

Building a MicroProfile and Jakarta EE application with Open Liberty

& OpenJ9

Jamie Lee Coleman
Software Developer/Advocate
Twitter: @jamie_lee_c
LinkedIn: jamie-coleman



JAKARTA™ EE

OpenJ9

Demo Environment

skills Network Labs

Skills Network Labs



Welcome to IBM Developer Skills Network to access cloud-hosted guides

The Open Liberty cloud-hosted guides run on the IBM Developer Skills Network. To prevent improper use, we require you to sign into the network to access the guides. You can sign in with an IBM Development Skills Network account or through one of the following single sign-on providers.

First time here? [Create an IBM Developer Skills Network account.](#)

Email

The email address you used to register with IBM Developer Skills Network

Password

[Need help logging in?](#)

Sign in



Sign In with GitHub



Sign In with Google



Sign In with IBMid



Sign In with LinkedIn

By signing in, I acknowledge that I understand how IBM Developer Skills Network is using my basic personal data, and that I am at least 16 years of age. Our [Privacy Notice](#) provides more details.

[Community](#) [Privacy](#) [Terms of use](#) [Accessibility](#)

If you have any questions, send an email to openliberty@groups.io.

© Copyright IBM Corp. 2021. All rights reserved except where noted. edX, Open edX and their respective logos are registered trademarks of edX Inc.

Skills Network Labs

The screenshot displays the Skills Network Labs interface. On the left, a guide titled "Getting started" is shown, indicating it is "Step 3 of 9". The guide provides instructions on how to open a terminal, navigate to the project directory, clone the repository, and build the application. On the right, a code editor shows the contents of a `pom.xml` file. The file is a Maven POM for a project named "guide-rest-intro" with a version of "1.0-SNAPSHOT". It includes various configurations for build, reporting, and dependencies. The terminal at the bottom shows the execution of the `git clone` command, which has successfully cloned the repository into the "guide-rest-intro" directory.

Skills Network Labs | Open Liberty

Instructions

◀ Step 3 of 9 ▶

Getting started

To open a new command-line session, select **Terminal > New Terminal** from the menu of the IDE.

Run the following command to navigate to the `/home/project` directory:

```
cd /home/project
```

The fastest way to work through this guide is to clone the [Git repository](#) and use the projects that are provided inside:

```
git clone https://github.com/openliberty/guide-rest-intro.git
cd guide-rest-intro
```

The **start** directory contains the starting project that you will build upon.

The **finish** directory contains the finished project that you will build.

Try what you'll build

The **finish** directory in the root of this guide contains the finished application. Give it a try before you proceed.

To try out the application, first go to the **finish** directory and run the following Maven goal to build the application and deploy it to Open Liberty:

```
cd finish
```

Lab | IBMCloud | Launch Application

File Edit Selection View Go Run Terminal Help

EXPLORER

- PROJECT
 - guide-rest-intro
 - github
 - finish
 - src
 - pom.xml
 - scripts
 - start
 - gIgnore
 - CONTRIBUTING.md
 - LICENSE
 - README.adoc

CODEWIND

JAVA DEPENDENCIES

```
pom.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
3 <modelVersion>4.0.0</modelVersion>
4
5 <groupId>io.openliberty.guides</groupId>
6 <artifactId>guide-rest-intro</artifactId>
7 <version>1.0-SNAPSHOT</version>
8 <packaging>war</packaging>
9
10 <properties>
11 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
13 <maven.compiler.source>1.8</maven.compiler.source>
14 <maven.compiler.target>1.8</maven.compiler.target>
15 <!-- Liberty configuration -->
16 <!-- tag:defaultHttpPort[] -->
17 <liberty.var.default.http.port>9080</liberty.var.default.http.port>
18 <!-- end:defaultHttpPort[] -->
19 <!-- tag:defaultHttpsPort[] -->
20 <liberty.var.default.https.port>9443</liberty.var.default.https.port>
21 <!-- end:defaultHttpsPort[] -->
22 <!-- tag:appContextRoot[] -->
23 <liberty.var.app.context.root>LibertyProject</liberty.var.app.context.root>
24 <!-- end:appContextRoot[] -->
25 </properties>
26
27 <dependencies>
28 <!-- Provided dependencies -->
```

theia@theiadocker-jcoleman: /home/project/open-liberty-masterclass/start/coffee-shop theia@theiadocker-jcoleman: /home/project x

```
theia@theiadocker-jcoleman: /home/project$ git clone https://github.com/openliberty/guide-rest-intro.git
Cloning into 'guide-rest-intro'...
remote: Enumerating objects: 1785, done.
remote: Counting objects: 100% (96/96), done.
remote: Compressing objects: 100% (83/83), done.
remote: Total 1785 (delta 25), reused 27 (delta 2), pack-reused 1609
Receiving objects: 100% (1785/1785), 333.00 KiB | 4.76 MiB/s, done.
Resolving deltas: 100% (626/626), done.
theia@theiadocker-jcoleman: /home/project$ cd guide-rest-intro
```

In 1, Col 1 LF UTF-8 Spaces: 4 XML

A Full Open

Stack

A Full Open Stack

MicroProfile

Jakarta EE

Open Liberty

OpenJ9



JAKARTA™ EE



OpenJ9

Eclipse

MicroProfile

What is MicroProfile?

- Eclipse MicroProfile is an open-source community specification for Enterprise Java microservices
- A community of individuals, organizations, and vendors collaborating within an open source (Eclipse) project to bring microservices to the Enterprise Java community

microprofile.io





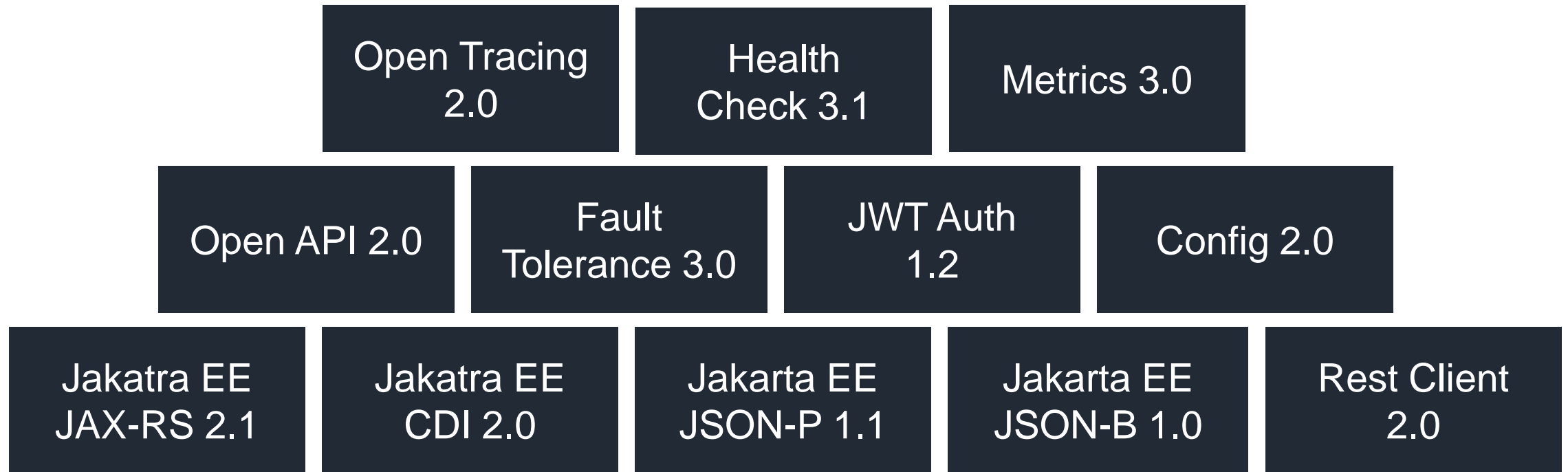
MicroProfile Contributors



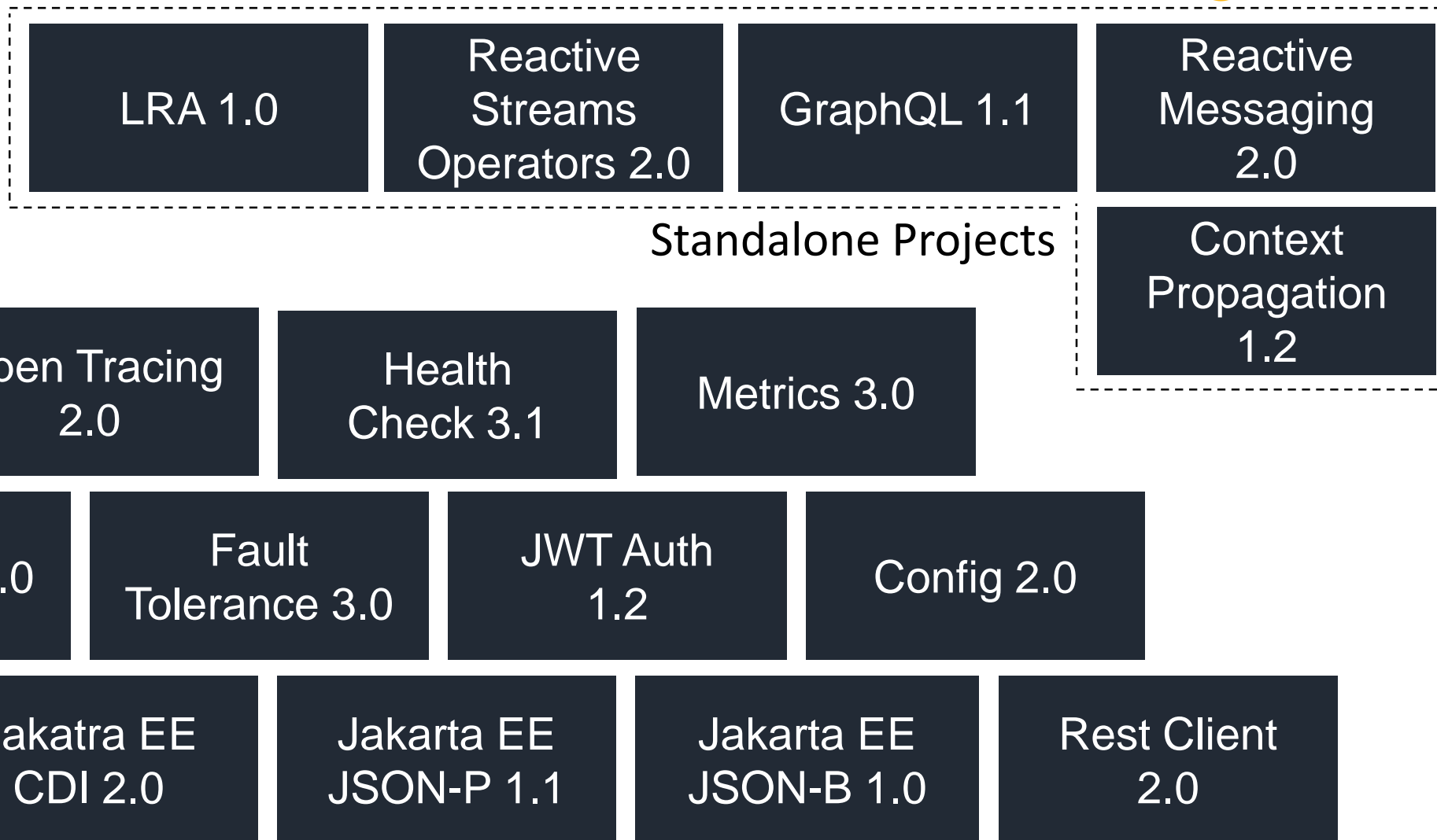
MicroProfile Vendors/Implementations



MicroProfile 4.1 Stack



MicroProfile 4.1 Stack



Open Liberty Overview



Docs Downloads Support

Fork the code

Open Liberty

An IBM Open Source Project

Jump on board and work at lightspeed

Build cloud-native apps and microservices while running only what you need. Open Liberty is the most flexible server runtime available to Java™ developers in this solar system. [Here's why.](#)

Download (122 MB)

[View more downloads](#)

Open Liberty Overview

Focus on **code**

Easy to make **fast** and **iterative changes**

Easy to write **tests**

True-to-production testing (as much as possible)

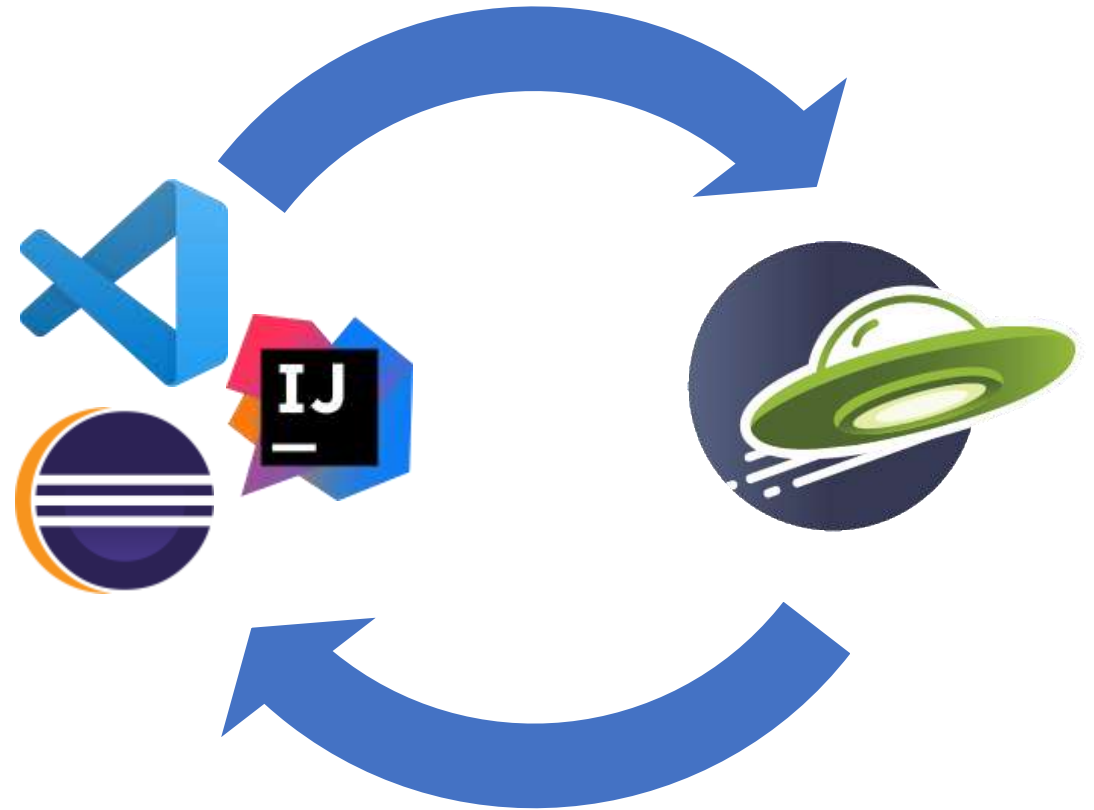
Ready for **containers**

Not-in-your-way tools and flexibility

Developer Experience: dev mode

- Boosts developer productivity
- Immediate feedback for code and config changes
- No re-build necessary

mvn liberty:dev



Developer productivity

IDEs



Repositories



Build



APIs



Testing



Jesse Gallagher
@Gidgerby

Have I mentioned lately how much of a delight @OpenLibertyIO is to work with? It's just thoroughly pleasant.



Tim Zöller
@javahippie

The @OpenLibertyIO dev mode is one of the best hot-reload features I have ever worked with, I am seriously impressed!

Just Enough Application Server

You control which features are loaded into each server instance

Java EE



```
<feature>jsf-2.3</feature>
```

jsp-2.3

jsf-2.3

servlet-4.0

http-2.0

appmgr

Kernel



API Support

- First shipped in WAS 8.5 in 2012
 - Servlet + JSP + JPA
- Web Profile 6 in 2014
- Java EE 7 in 2016 – first commercial product to certify
- Java EE 8 in 2018 – first to certify
- Jakarta EE 8 in 2019 – first to certify
- Eclipse MicroProfile – first to deliver 1.0-1.4, 2.0-2.1, 3.0



Liberty Zero Migration

- Zero config migration
 - Write once, run forever
- Zero app migration
 - No behavior changes in existing features
 - New behaviors in new features
- Choose your Java
 - Java 14, 11, 8
 - AdoptOpenJDK
 - IBM
 - OpenJDK
 - Oracle



Developer-Oriented Docs & Guides

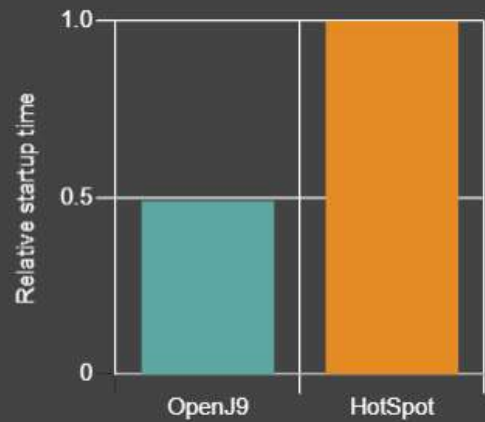


A screenshot of the Open Liberty website's 'Guides' page. The browser address bar shows 'openliberty.io/guides/'. The page has a dark blue header with navigation links: 'Get Started', 'Guides', 'Docs', 'Support', and 'Blog'. Below the header, the word 'Guides' is prominently displayed, followed by the tagline 'The quickest way to learn all things Open Liberty, and beyond!'. A search box labeled 'Filter guides' is visible on the right. The main content area is organized into sections: 'DEVELOP (37 guides)' with sub-sections like 'Getting started', 'RESTful service', and 'Reactive service'; 'BUILD AND TEST (10 guides)' with sub-sections like 'Build', 'Test', and 'Containerize'; and 'DEPLOY (10 guides)' with sub-sections like 'Kubernetes'. The 'Getting started' section is currently selected, showing two guide cards: 'Getting started with Open Liberty' (25 minutes) and 'Injecting dependencies into microservices' (15 minutes). The 'RESTful service' section shows four guide cards: 'Creating a RESTful web service', 'Consuming RESTful services with template interfaces', 'Consuming a RESTful web service', and 'Documenting RESTful APIs'. A sidebar on the left contains a list of all guides categorized by section.

OpenJ9

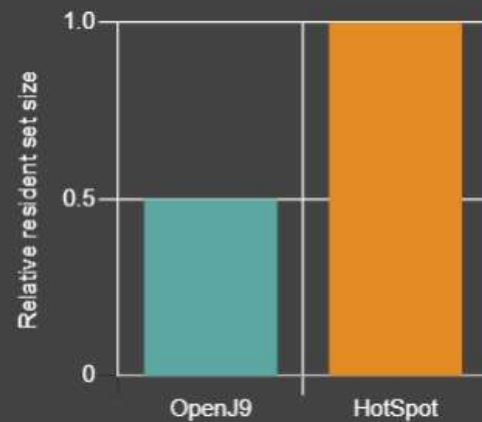
Overview of OpenJ9

51% faster startup time



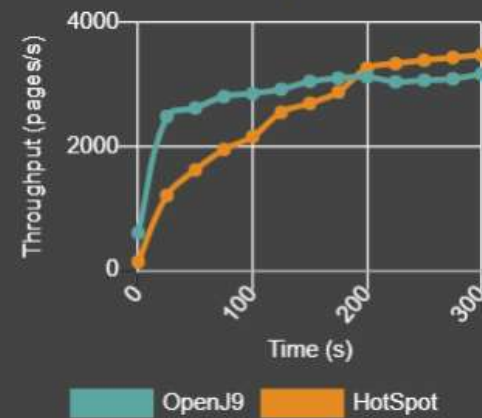
By using shared classes cache and AOT technology, OpenJ9 starts in roughly half the time it takes HotSpot.

50% smaller footprint after startup



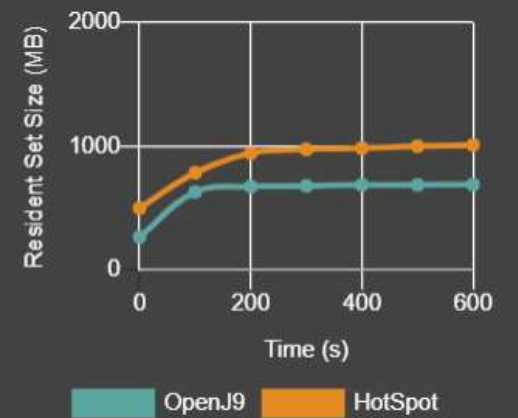
After startup, the OpenJ9 footprint is half the size of HotSpot, which makes it ideal for cloud workloads.

Faster ramp-up time in the cloud



OpenJ9 reaches peak throughput much faster than HotSpot making it especially suitable for running short-lived applications.

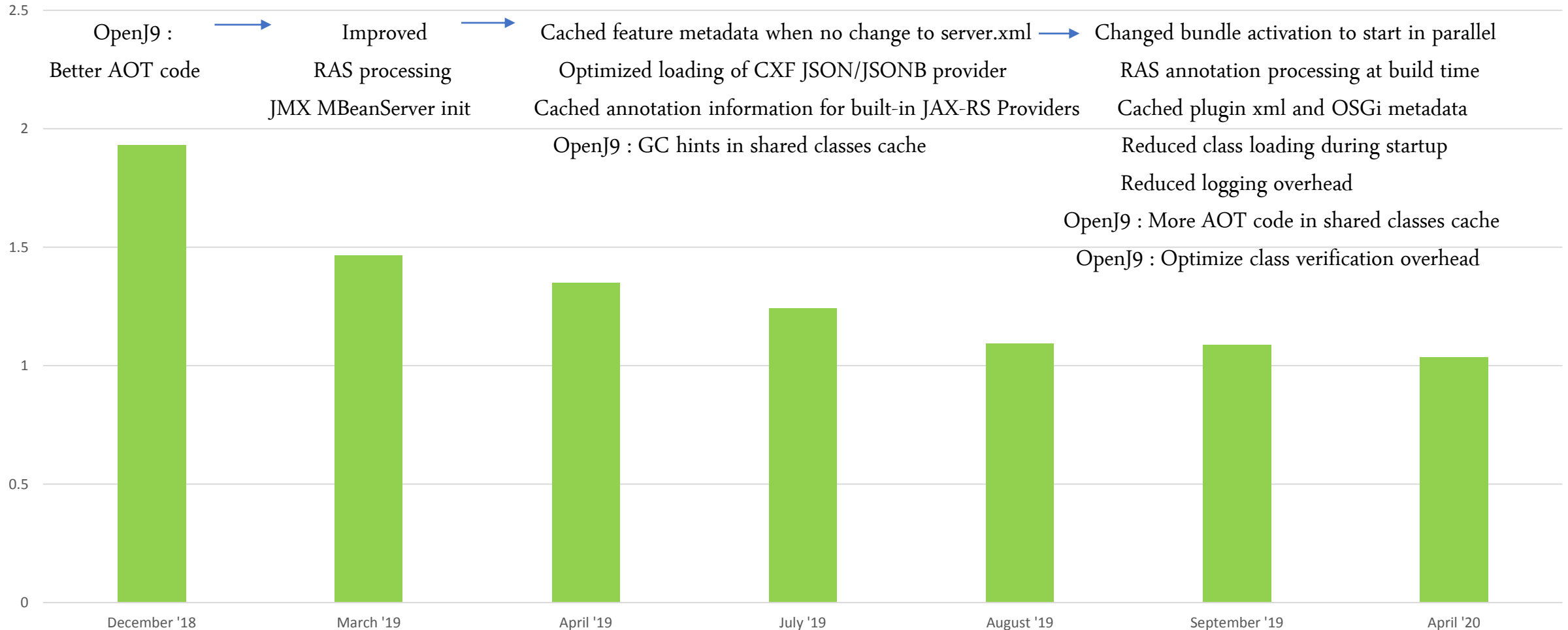
33% smaller footprint during load



Consistent with the footprint results after startup, the OpenJ9 footprint remains much smaller than HotSpot when load is applied.

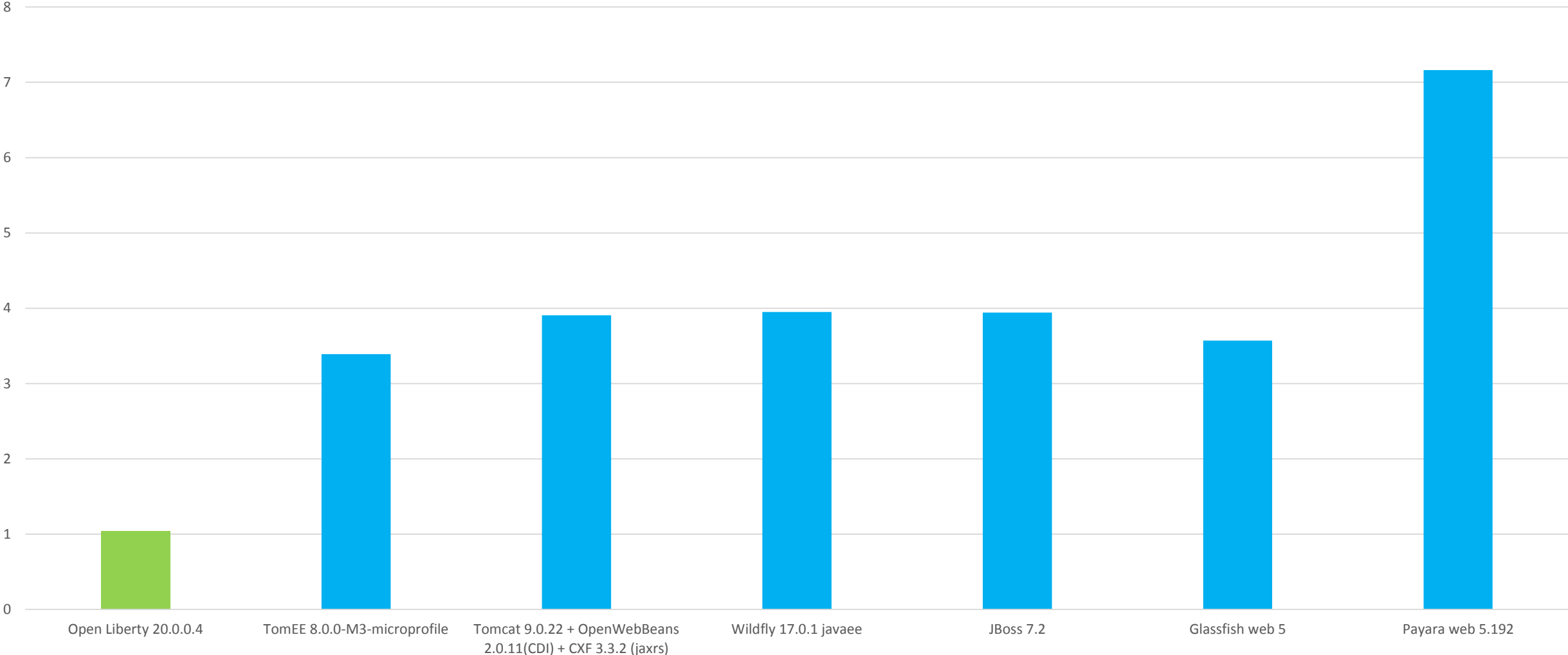
Open Liberty with OpenJ9: the road to one second startup time

2018-2020 Progression of OpenLiberty+OpenJ9 startup time (seconds)



Open Liberty startup time comparison (using OpenJ9 JVM)

PingPerf application startup time with OpenJ9 Shared Classes Cache (in seconds)



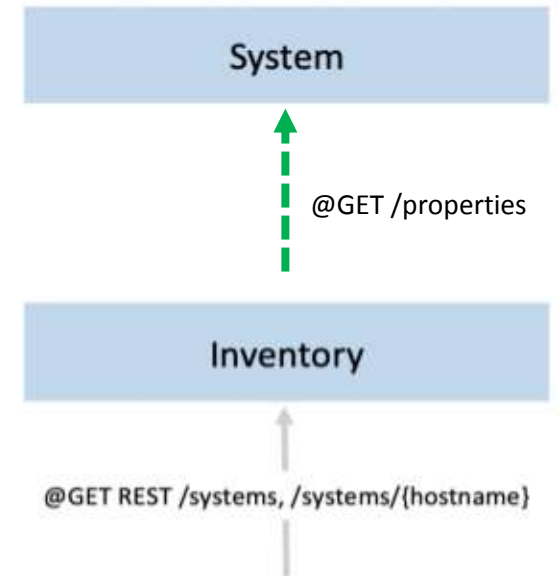
Lab 1

RESTful Services

Lab's Application

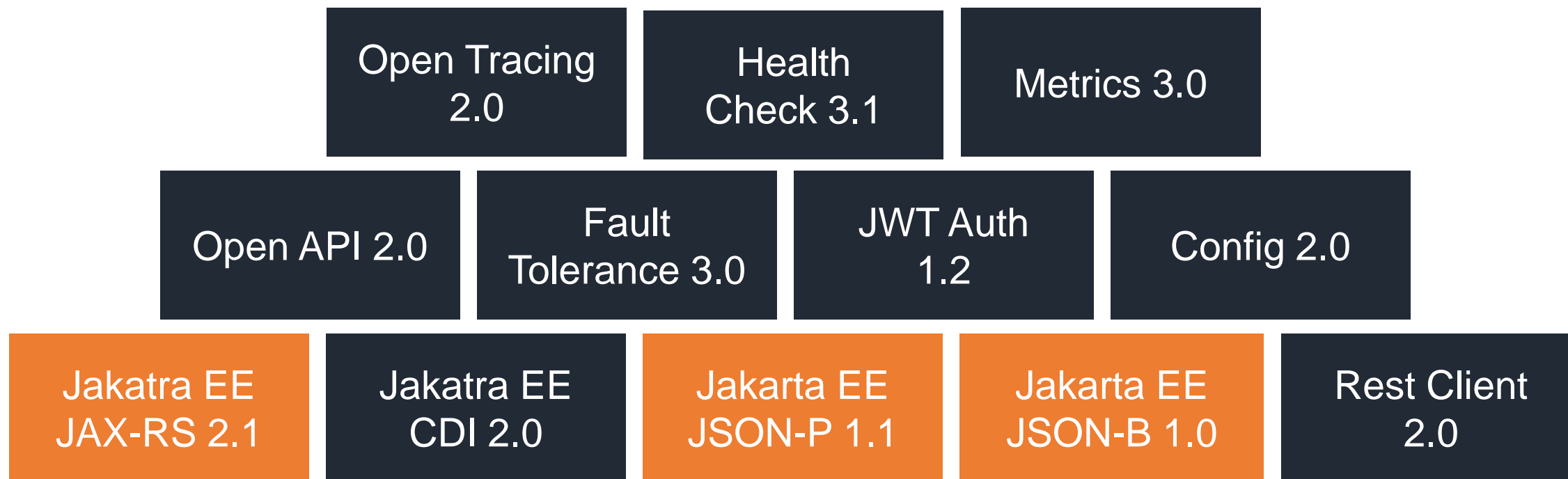
Two microservices: System and Inventory.

- The **system** service returns the system property information for a specific host:
<http://localhost:9080/system/properties>
- The **inventory** service tracks the the number of systems you have and it invokes the **system** service to retrieve information about a system:
<http://localhost:9080/inventory/systems>





MicroProfile 4.1 Stack





JAX-RS



```
@Path("/brews")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public class BrewsResource {

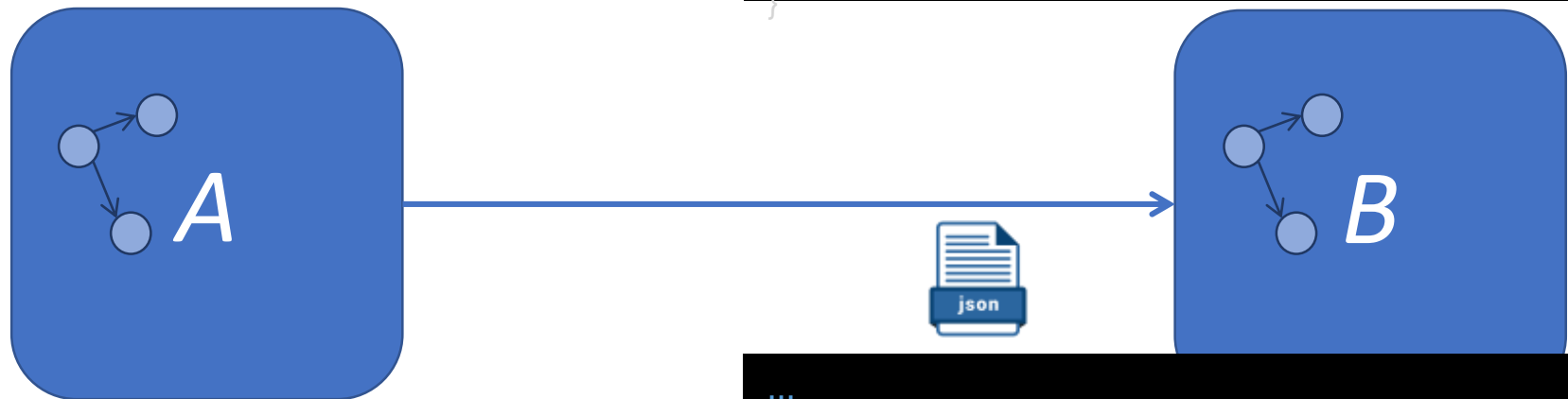
    @POST
    public Response startCoffeeBrew(CoffeeBrew brew) {...}
}
```

MicroProfile Core Technologies



JSON-B & JSON-P

```
public class CoffeeBrew {  
  
    private CoffeeType type;  
  
    public CoffeeType getType() {  
        return type;  
    }  
    public void setType(CoffeeType type) {  
        this.type = type;  
    }  
}
```



```
...  
  
@POST  
@Consumes(MediaType.APPLICATION_JSON)  
public Response startCoffeeBrew(CoffeeBrew brew) {  
    CoffeeType coffeeType = brew.getType();  
    ...  
}
```

<https://openliberty.io/docs/21.0.0.6/json-p-b.html>

Time To Code



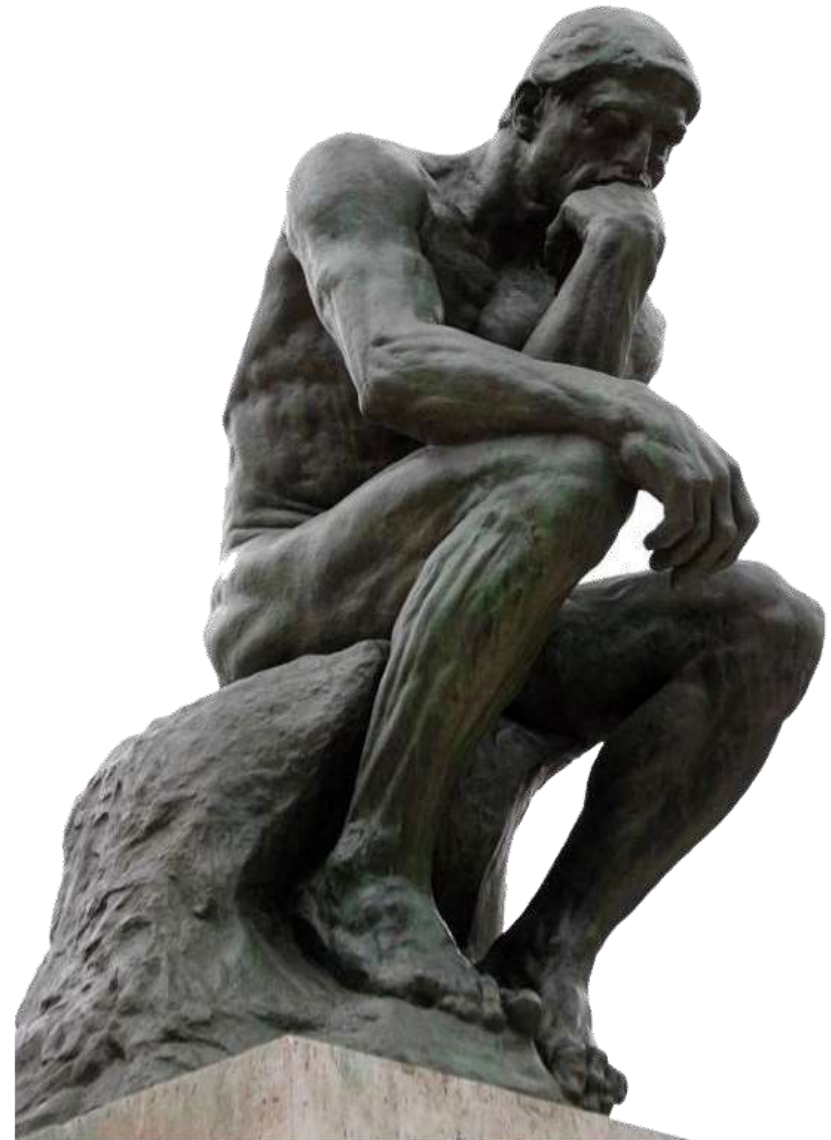
To get started visit the following URL in your browser:

<https://openliberty.skillsnetwork.site/cloud-native-java-made-easy-microprofile-jakarta-ee>

Lab 2

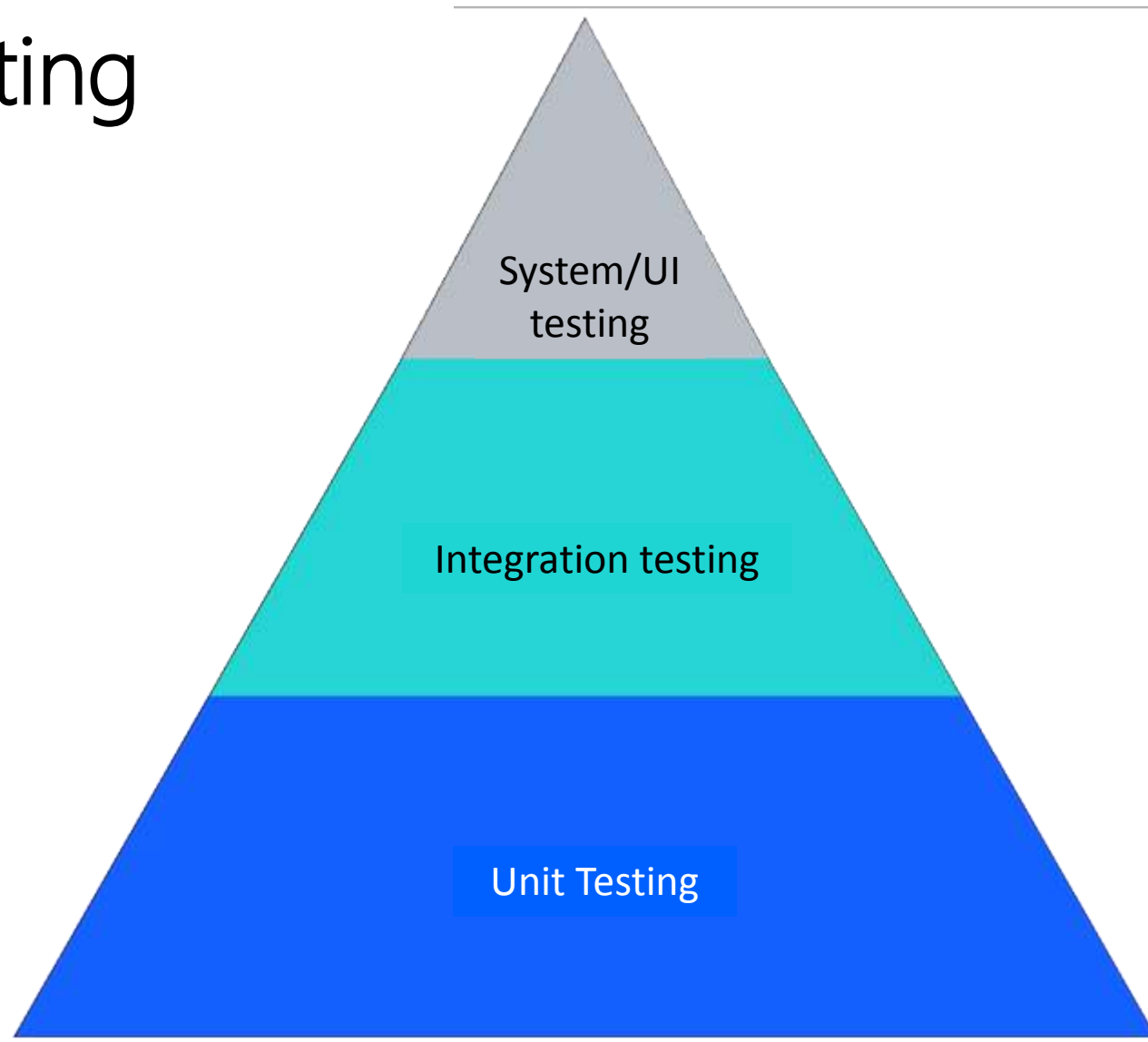
True to Production Testing

Why do we write
automated tests?

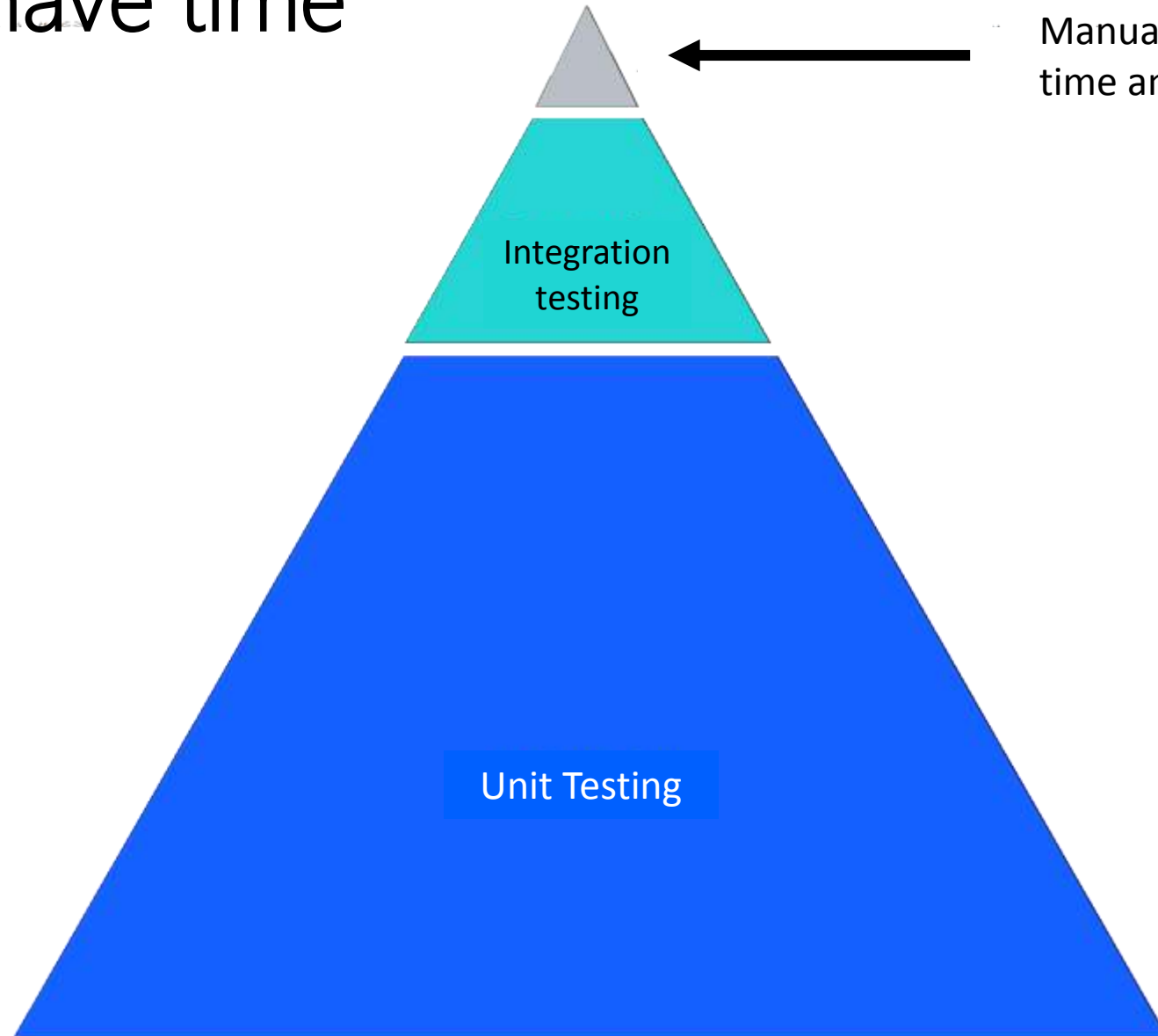


To have confidence that our applications
work the way we want them to!

Types of testing



What we have time to do



Manual Testing if I have time and remember

Problem with just using Unit tests

- ✓ Tap turns on
- ✓ Tap turns off
- ✓ Drain works
- ✓ Sink does not overflow





THE TWELVE-FACTOR APP

1. Codebase
2. Dependencies
3. Config
4. Backing Services
5. Build, Release, Run
6. Stateless Processes
7. Port Binding
8. Concurrency
9. Disposability
10. Dev-Prod Parity
11. Logs
12. Admin processes

Dev-Prod Parity

Keep development, staging, and production as similar as possible

Comprised of three common issues:

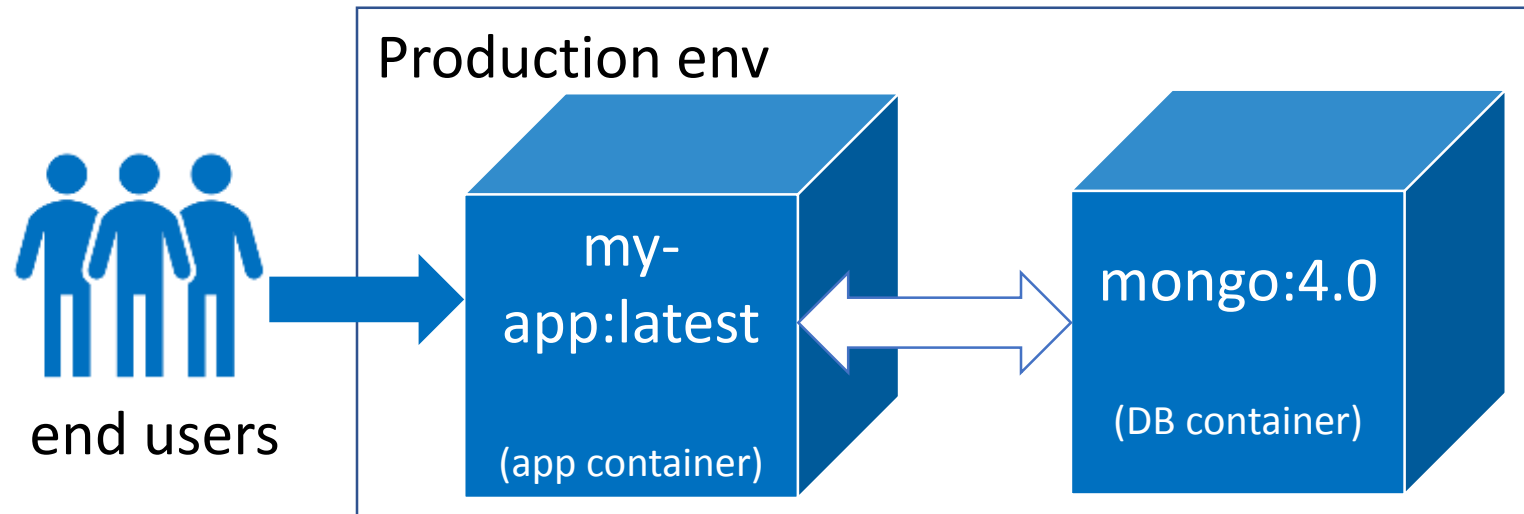
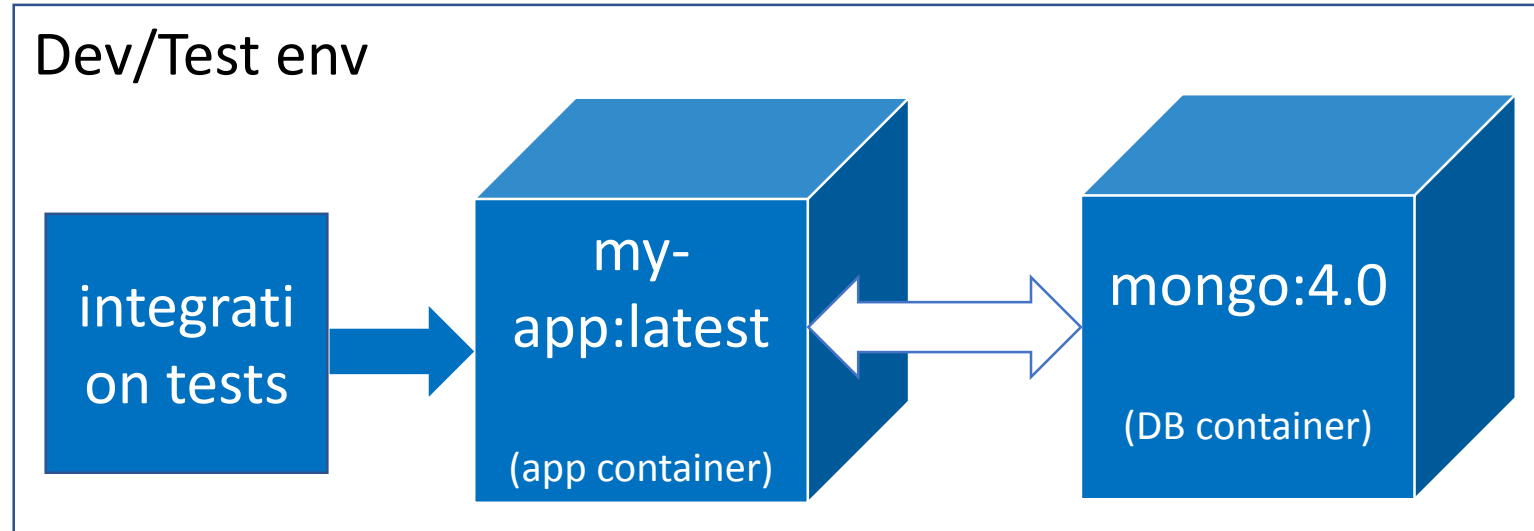
- The time gap
- The personnel gap
- The tools gap

“The twelve-factor developer resists the urge to use different backing services between development and production, even when adapters theoretically abstract away any differences in backing services.”

IT WORKS
ON MY
MACHINE

Testcontainers

- Integration tests that are easy to setup, write, and run
- Test your apps the same way they run in production
- Tests are portable to any compatible implementation:
 - Liberty
 - Wildfly
 - Payara
 - TomEE
 - etc...



MicroShed Testing

- Integration tests that are easy to setup, write, and run
- Test your apps the same way they run in production...in Containers
- Can run multiple containers on same network (e.g. test DB integration)
- <http://microshed.org/microshed-testing/>



```
@MicroShedTest
public class MyTest {

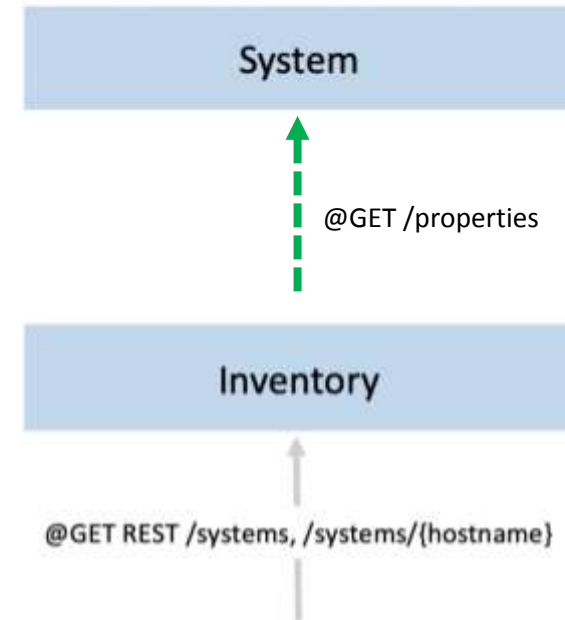
    // Search for Dockerfile.
    // Start app in Container.
    // Wait for Container before running tests.
    @Container
    public static MicroProfileApplication app
        = new MicroProfileApplication()
            .withAppContextRoot("/myservice");

    // Inject JAX-RS REST Client
    @Inject
    public static MyService mySvc;

    // A test method like any other
    @Test
    public void testMyService() {
        ...
    }
}
```

Aims for Module 2:

- Learn how to run the tests in true-to-production environments by using containers with MicroShed Testing.



Time To Code



To get started visit the following URL in your browser:

<https://openliberty.skillsnetwork.site/cloud-native-java-made-easy-microprofile-jakarta-ee>

Application
considerations for cloud
deployments

MicroProfile 4.1 Stack

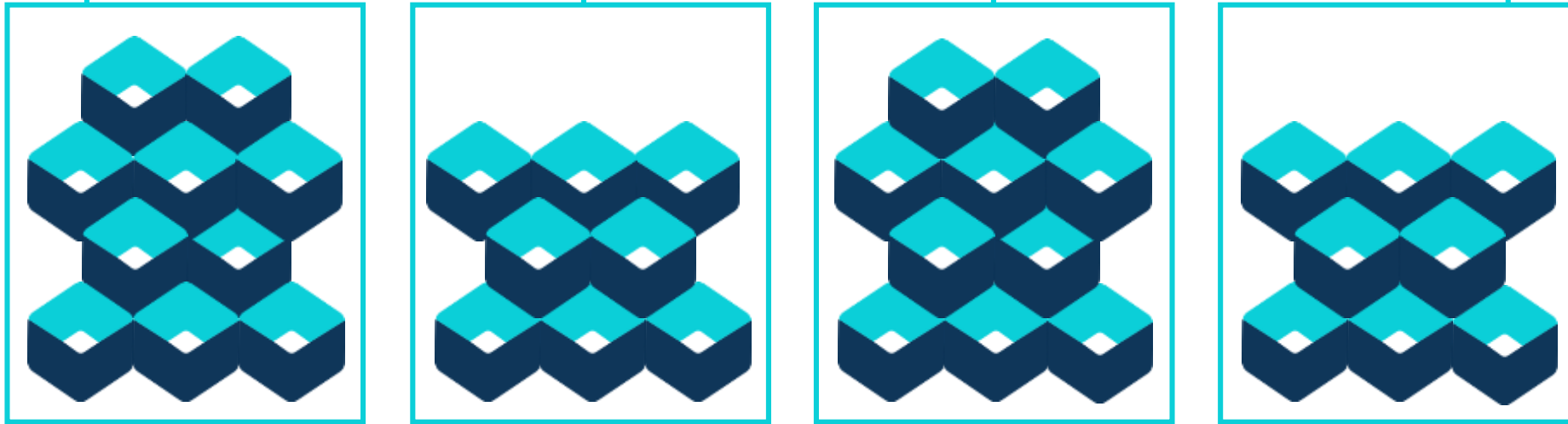


Containers are not enough!



Kubernetes

Regain control with Kubernetes
Organize and govern the container chaos



Horizontal Scaling



Self-Healing



Automated Rollouts & Rollbacks



Secret & Configuration Management



Service Discovery & Load Balancing



Intelligent Scheduling

Lab 3: Health or Configuration

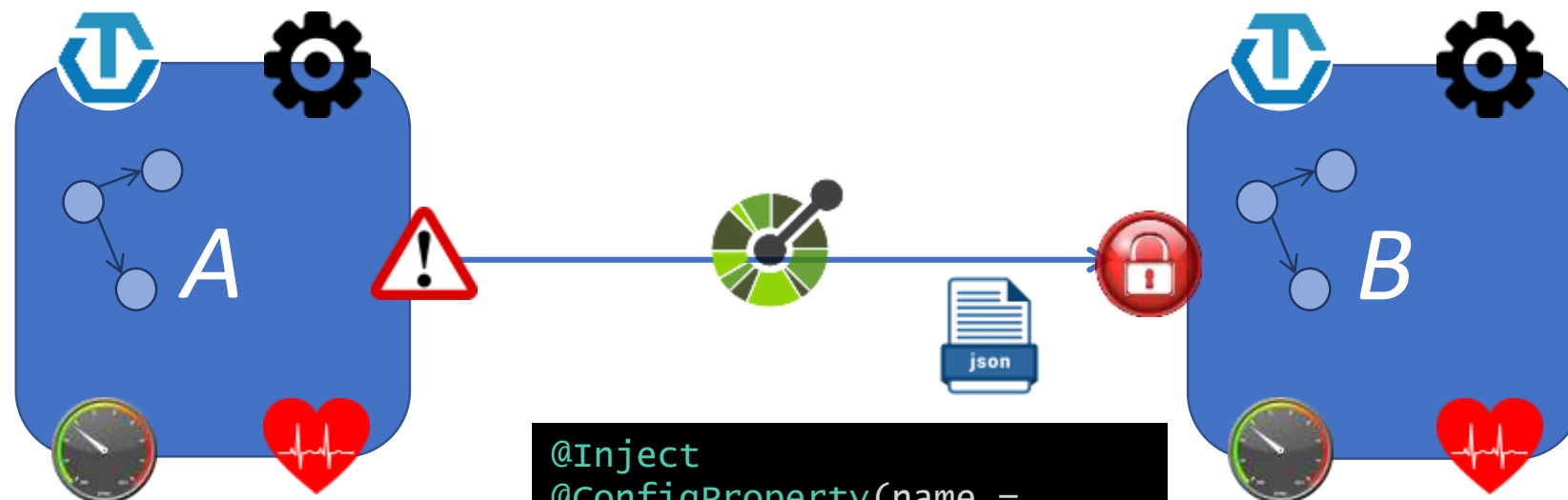


MicroProfile with Kubernetes

Config

```
kubectl create configmap greeting-config --from-literal  
message=Greetings...
```

```
env:  
- name: GREETING  
valueFrom:  
configMapKeyRef:  
name: greeting-  
config  
key: message
```



```
@Inject  
@ConfigProperty(name =  
"GREETING")  
private String greeting;
```



MicroProfile with Kubernetes

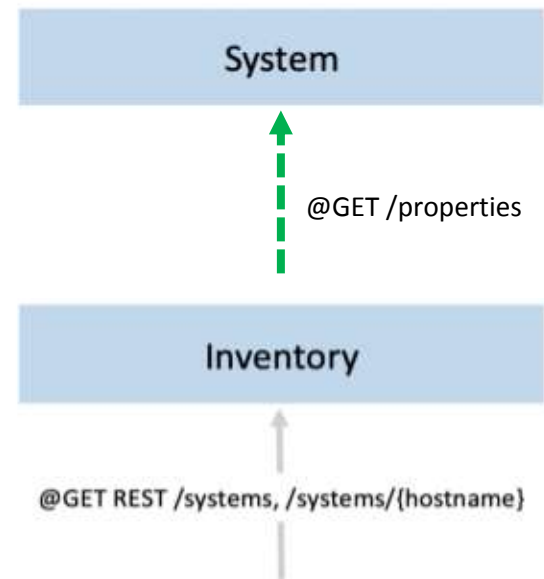
Health

```
readinessProbe:  
  httpGet:  
    path: /health  
    port: 9080  
  initialDelaySeconds:  
    15  
  periodSeconds: 5  
  failureThreshold: 1
```



Aims for Module 3

- Learn how to use MicroProfile Health to report the health status of microservices and take appropriate actions based on this report.
- Learn how to externalize and inject both static and dynamic configuration properties for microservices using MicroProfile Config.



Time To Code



To get started visit the following URL in your browser:

<https://openliberty.skillsnetwork.site/cloud-native-java-made-easy-microprofile-jakarta-ee>

Recap

MicroProfile

- No vendor lock in
- Full set of cloud ready APIs
 - Config
 - Health
 - Metrics
 - Fault Tolerance
 - ...
- Big community

Open Liberty

- Modular Application server
- Light weight
- Easy to configure
- Jakarta EE 8 certified
- Production ready
- Official Docker images available
- Optimized for development

All Open Source!

Jakarta EE

- Open-source Enterprise Java APIs
- No vendor lock-in
- Big community

Open J9

- Low memory footprint
- Fast startup time
- High application throughput
- Smoother ramp-up in the cloud
- Easy to use Docker images



JAKARTA® EE

Links and Materials

All Open Source!

MicroProfile

- MicroProfile Starter: <https://start.microprofile.io/>
- What is MicroProfile? <https://developer.ibm.com/components/open-liberty/series/what-is-microprofile/>
- MicroProfile Homepage: <https://projects.eclipse.org/proposals/eclipse-microprofile>

Open Liberty

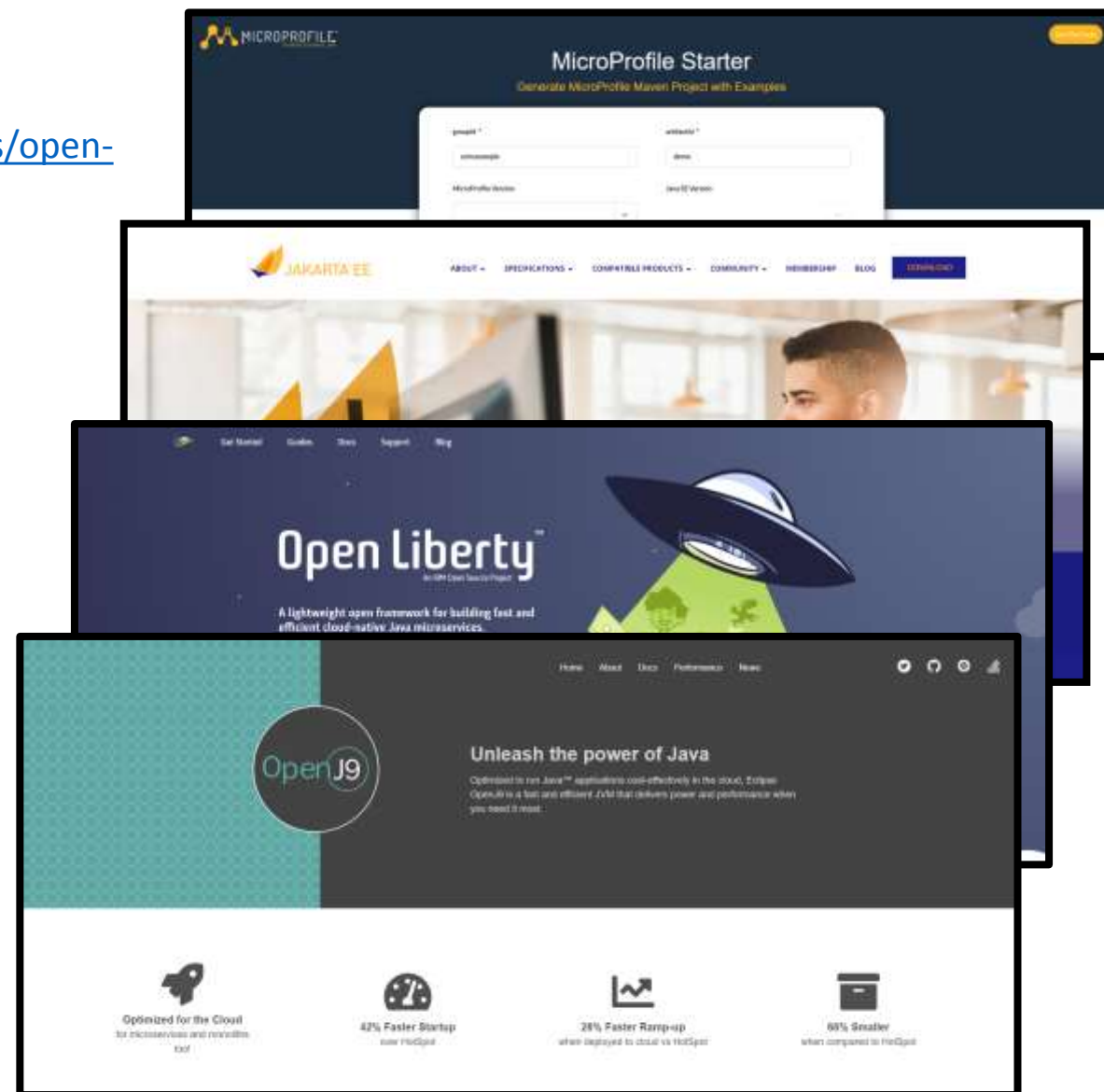
- Open Liberty Site: <https://openliberty.io/>
- Open Liberty Guides: <https://openliberty.io/guides>
- Open Liberty Docs: <https://openliberty.io/docs>

OpenJ9

- OpenJ9 Homepage: <https://www.eclipse.org/openj9/>
- OpenJ9 Docker Downloads Page: <https://hub.docker.com/u/adoptopenjdk>
- OpenJ9 Docs: <https://www.eclipse.org/openj9/docs/>

Jakarta EE

- Jakarta EE Homepage: <https://jakarta.ee/>
- Jakarta EE Blog: <https://jakartablogs.ee/>



Thank you



JAKARTA™ EE

OpenJ9

Jamie Lee Coleman
Software Developer/Advocate
Twitter: @jamie_lee_c
LinkedIn: jamie-coleman

