:
  - Immutable
  - Optionally persistent
  - Shareable
- Event types: [1]
  - Notification
  - State Transfer (Command)
  - Event-Sourcing/CQRS

[1] https://martinfowler.com/articles/201701-event-driven.html

Request-reply vs. event-driven

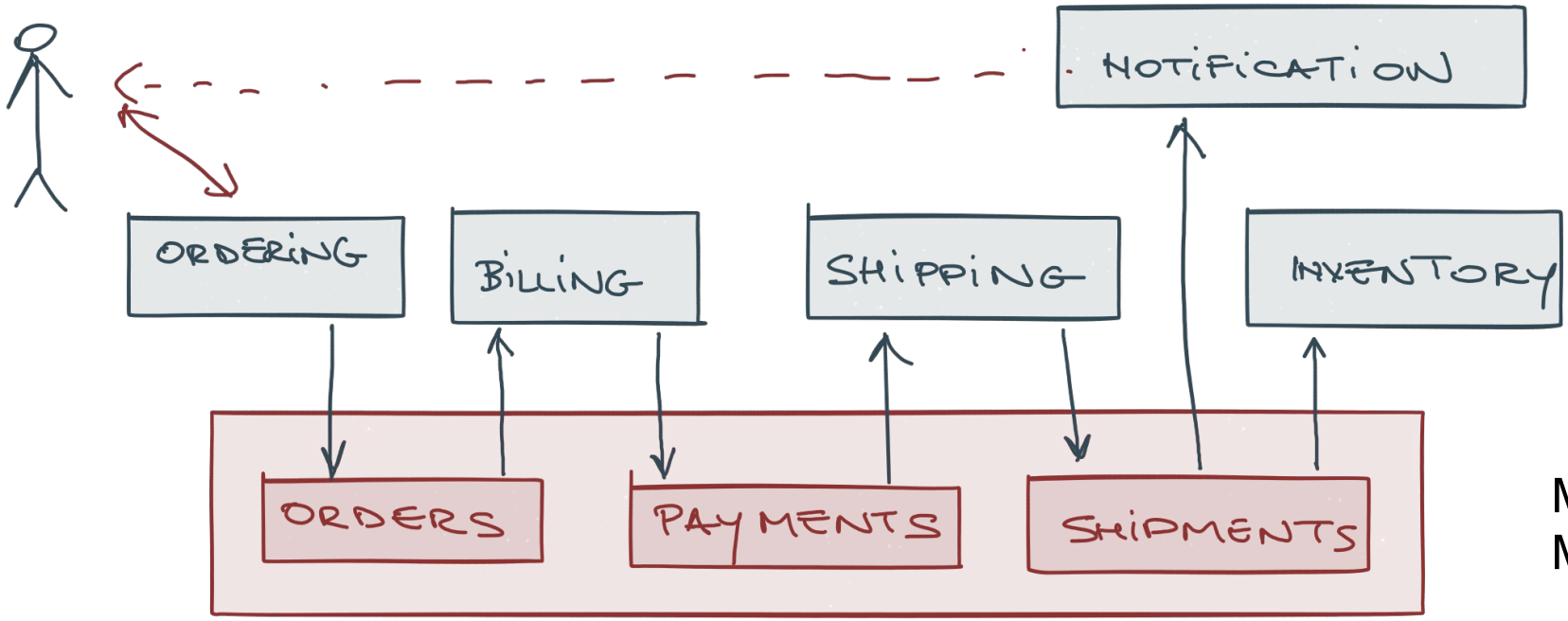


Synchronous & ephemeral
Low composability
Simplified model
Low tolerance to failure
Best practices evolved as REST



Asynchronous and persistent Decoupled
Highly composable
Complex model
High tolerance to failure
Best practices are still evolving

Event-driven microservices



Applications

Messaging Middleware

TRADITIONAL MESSAGING

EVENT STREAMING

Advantages

- Store-and-forward
- individual message exchanges (transactionality, acknowledgment, error handling/DLQs), P2P/competing consumer support
- Publish-subscribe support with limitations

Trade-offs

- No replay support
- Requires fast and/or highly available storage infrastructure
- No ordering at scale

Advantages

- long-term persistence, replay, semantic partitioning, large publisher/subscriber imbalances, replay and late-coming subscribers
- Shared nothing data storage model
- Repeatable ordering at scale

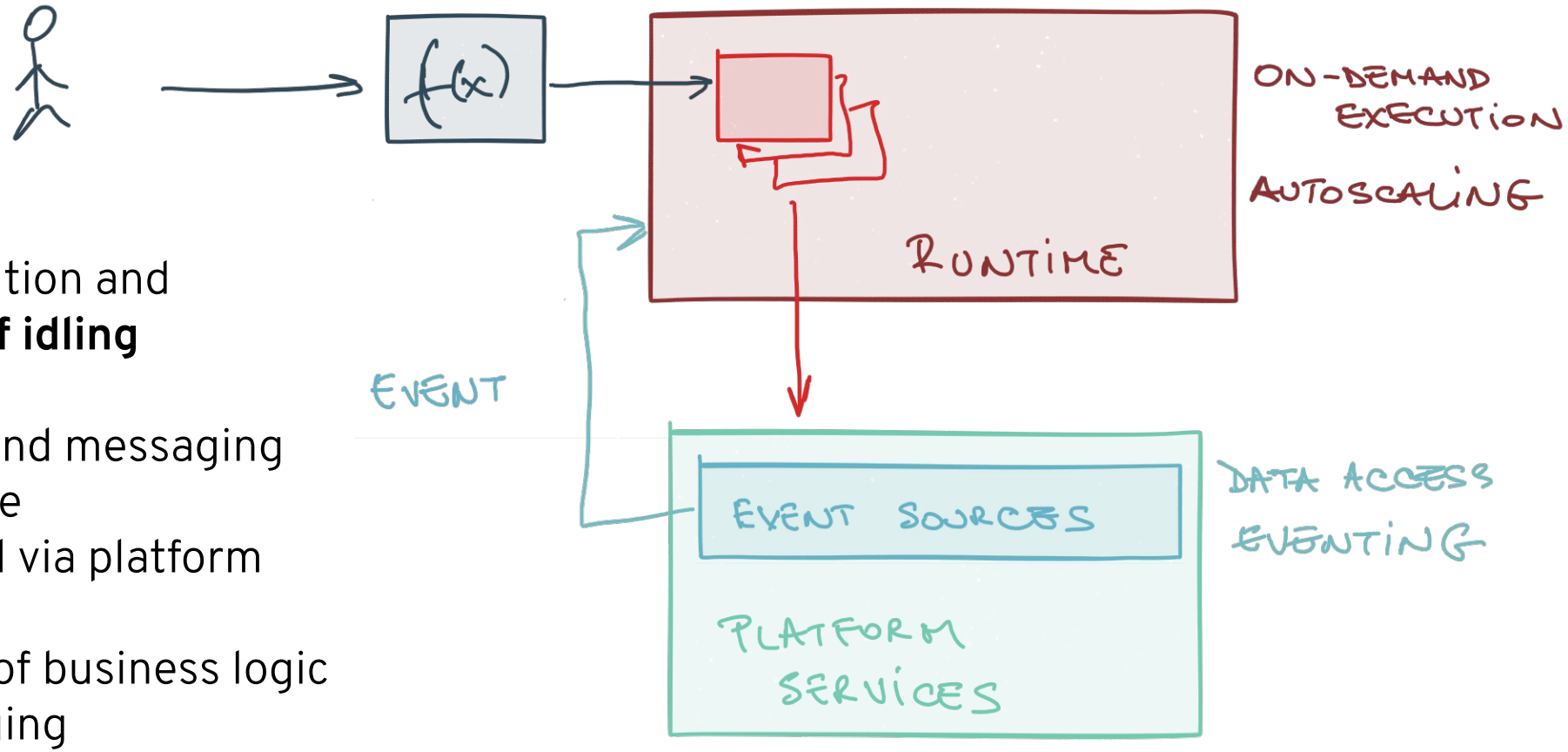
Trade-offs

- Weak support for individual message acknowledgment, p2p/competing consumers
- Larger data footprint and extremely fast storage access

Some challenges with microservices ...

- Utilization
  - Idling under low traffic
  - High resource consumption - memory, disk
- Connectivity
  - Must know broker location
  - Integration with event sources
- Abstraction
  - Must know broker type
  - Dependence on data/payload formats
- Observability
- Security

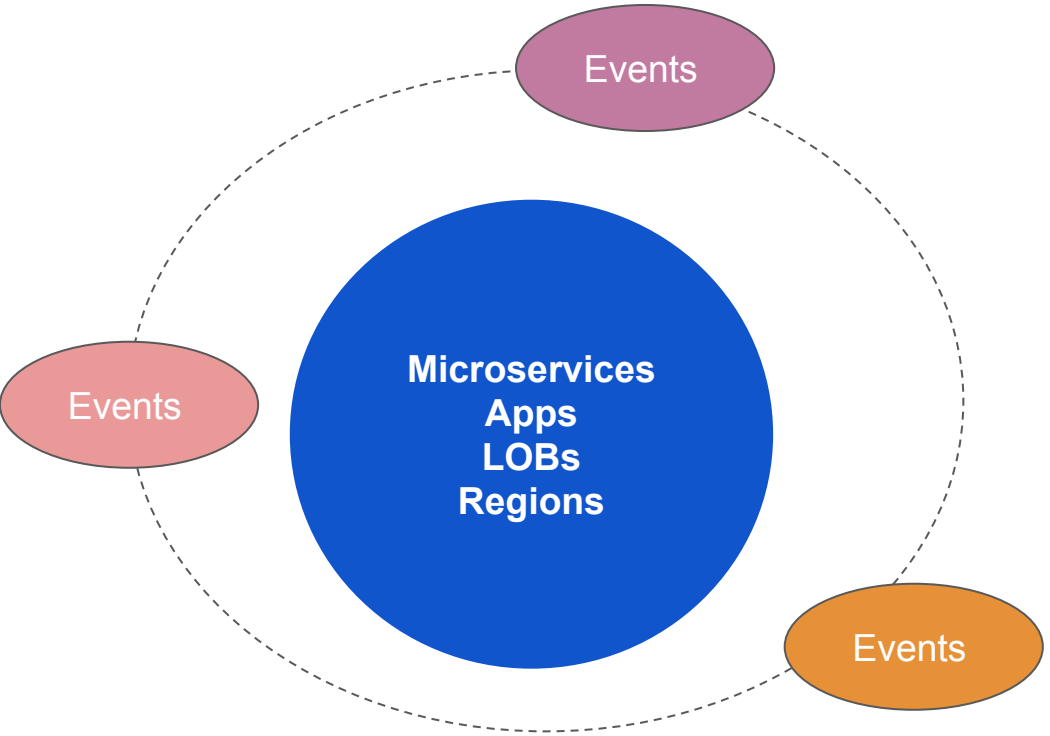
Serverless model: event-based and elastic



- Elastic execution and **avoidance of idling** (autoscale)
- Utility data and messaging infrastructure
  - Provided via platform services
- **Decoupling** of business logic from messaging infrastructure

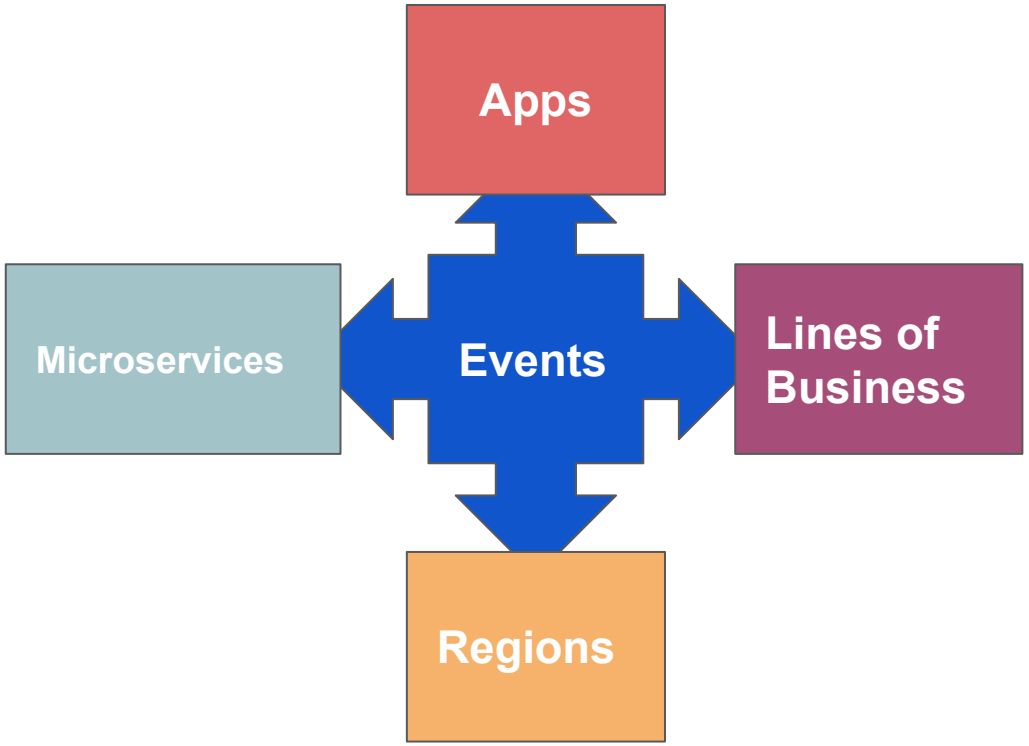
Red Hat

RETHINKING EVENT-DRIVEN ARCHITECTURE



System and data-centric

Events are designed to **respond to ad-hoc** connectivity needs

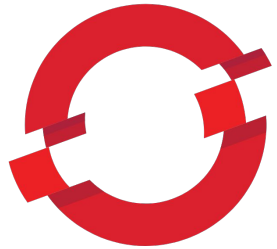


Event-centric

Events are first class citizens that **describe the interactions in the enterprise**

Red Hat

Your technology radar



OPENSHIFT

- Service Mesh (e.g. Istio):
  - Provide microservice interconnectivity
- Serverless platforms (e.g. Knative)
  - Container build and on-demand scheduling
- Container-native frameworks (e.g. Quarkus)
  - Optimize Java workloads for containerized apps
- Strimzi - Kafka operator for Kubernetes/OpenShift
- EnMasse - Messaging-as-a-Service for Kubernetes/OpenShift
- FaaS frameworks (e.g. Camel-K)
  - Schedule integration code directly on platform or via Knative

Istio

RED HAT AND REACTIVE

Red Hat AMQ

MESSAGING BACKBONE

Red Hat Data Grid

DISTRIBUTE, REACT ON DATA

Red Hat Application Runtimes

REACTIVE / FAAS FRAMEWORKS

Red Hat Middleware Portfolio

RULES EVALUATION
COMPLEX EVENTS
AUTOMATION

Red Hat Integration

CAMEL K
REACTIVE INTEGRATION
SERVERLESS

Red Hat CodeReady

REACTIVE DEVELOPER TOOLING

Red Hat OpenShift

APP ENVIRONMENT
INFRASTRUCTURE
SERVERLESS / KNATIVE
OPERATOR HUB

GOAL FOR MODULE

In this module you will learn:

- How to adopt event-driven programming and architectures using capabilities of a Red Hat solution
- Adding high performance distributed in-memory cache using Red Hat Data Grid
- Introduce Apache Kafka for responding and processing of application events
- Use Knative Serving and Eventing to begin to move applications to a serverless, event-driven architecture to more efficiently process app events

LAB INSTRUCTIONS

- **Everything is done in browser** - no local commands or installs needed on your laptop
- Tested with **Chrome 75.0.3770.142**, **Firefox 60.8.0esr**. → **Safari 12.x does not work!**
- If things get weird, just reload browser page
- **Turn off VPN** (we use websockets extensively), **pause AdBlock** for the lab domain (there are no ads)
- To recreate the lab locally, visit

  `github.com/RedHat-Middleware-Workshops/cloud-native-workshop-v2-infra`
- Everyone should have their own **unique logins,** e.g.: `user45` / `r3dh4t1!`

Get Started at `[LINK]`

Credentials: `userXX` / `r3dh4t1!`

If you get stuck, raise hand

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat