CLOUD-NATIVE ROADSHOW

DEVELOPER TRACK

redhat.

Hands-on-Labs:

dev.roadshow.openshift.com

LAB GUIDE

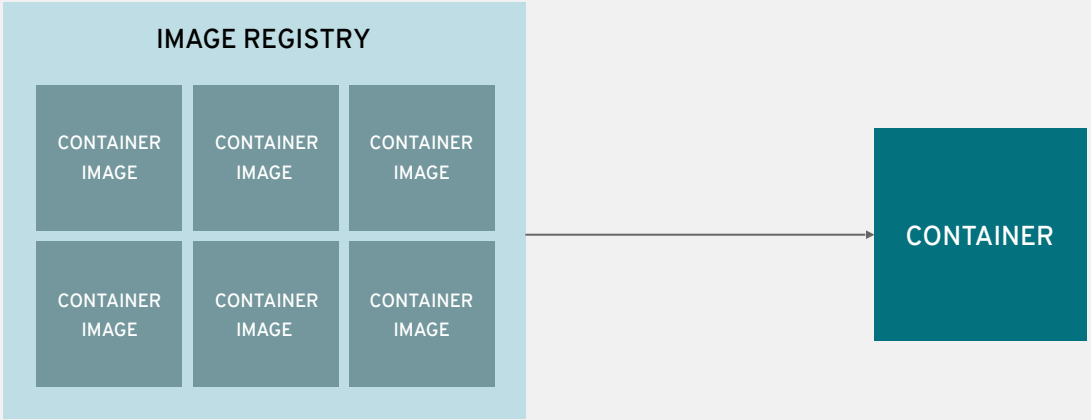
Lab 1	Getting Started
Lab 2	Enterprise Microservices with WildFly Swarm
Lab 3	Microservices with Spring Boot
Lab 4	Reactive Microservices with Eclipse Vert.x
Lab 5	Web UI with Node.js and AngularJS
Lab 6	Monitoring Application Health
Lab 7	Service Resilience and Fault Tolerance
Lab 8	Application Configuration
Lab 9	Continuous Delivery
Lab 10	Debugging Applications

OPENSHIFT CONCEPTS OVERVIEW

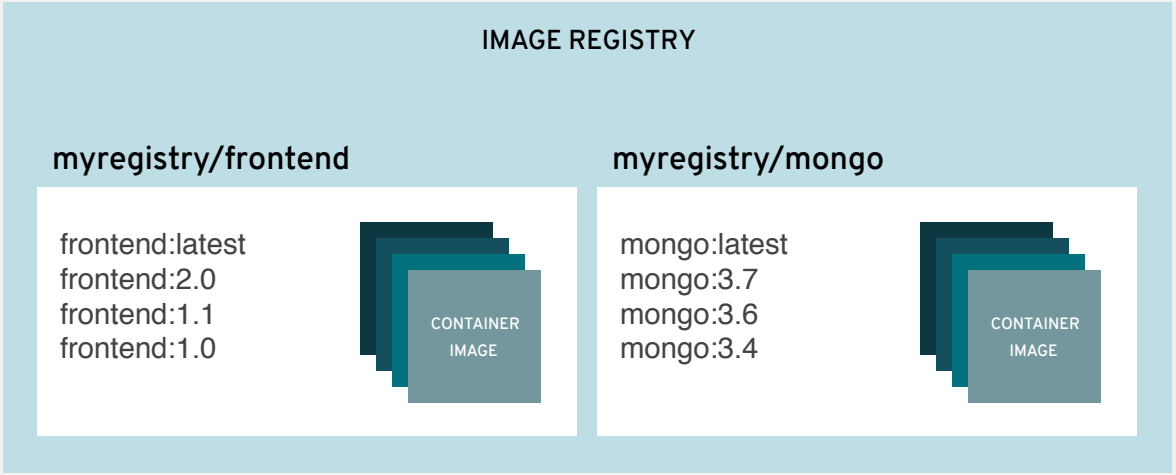
A container is the smallest compute unit

containers are created from container images

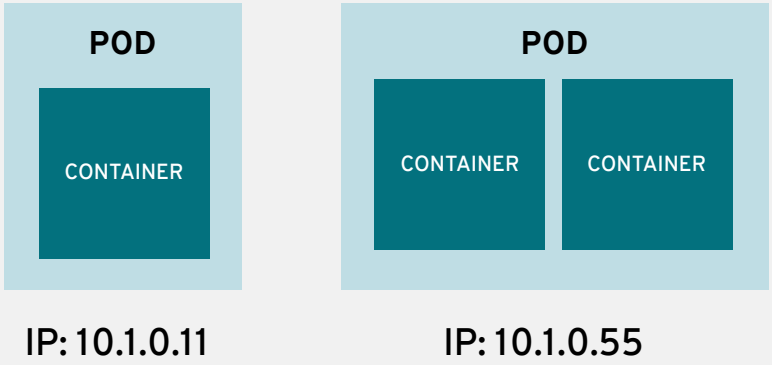
container images are stored in an image registry



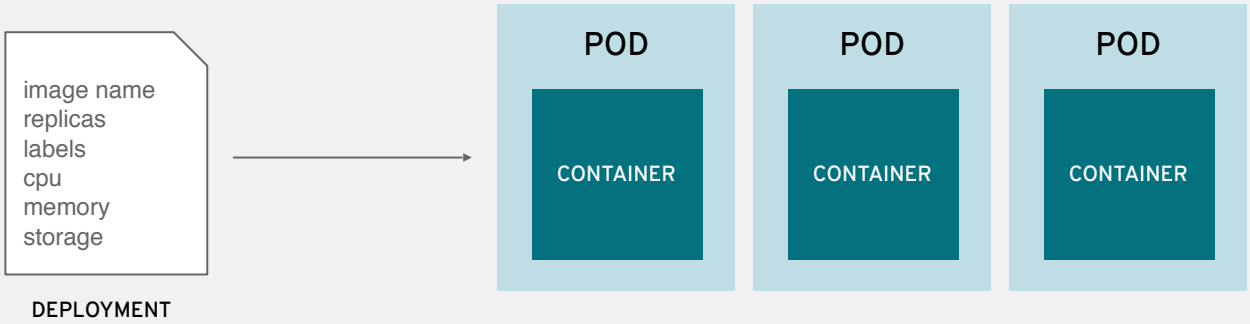
an image repository contains all versions of an image in the image registry



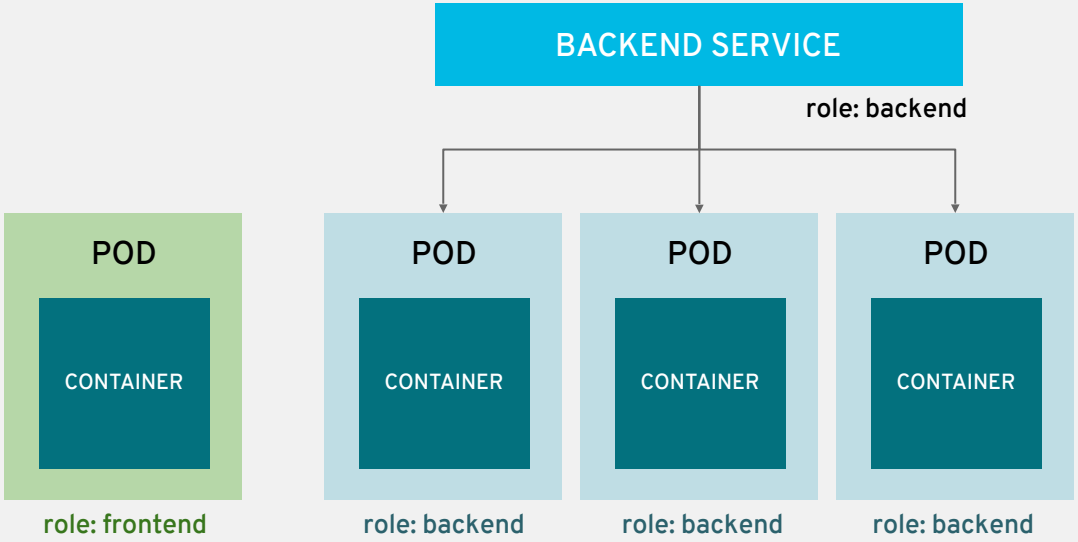
containers are wrapped in pods which are units of deployment and management



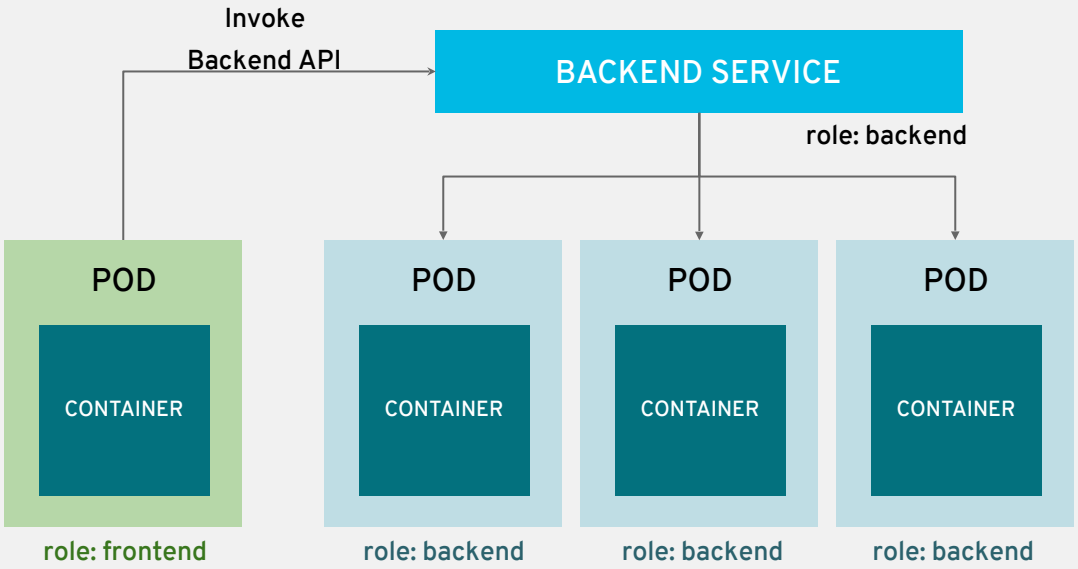
pods configuration is defined in a deployment



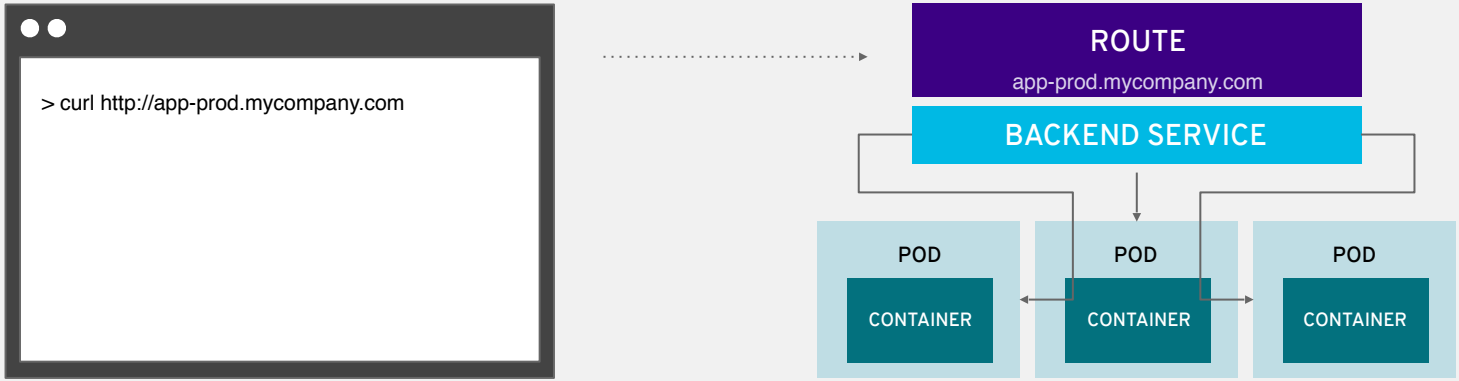
services provide internal load-balancing and service discovery across pods



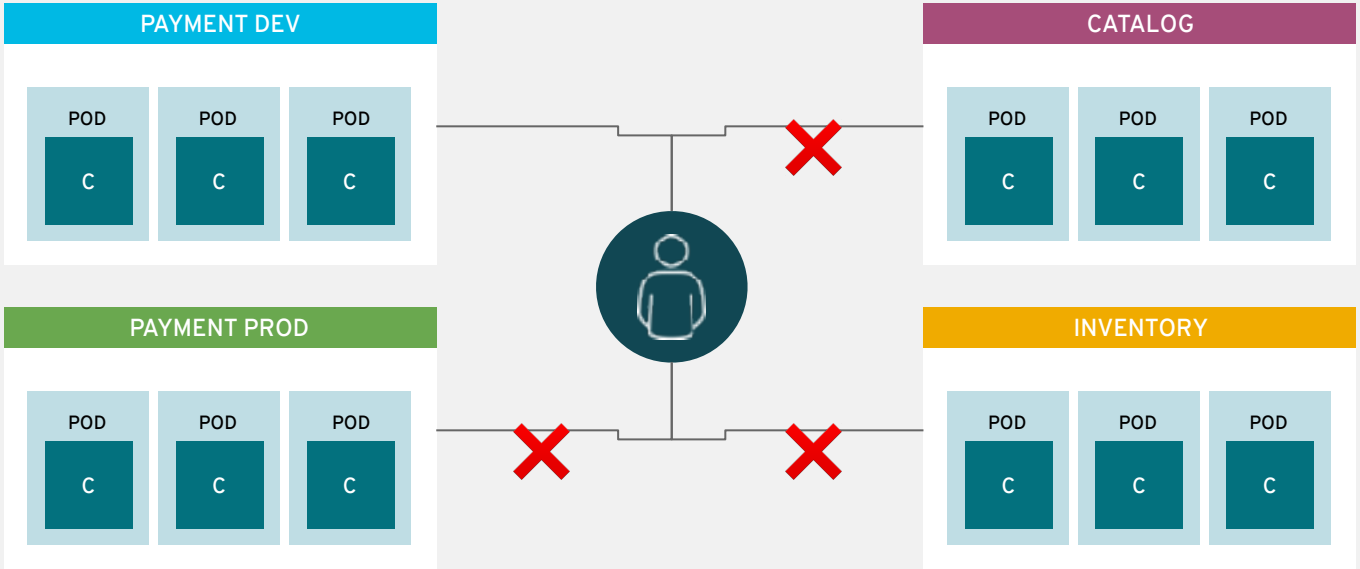
apps can talk to each other via services



routes add services to the external load-balancer and provide readable urls for the app



projects isolate apps across environments, teams, groups and departments



LAB 1

GETTING STARTED

LAB 1: Getting Started With OpenShift

- Explore OpenShift CLI
- Download lab projects

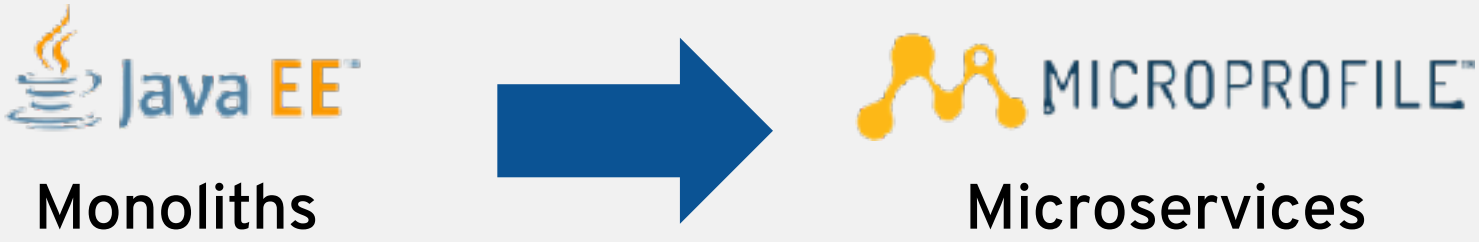
LAB 2

ENTERPRISE MICROSERVICES WITH WILDFLY SWARM

- Microservices for Java EE developers
- Small subset of Java EE technologies
- Package as an uber-jar
- Package only what you need
- Built from WildFly
- Project generator at launch.openshift.io

ECLIPSE MICROPROFILE

Enterprise Java for a Microservices Architecture



Monoliths

Microservices

BOOTSTRAPPING WILDFLY SWARM

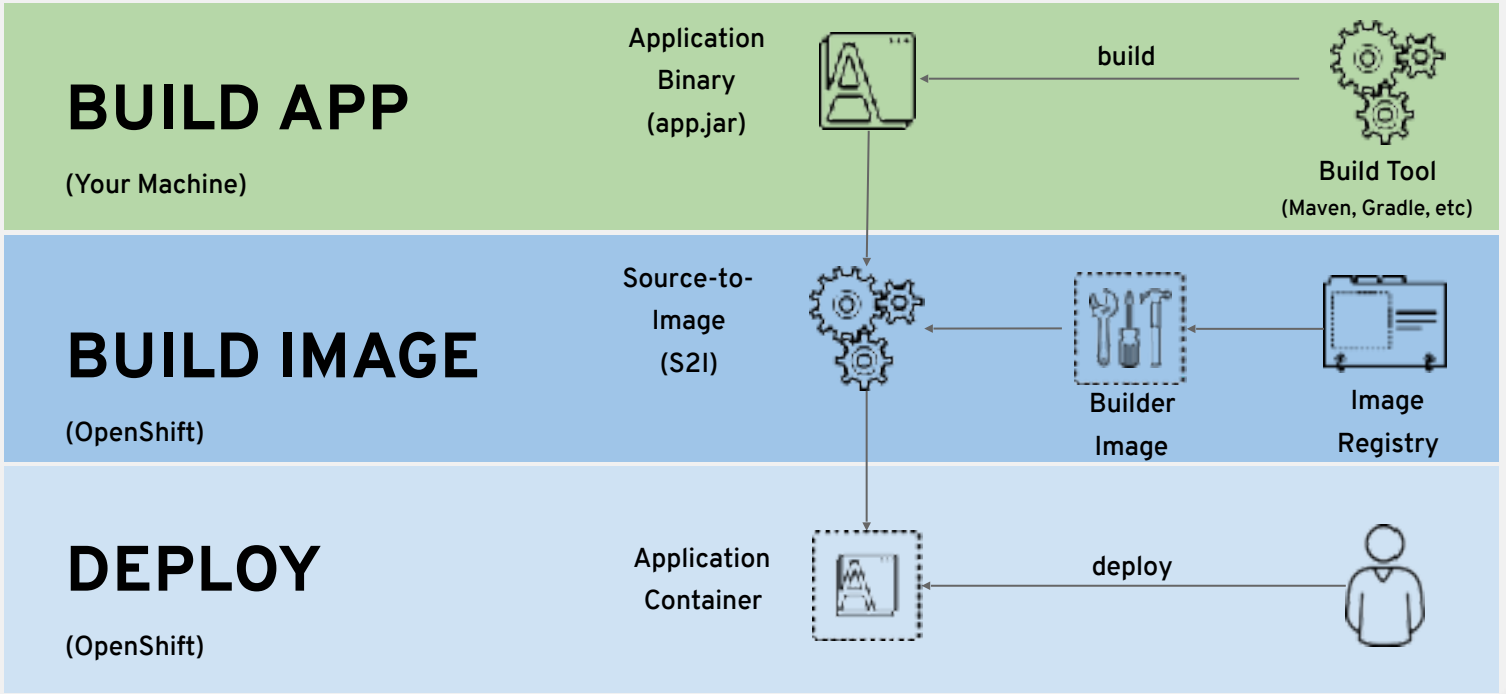
OPTION 1

```
public class InventoryMain {
  public static void main(String...args) {
    Swarm swarm = new Swarm();
    JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);
    deployment.addClass(InventoryResource.class); // Add REST resource
    ...
    swarm.start();
    swarm.deploy(deployment);
  }
}
```

OPTION 2

```
@ApplicationPath("/")
public class InventoryApplication extends Application {
}
```

DEPLOY APPS WITH SOURCE-TO-IMAGE(S2I)



DEPLOY APPS WITH SOURCE-TO-IMAGE(S2I)

BUILD APP (Your Machine)	`$ mvn package`
BUILD IMAGE (OpenShift)	`$ mvn fabric8:build`
DEPLOY (OpenShift)	`$ mvn fabric8:deploy`

DEPLOY APPS WITH SOURCE-TO-IMAGE(S2I)

BUILD APP

(Your Machine)

$ mvn package

BUILD IMAGE

(OpenShift)

$ mvn fabric8:build

DEPLOY

(OpenShift)

$ mvn fabric8:deploy

$ mvn fabric8:deploy

LAB 2: Enterprise Microservices with WildFly Swarm

- Explore WildFly Swarm Maven project
- Create a domain model
- Create a RESTful service
- Run WildFly Swarm locally
- Deploy WildFly Swarm on OpenShift

LAB 3

MICROSERVICES WITH SPRING BOOT

- Microservices for Developers using Spring Framework
  - 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
- More Red Hat technologies to come

BOOTSTRAPPING SPRING BOOT

```
@SpringBootApplication
public class CatalogApplication {
    public static void main(String[] args) {
        SpringApplication.run(CatalogApplication.class, args);
    }
}
```

LAB 3: Microservices with Spring Boot

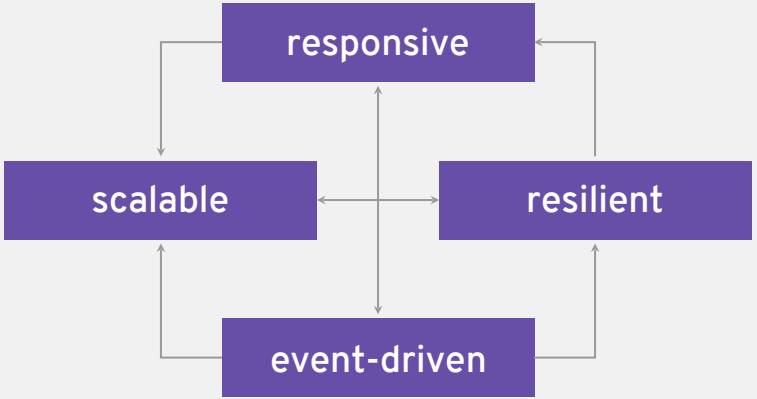
- Explore Spring Boot Maven project
- Create a domain model
- Create a RESTful service
- Run Spring Boot locally
- Deploy Spring Boot on OpenShift

LAB 4

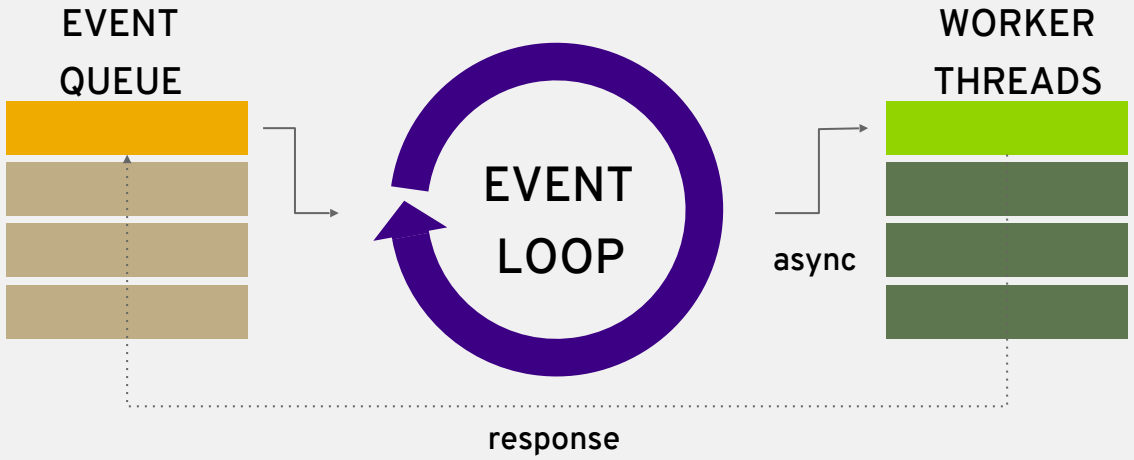
REACTIVE MICROSERVICES WITH ECLIPSE VERT.X

VERT.X

- Reactive Microservices for JVM
- Ideal for High Concurrency and Low Latency Services
- Lightweight Messaging
- Event Driven Non-Blocking I/O
- Un-opinionated
- Use with any framework including Spring Boot and WildFly Swarm



VERT.X EVENT LOOP



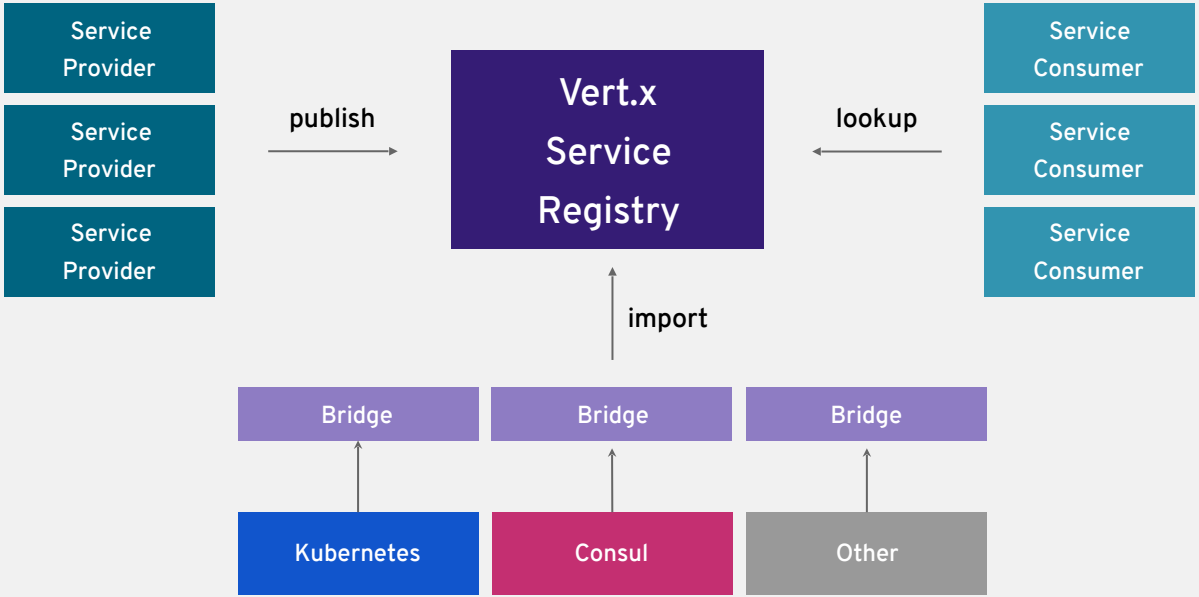
Handle Thousands of Requests
With Few Threads

FREE E-BOOK

Building Reactive Microservices in Java

bit.ly/free-vertx-ebook

SERVICE DISCOVERY



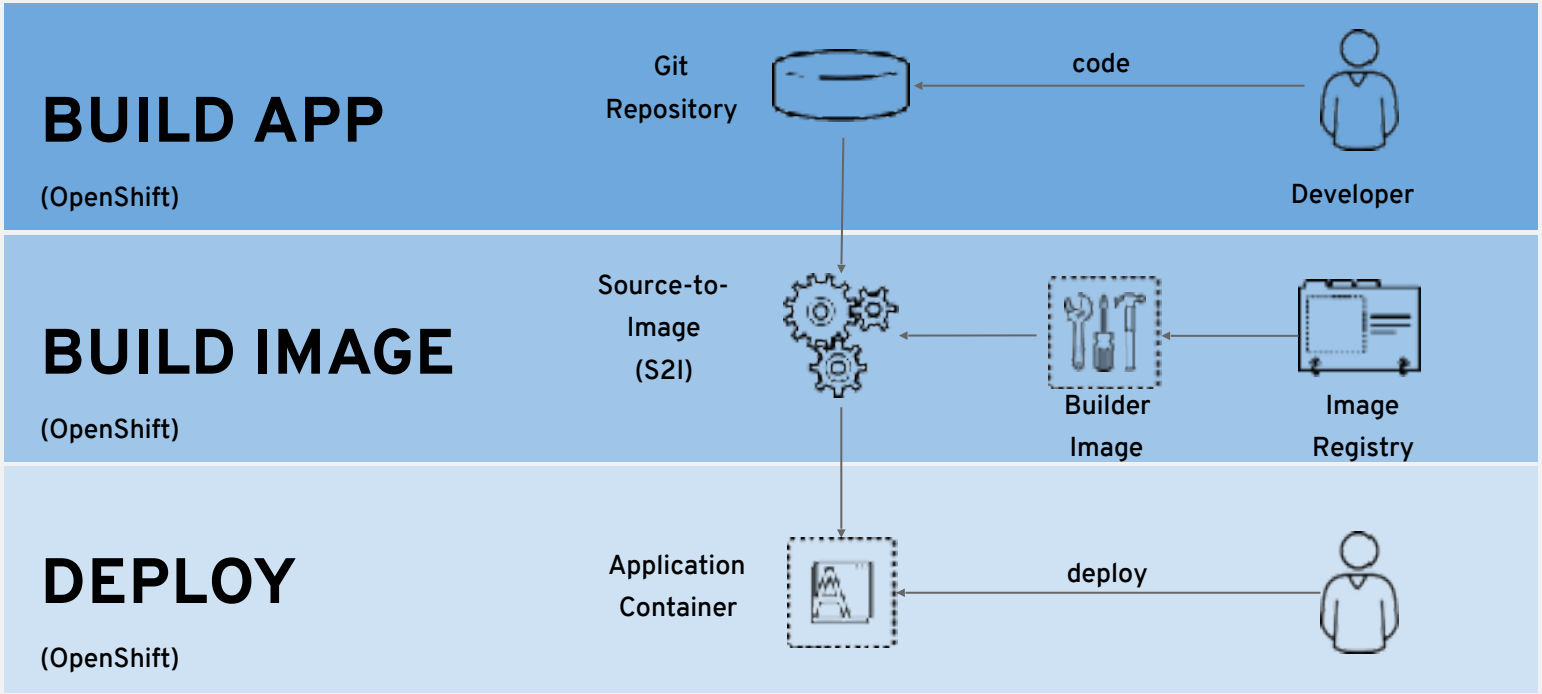
LAB 4: Reactive Microservices with Eclipse Vert.x

- Explore Vert.x Maven project
- Create an API gateway
- Run Vert.x locally
- Deploy Vert.x on OpenShift

LAB 5

WEB UI WITH
NODE.JS AND ANGULARJS

DEPLOY APP SOURCE CODE WITH S2I



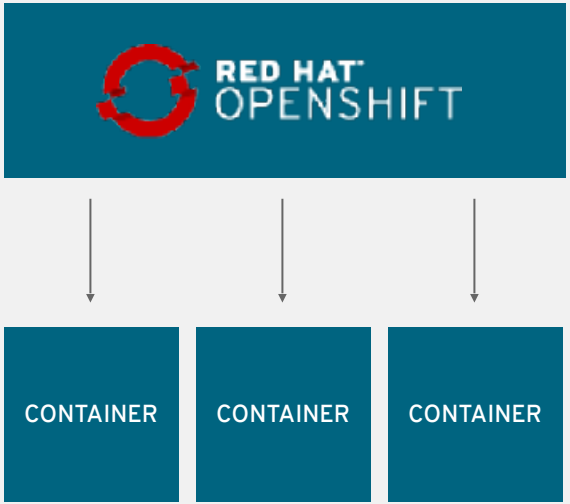
LAB 5: Web UI with Node.js and AngularJS

- Explore Node.js project
- Deploy Node.js and AngularJS on OpenShift

LAB 6

MONITORING APPLICATION HEALTH

HEALTH PROBES



PROBE TYPES

Is it ready?

Is it alive?

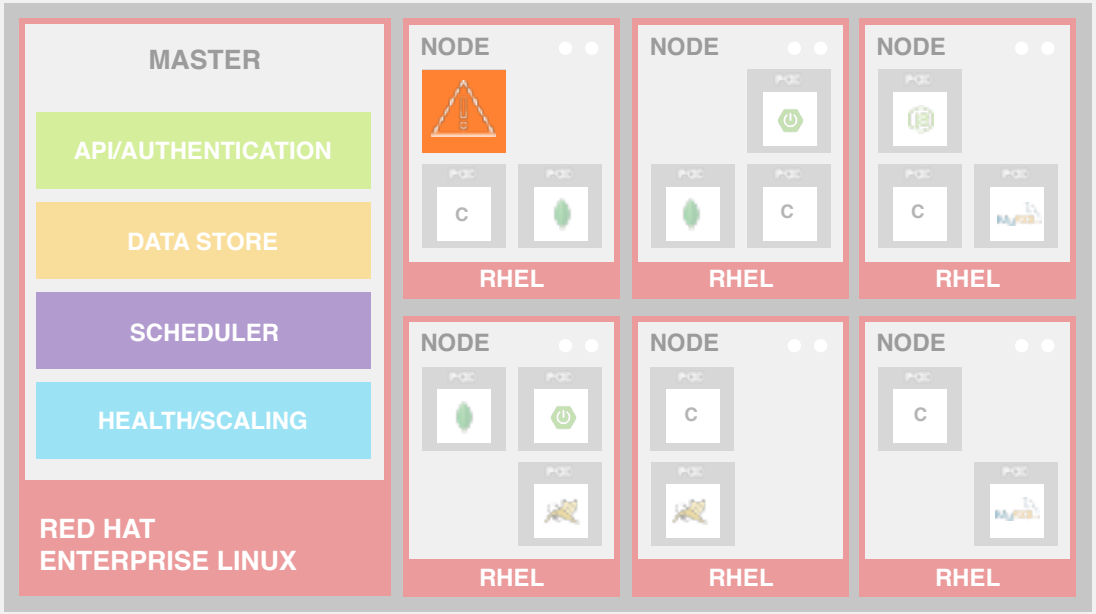
PROBE CHECKS

HTTP

Shell Command

TCP Port

AUTO-HEALING FAILED CONTAINERS



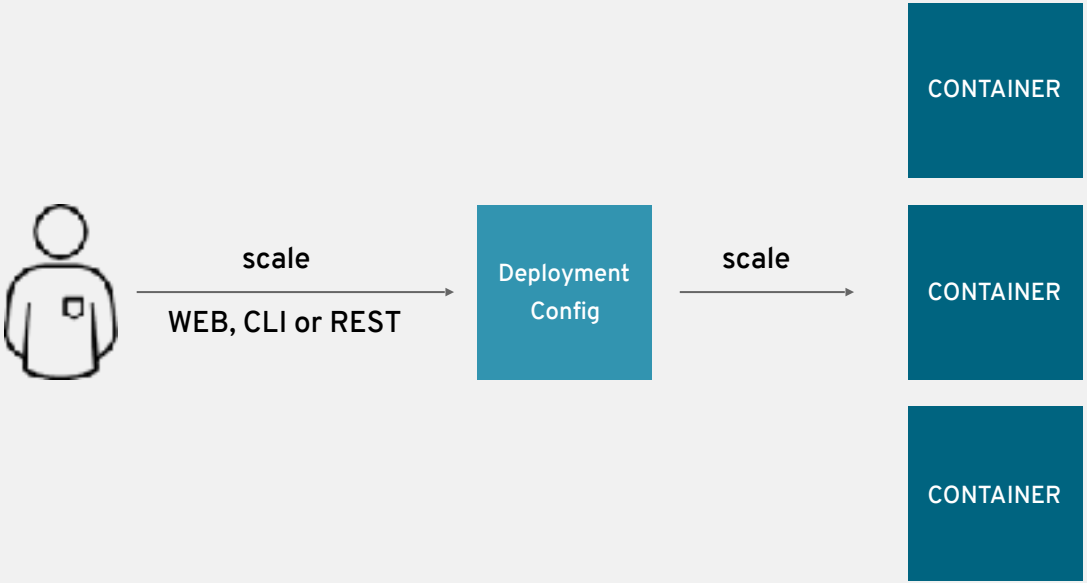
LAB 6: Monitoring Application Health

- Review Health REST endpoints in the services
- Add health probes to Catalog service
- Add health probes to Inventory service
- Add health probes to API Gateway service
- Add health probes to Web front-end
- Explore container metrics

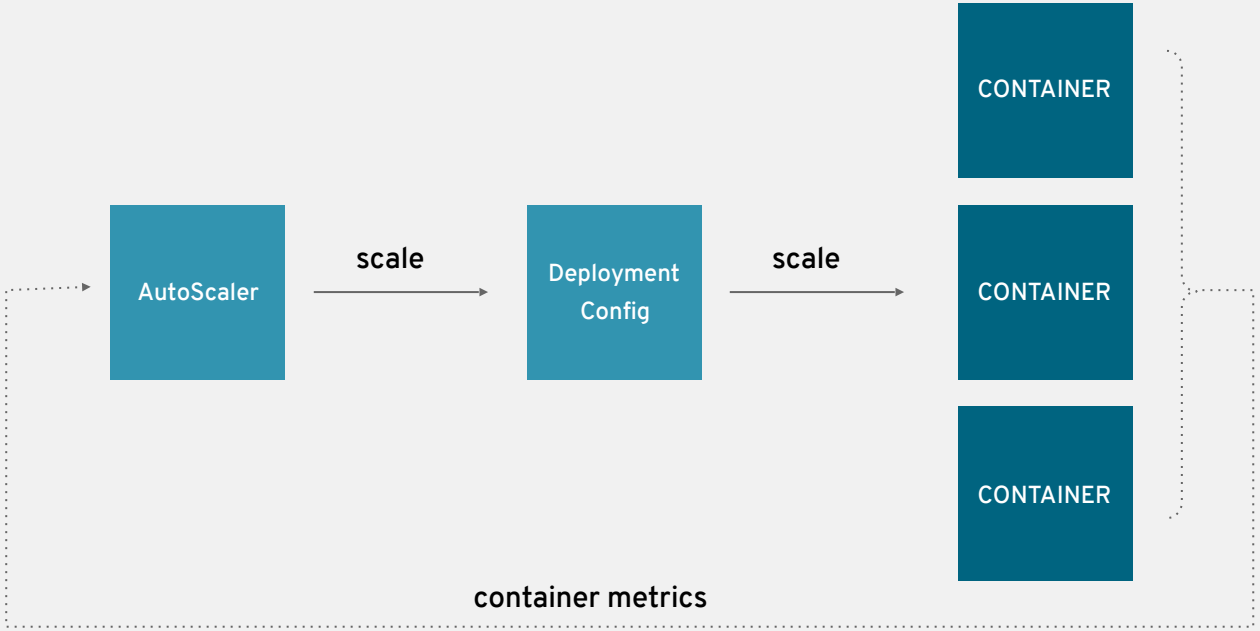
LAB 7

SERVICE RESILIENCE AND FAULT TOLERANCE

SCALING PODS

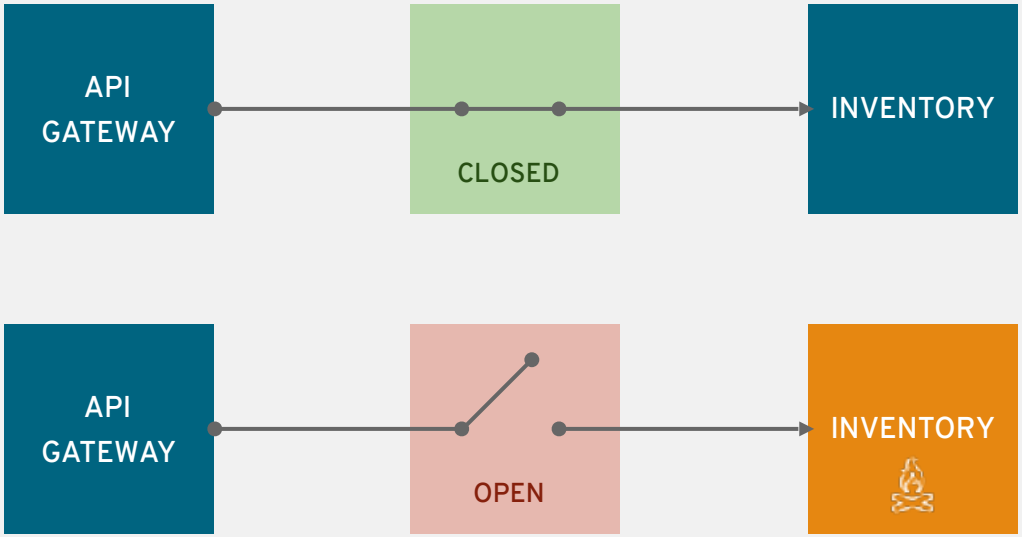


AUTO-SCALING PODS



Preventing Cascading Failures to Bring Down the System

CIRCUIT BREAKER PATTERN



LAB 7: Service Resilience and Fault Tolerance

- Scale up Web front-end
- Add auto-scaling to Web front-end
- Explore auto-healing failed application pods
- Add a Circuit Breaker to API Gateway
- Deploy the new API Gateway on OpenShift

LAB 8

APPLICATION CONFIGURATION

CONFIG MAPS IN OPENSHIFT

- Config maps inject config data into containers
- Config maps can hold
  - Properties (key-value pairs)
  - Files (JSON, XML, etc)
- Containers see config maps as
  - Files on the filesystem
  - Environment variables
- Secrets are like config maps for sensitive data
  - Credentials, certificates, SSH keys, etc

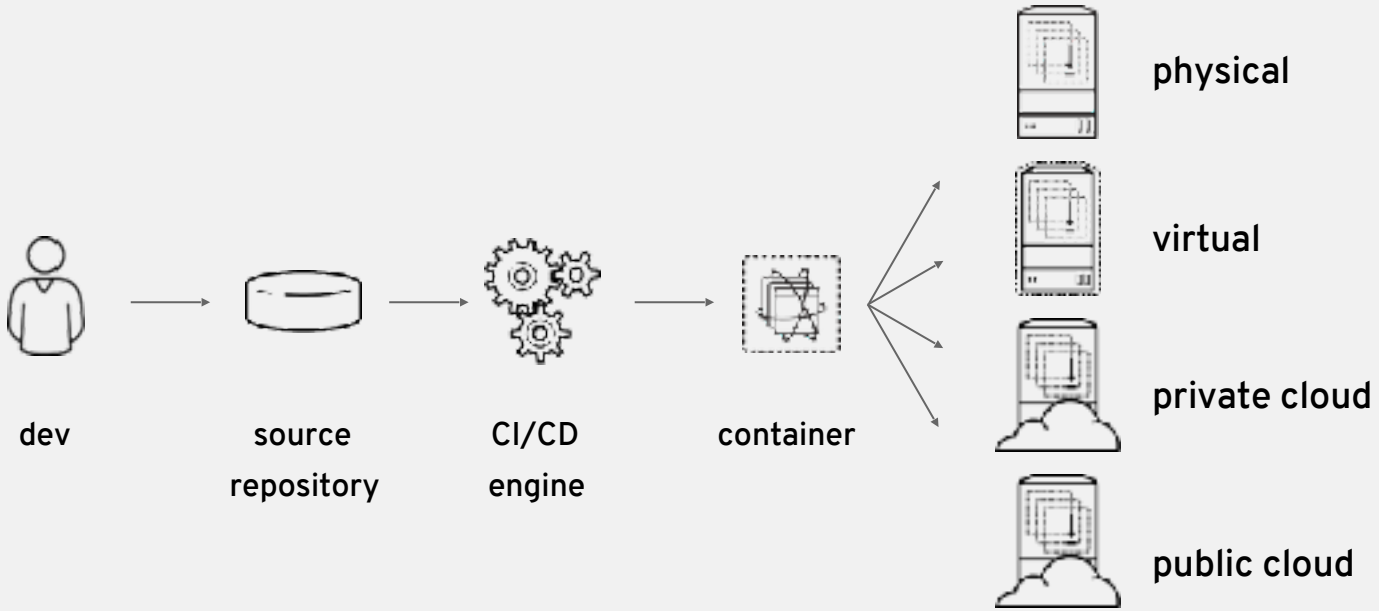
LAB 8: Application Configuration

- Create Inventory and Catalog PostgreSQL databases on OpenShift
- Externalize WildFly Swarm (Inventory) configuration
- Externalize Spring Boot (Catalog) configuration
- Review externalizing sensitive configuration data

LAB 9

CONTINUOUS DELIVERY

DEPLOYMENT PIPELINES



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("...")
          }
        }
      }
    }
  }
}
```

LAB 9: Automating Deployments Using Pipelines

- Create a Git Repository for Inventory source code
- Push Inventory source code to the Git repository
- Define the deployment pipeline as a `Jenkinsfile`
- Create an OpenShift Pipeline using the `Jenkinsfile`
- Add a Webhook to run the pipeline on every code change

LAB 10

DEBUGGING APPLICATIONS

LAB 10: Debugging Applications

- Investigate the bug
- Enable remote debugging on an OpenShift pod
- Run remote debug and line-by-line code execution
- Fix the bug

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

redhat.

How to Start?

Go to bit.ly/cloud-native-roadshow

How it Goes?

Instructors explain the concepts before each lab

and then you are off to do your magic!

Having an Issue?

Raise your hand. An instructor will come to you.