Building a MicroProfile and Jakarta EE application with Open Liberty

& OpenJ9

JAKARTA EE

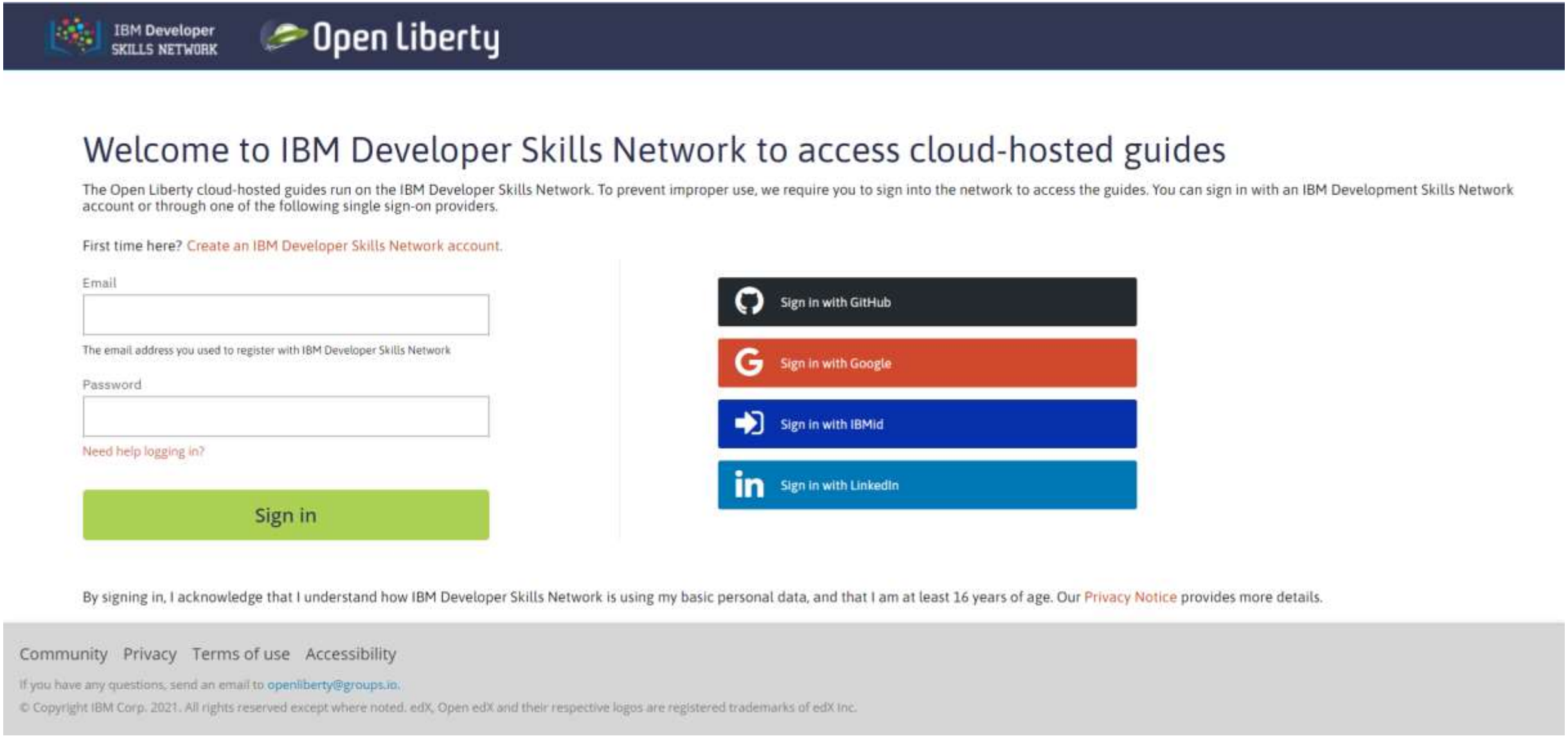
OpenJ9

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

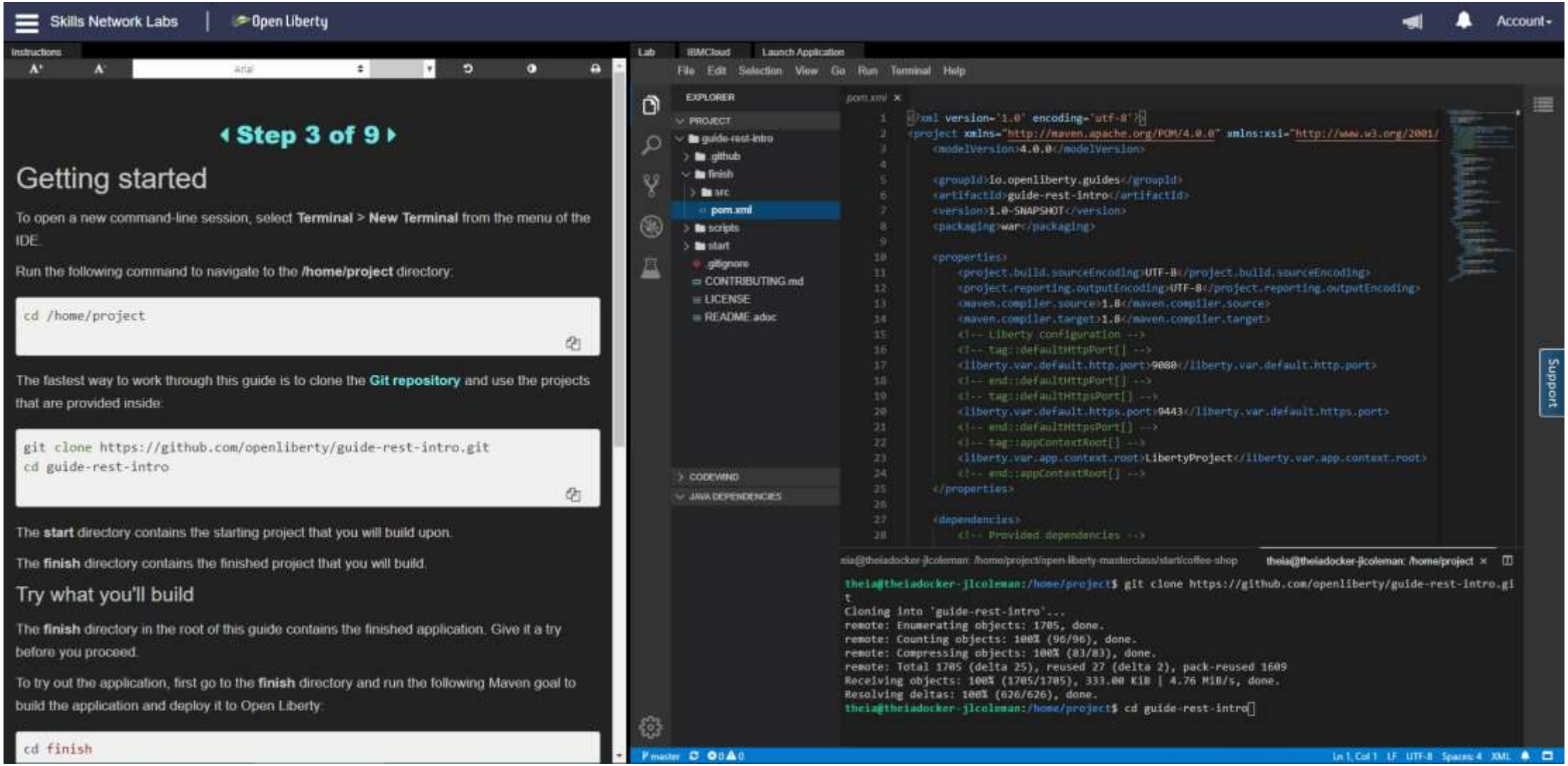
Demo Environment

Skills Network Labs

Skills Network Labs



Skills Network Labs



A Full Open Stack

A Full Open Stack

MicroProfile

Jakarta EE

Open Liberty

OpenJ9

JAKARTA™ EE

OpenJ9

Eclipse MicroProfile

What is MicroProfile?

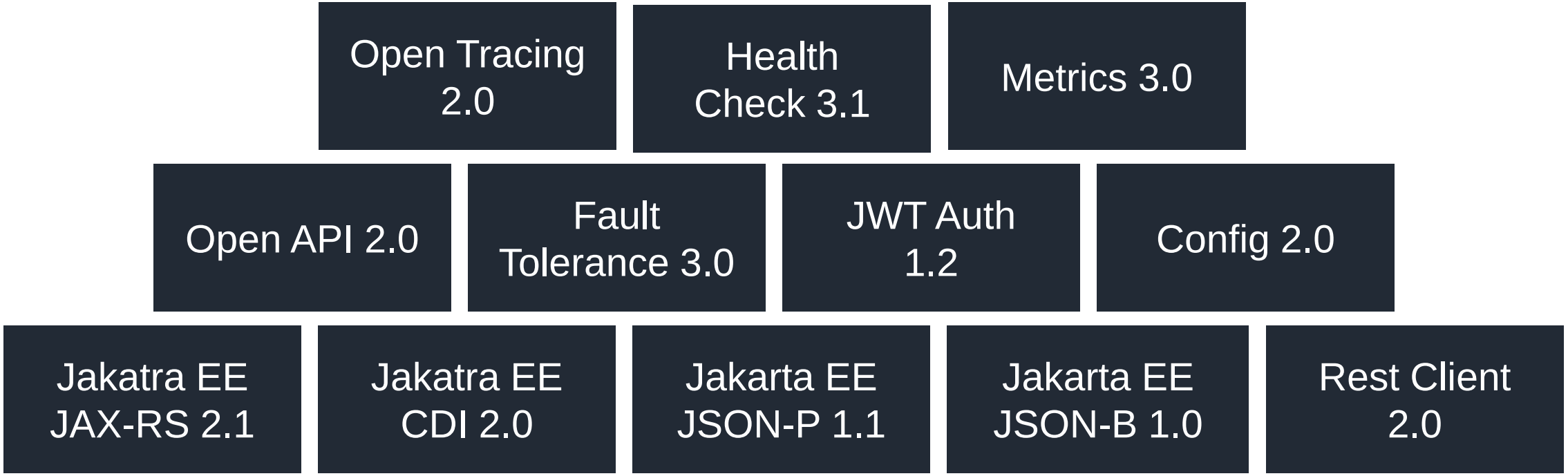
- Eclipse MicroProfile is an open-source community specification for Enterprise Java microservices
- A community of individuals, organizations, and vendors collaborating within an open source (Eclipse) project to bring microservices to the Enterprise Java community

microprofile.io

MicroProfile Contributors

MicroProfile Vendors/Implementations

MicroProfile 4.1 Stack



https://developer.ibm.com/components/open-liberty/series/what-is-microprofile/

MicroProfile 4.1 Stack

LRA 1.0

Reactive Streams Operators 2.0

GraphQL 1.1

Reactive Messaging 2.0

Standalone Projects

Context Propagation 1.2

Open Tracing 2.0

Health Check 3.1

Metrics 3.0

Open API 2.0

Fault Tolerance 3.0

JWT Auth 1.2

Config 2.0

Jakatra EE JAX-RS 2.1

Jakatra EE CDI 2.0

Jakarta EE JSON-P 1.1

Jakarta EE JSON-B 1.0

Rest Client 2.0

https://developer.ibm.com/components/open-liberty/series/what-is-microprofile/

Open Liberty Overview

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)

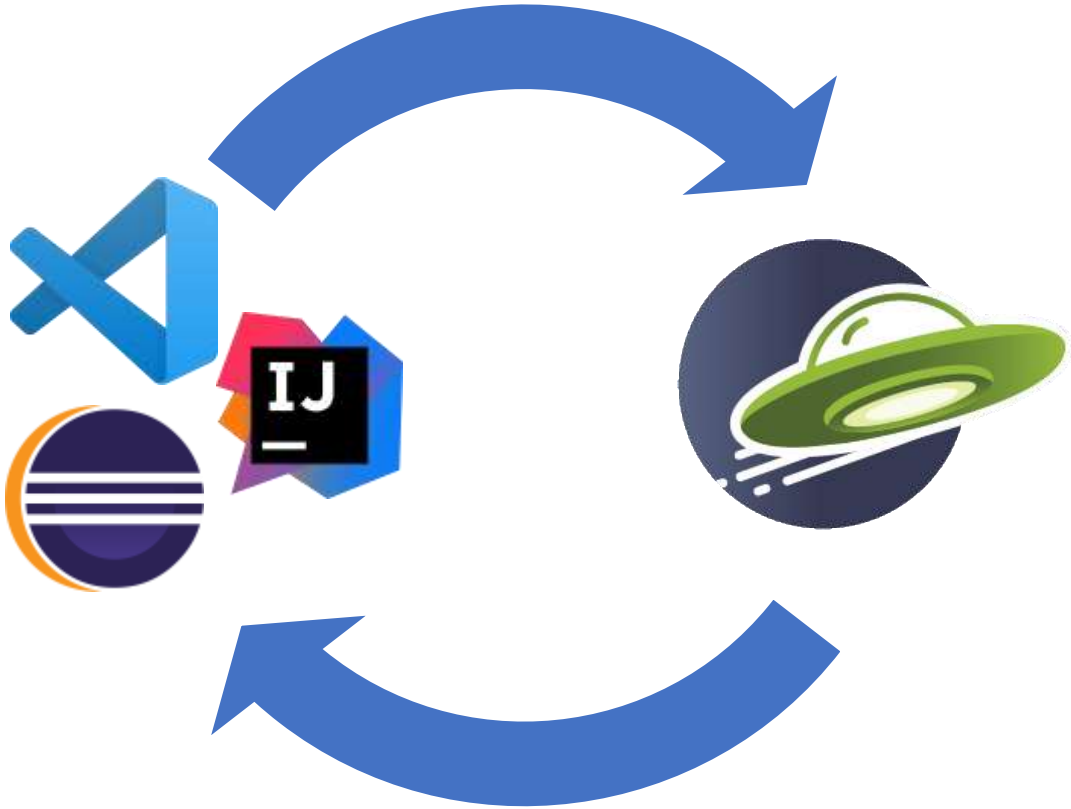
Ready for containers

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



https://developer.ibm.com/tutorials/achieve-rapid-iterative-cloud-native-application-development/

Developer productivity

IDEs


Dev Mode

Repositories

The Central Repository

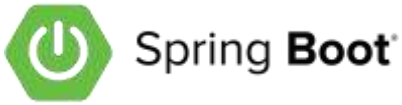
dockerhub

Build

Maven™

Gradle

APIs



Testing

Jesse Gallagher
@Gidgerby

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

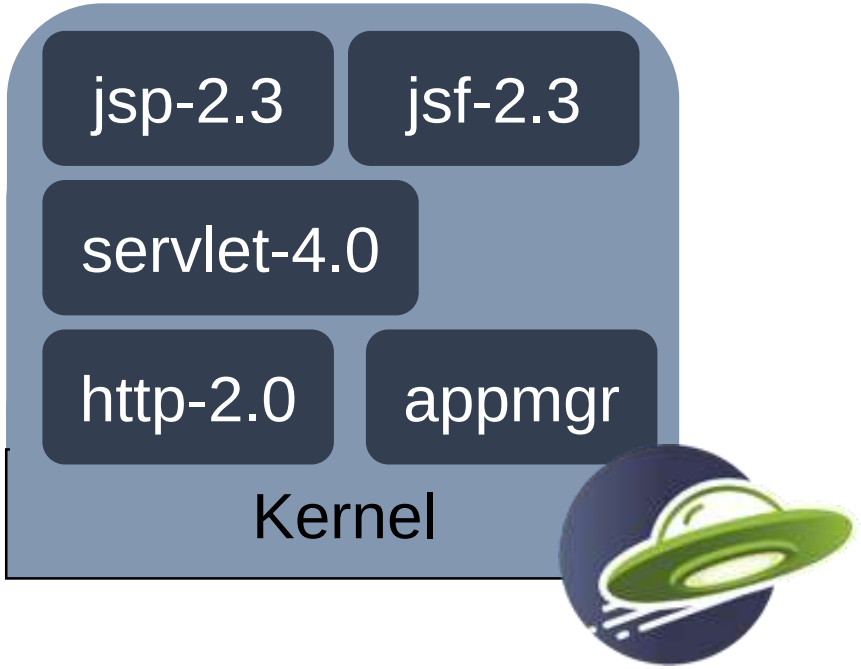
Tim Zöller
@javahippie

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


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



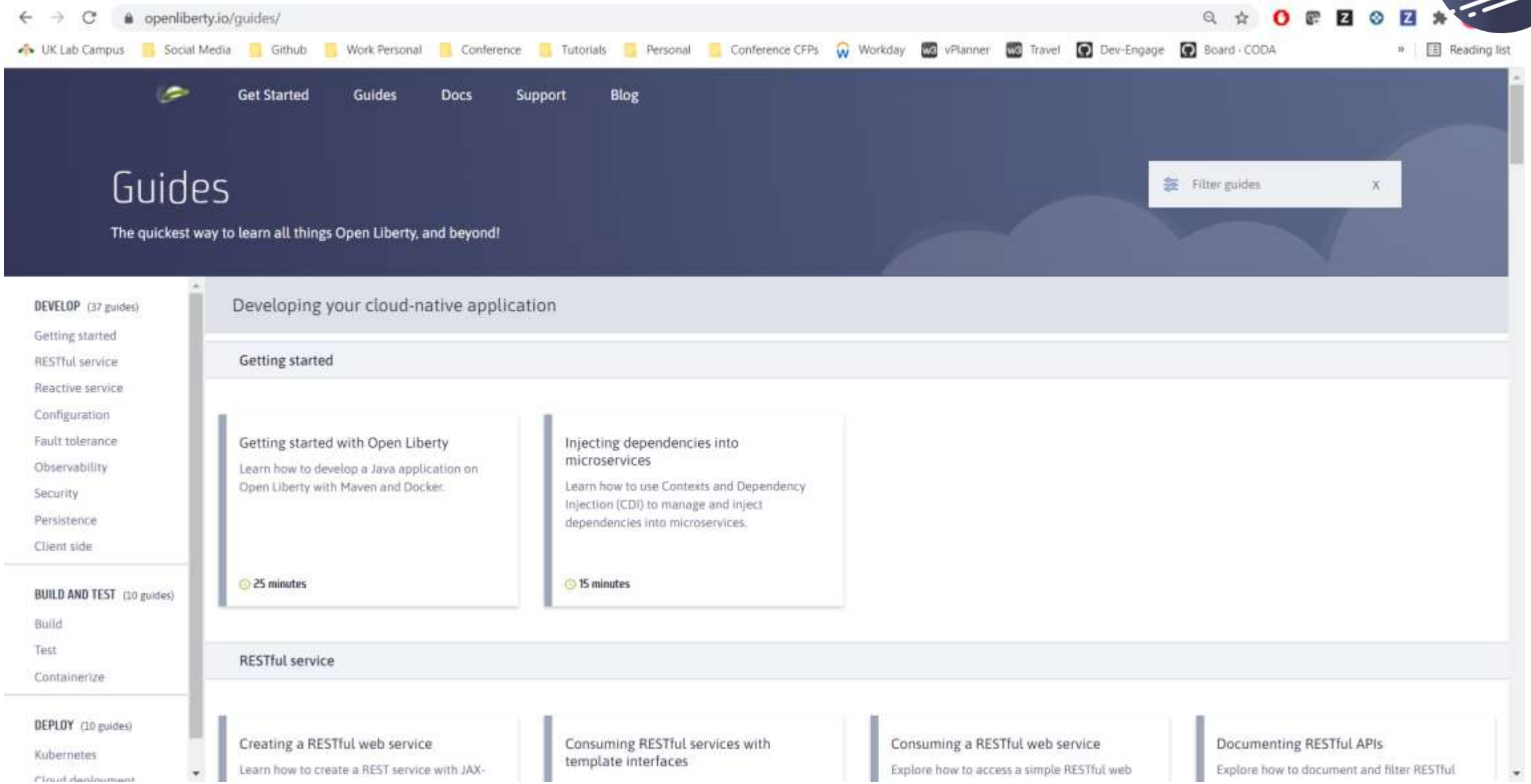
API Support

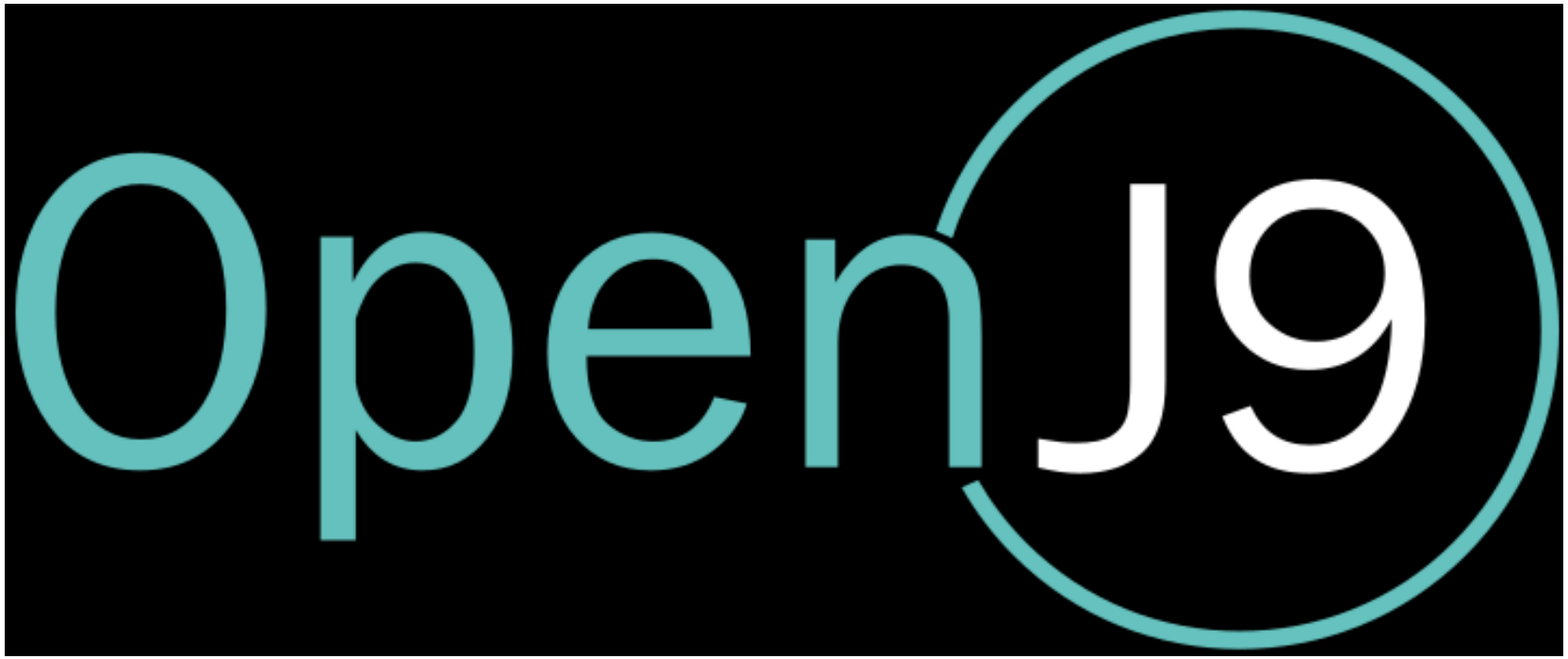
- First shipped in WAS 8.5 in 2012
  - Servlet + JSP + JPA
- Web Profile 6 in 2014
- Java EE 7 in 2016 – first commercial product to certify
- Java EE 8 in 2018 – first to certify
- Jakarta EE 8 in 2019 – first to certify
- Eclipse MicroProfile – first to deliver 1.0-1.4, 2.0-2.1, 3.0

Liberty Zero Migration

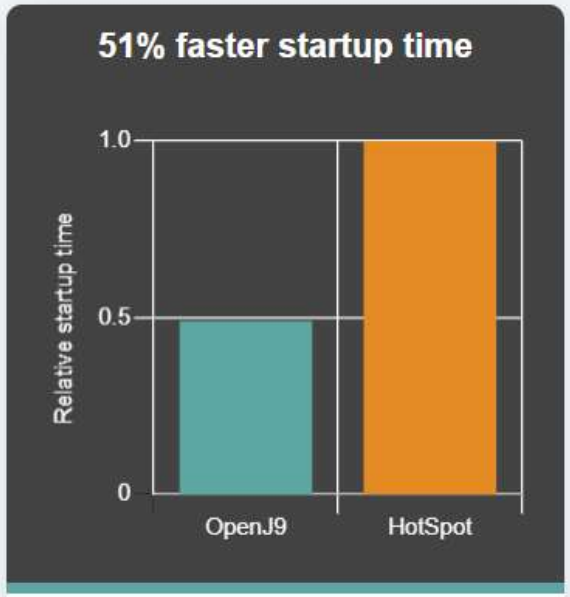
- Zero config migration
  - Write once, run forever
- Zero app migration
  - No behavior changes in existing features
  - New behaviors in new features
- Choose your Java
  - Java 14, 11, 8
  - AdoptOpenJDK
  - IBM
  - OpenJDK
  - Oracle

Developer-Oriented Docs & Guides

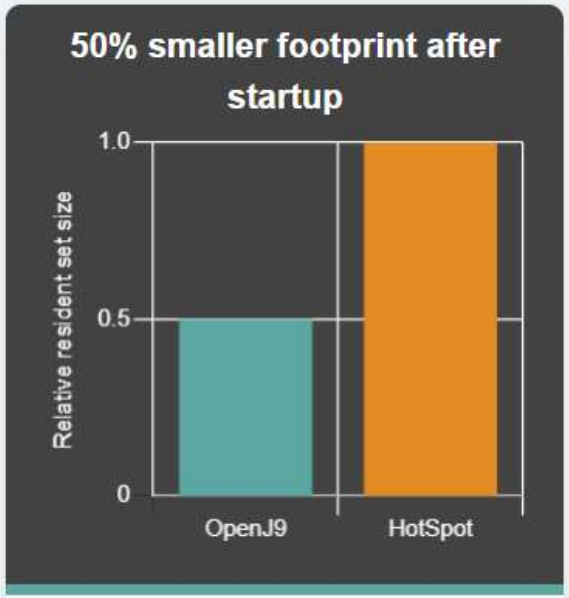




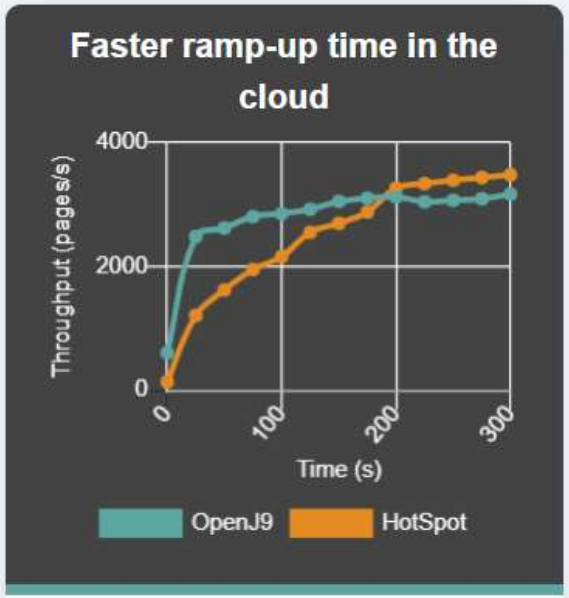
Overview of OpenJ9



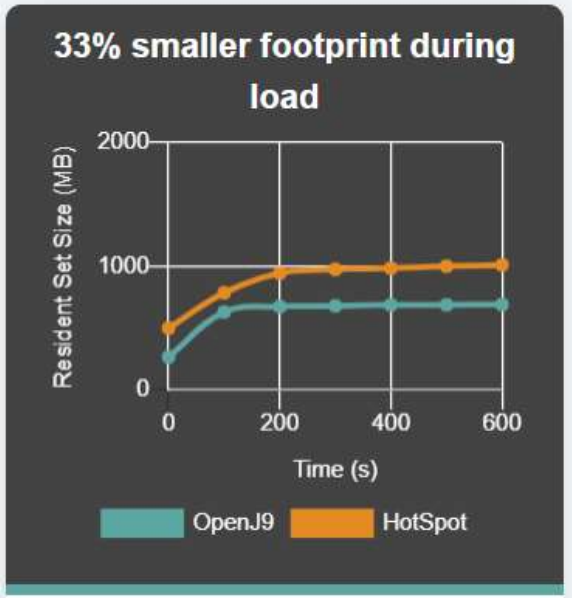
By using shared classes cache and AOT technology, OpenJ9 starts in roughly half the time it takes HotSpot.



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

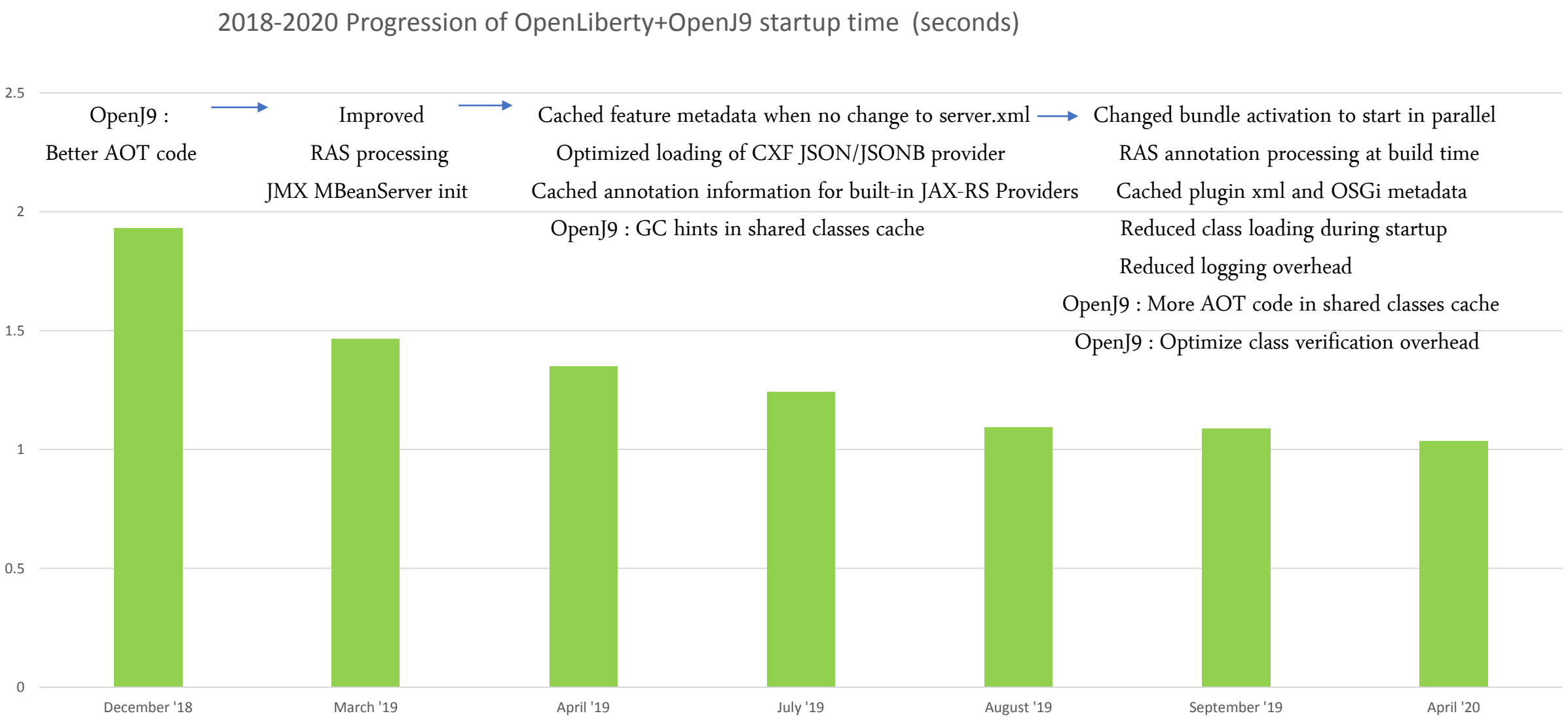


OpenJ9 reaches peak throughput much faster than HotSpot making it especially suitable for running short-lived applications.

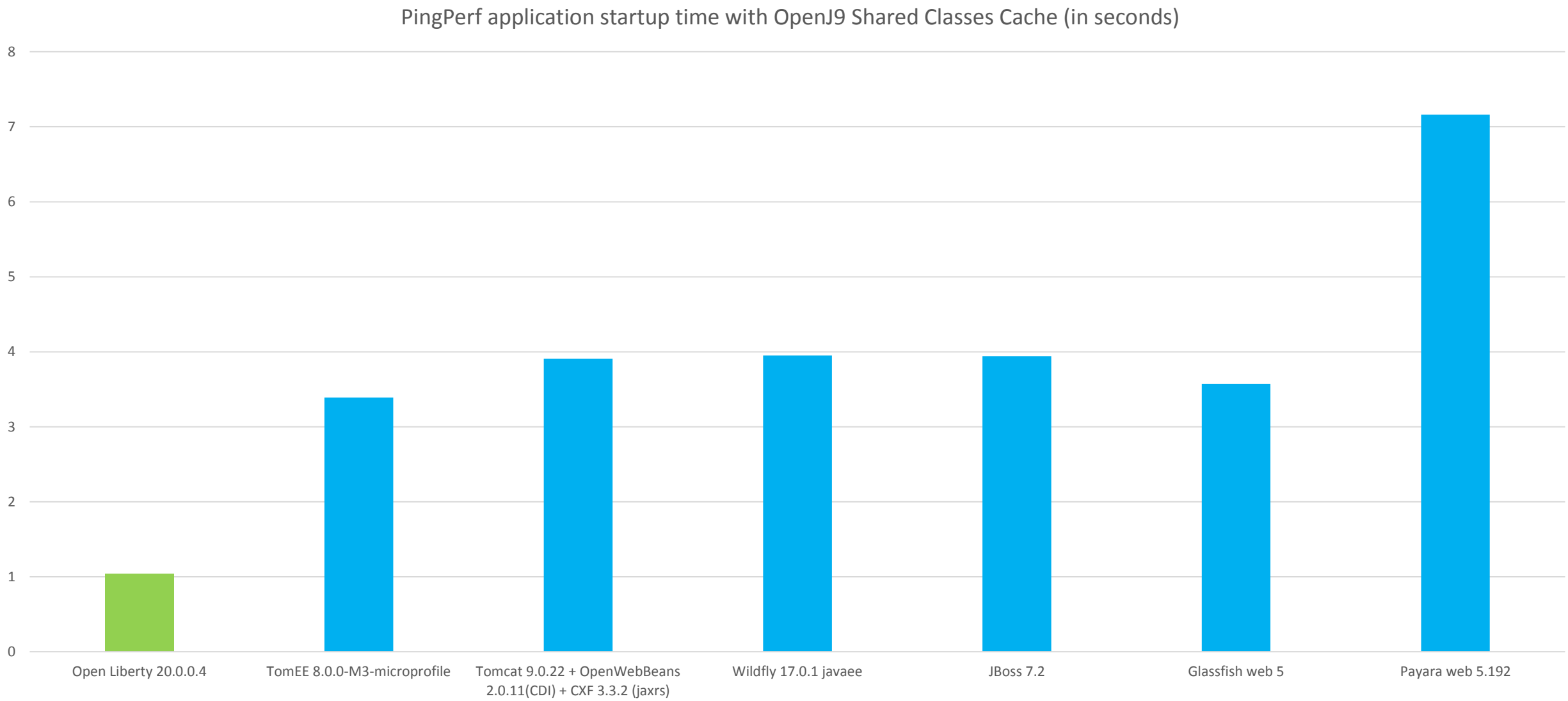


Consistent with the footprint results after startup, the OpenJ9 footprint remains much smaller than HotSpot when load is applied.

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



Open Liberty startup time comparison (using OpenJ9 JVM)



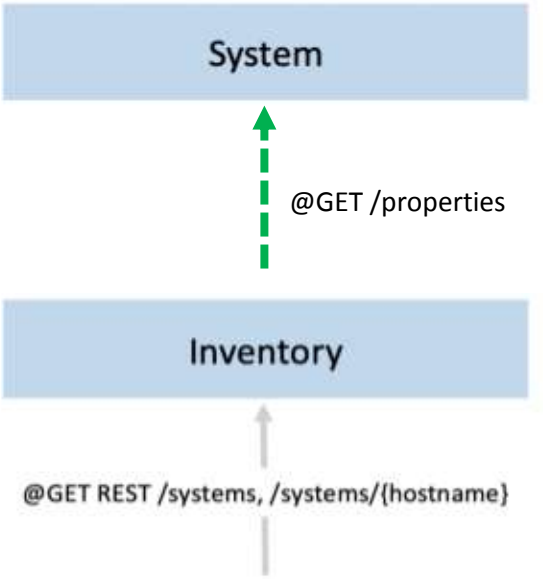
Lab 1

RESTful Services

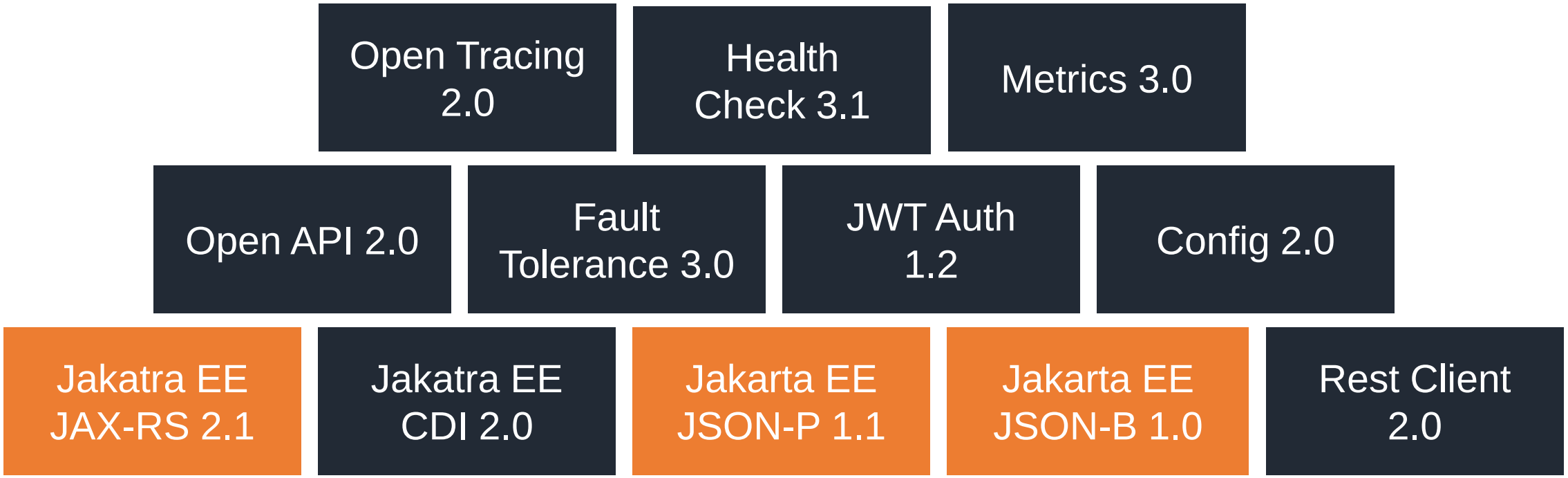
Lab's Application

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



MicroProfile Core Technologies

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

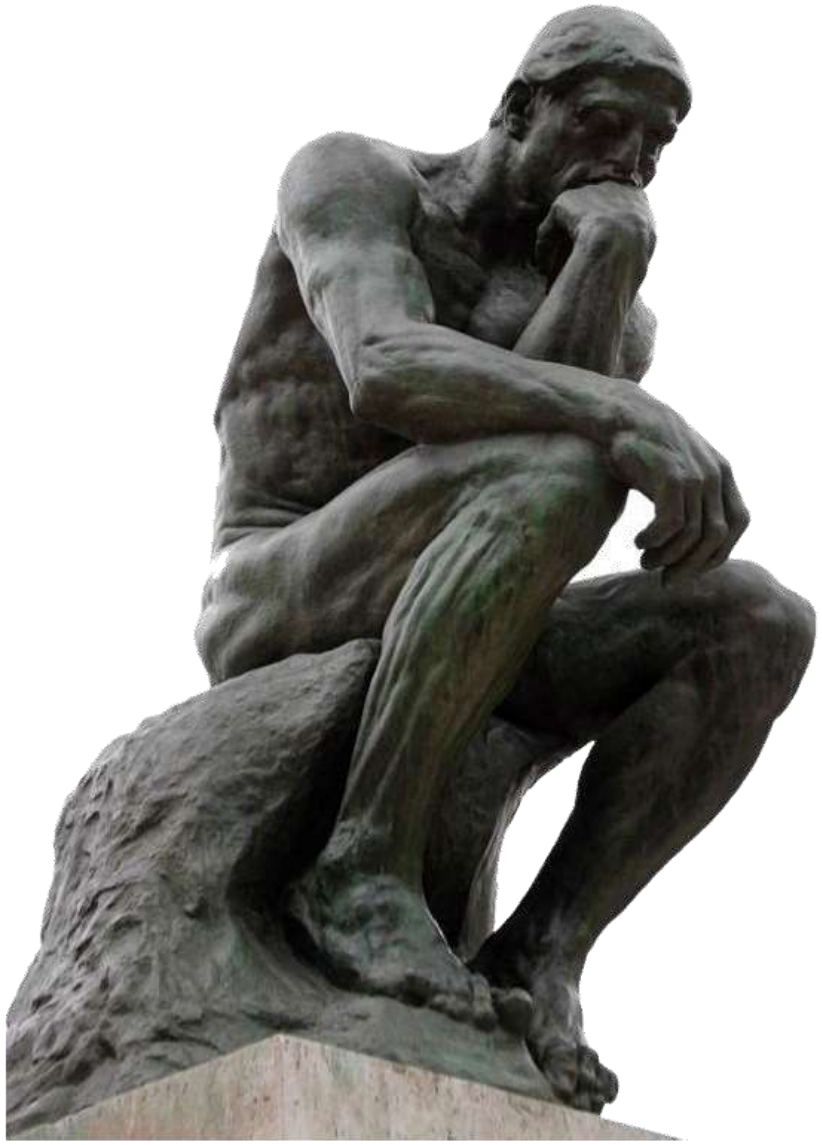
To get started visit the following URL in your browser:

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

Lab 2

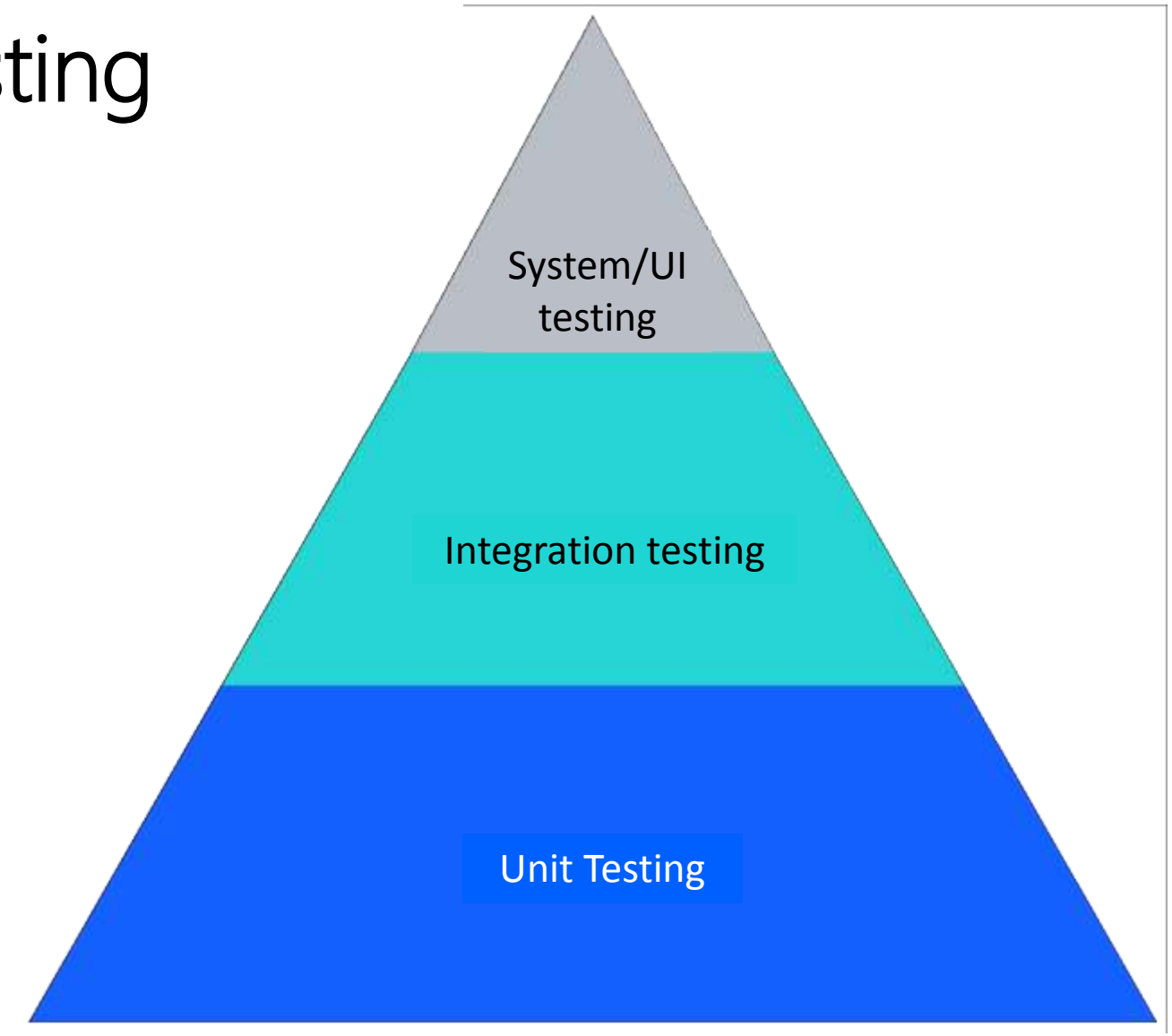
True to Production Testing

Why do we write automated tests?



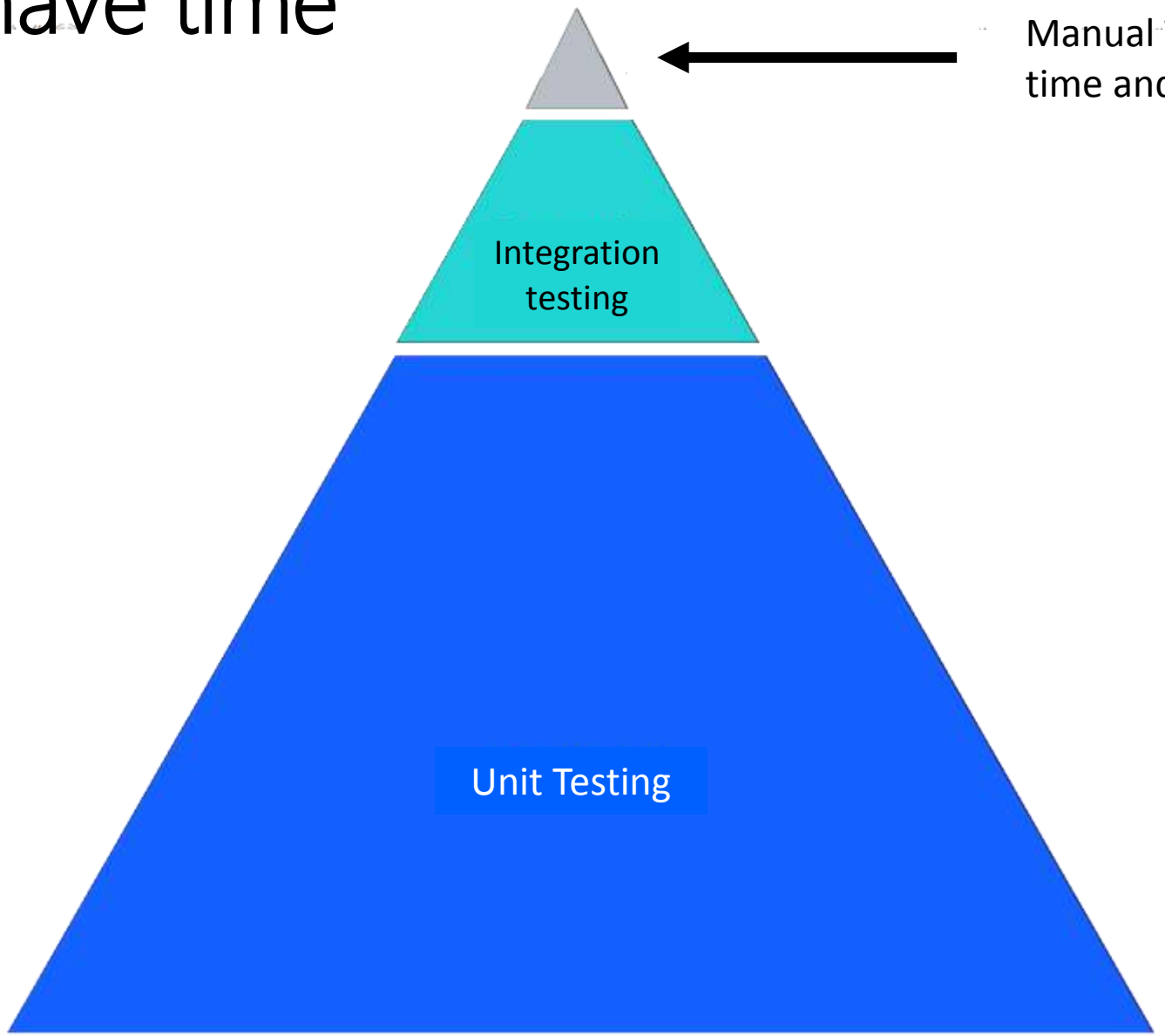
To have confidence that our applications work the way we want them to!

Types of testing



https://developer.ibm.com/series/testing-cloud-native-java-applications/

What we have time to do



https://developer.ibm.com/series/testing-cloud-native-java-applications/

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:

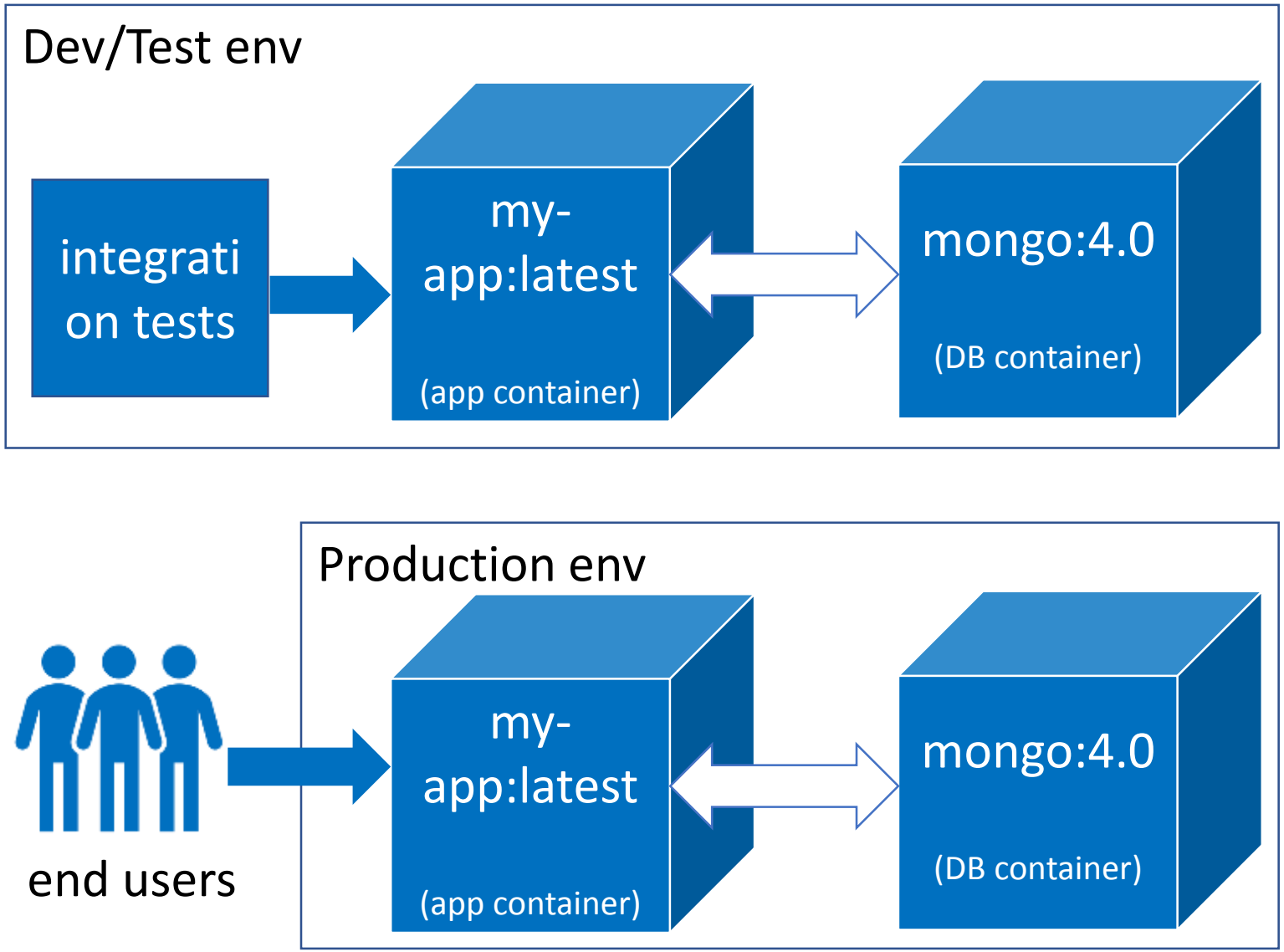
- The time gap
- The personnel gap
- The tools gap

“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

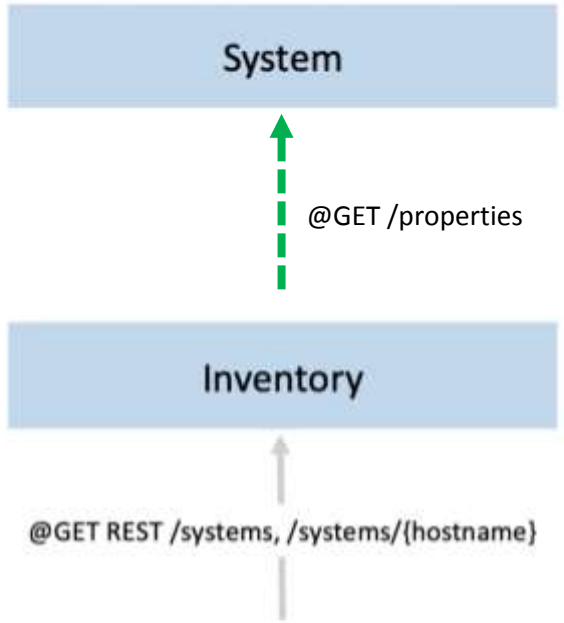
- Integration tests that are easy to setup, write, and run
- Test your apps the same way they run in production...in Containers
- Can run multiple containers on same network (e.g. test DB integration)
- http://microshed.org/microshed-testing/

```
@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() {
        ...
    }
}
```

microshed testing

Aims for Module 2:

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



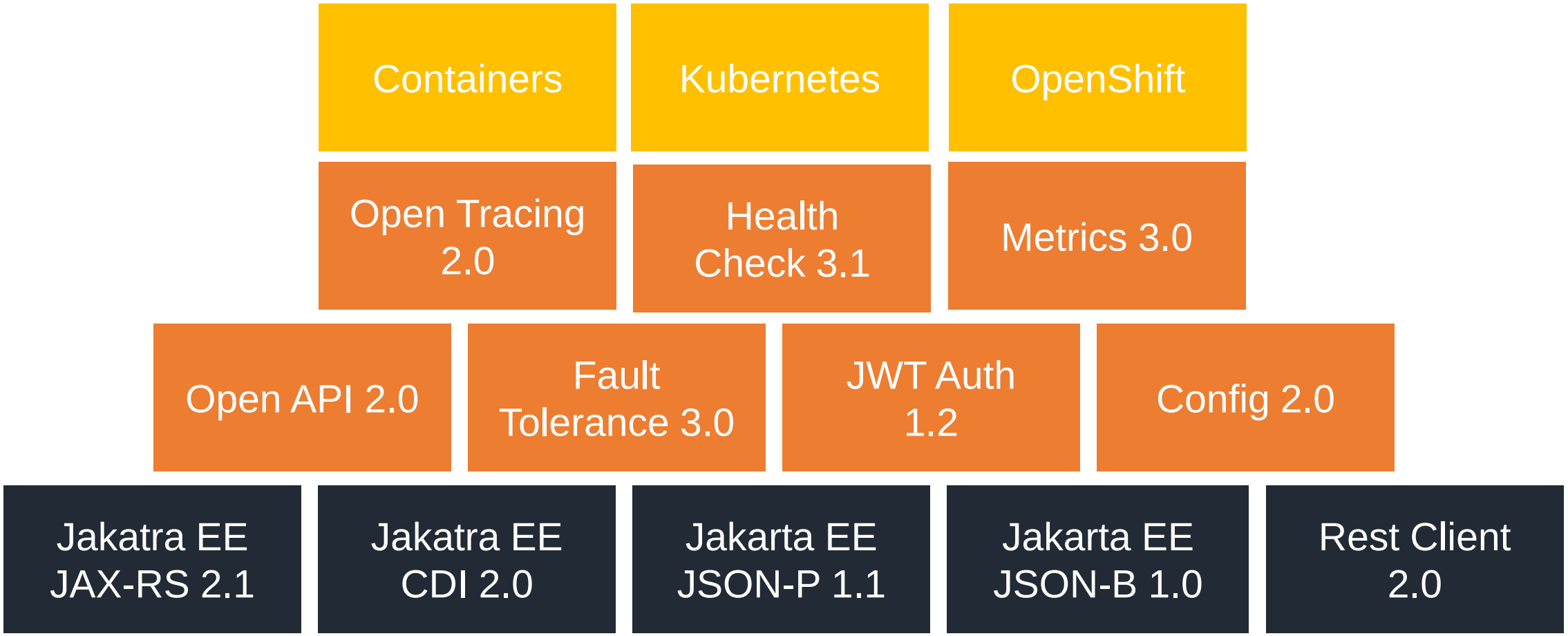
Time To Code

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

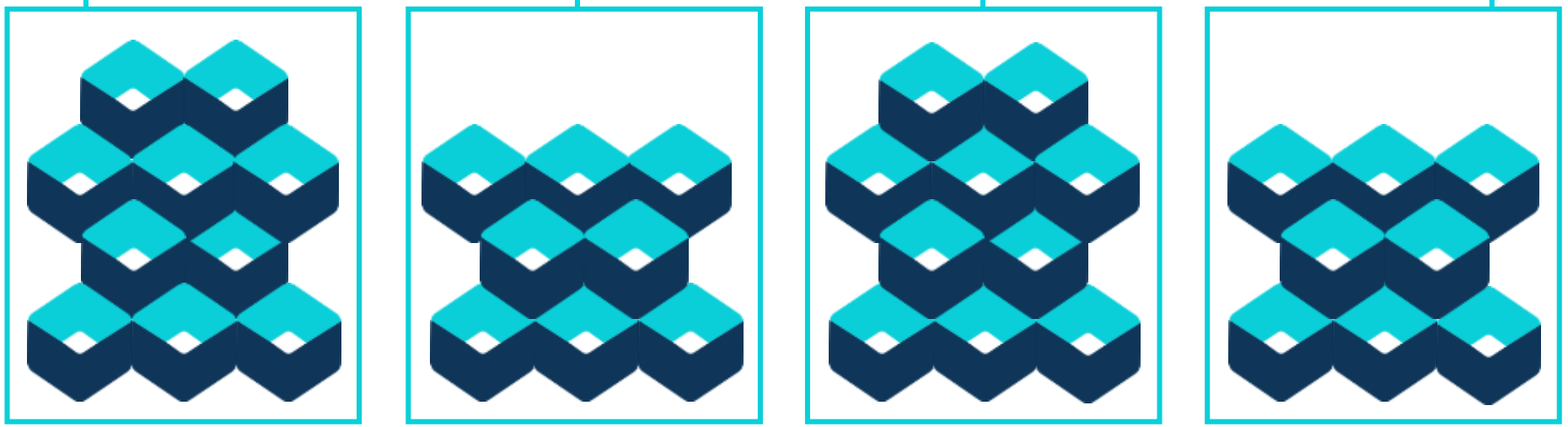
MicroProfile 4.1 Stack



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

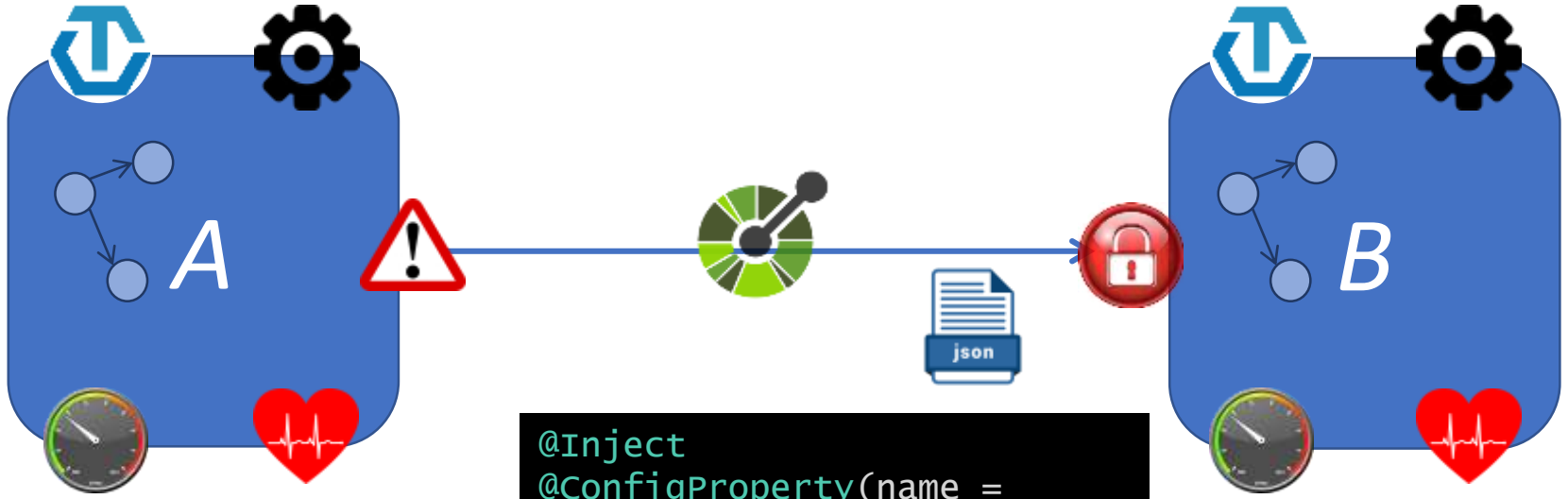
Intelligent Scheduling

Lab 3: Health or Configuration

Config

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

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



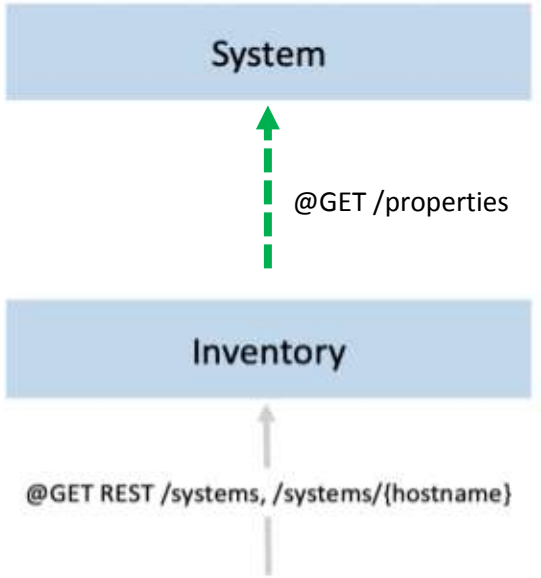

```
@Inject
@ConfigProperty(name =
"GREETING")
private String greeting;
```

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

All Open Source!

MicroProfile

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

Open Liberty

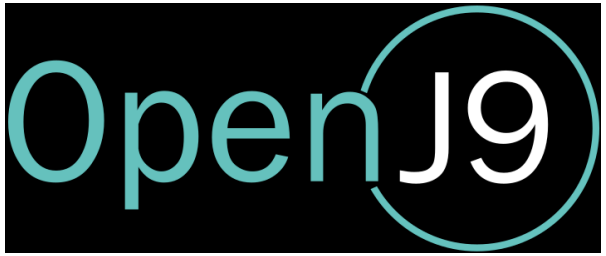
- Modular Application server
- Light weight
- Easy to configure
- Jakarta EE 8 certified
- Production ready
- Official Docker images available
- Optimized for development

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



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

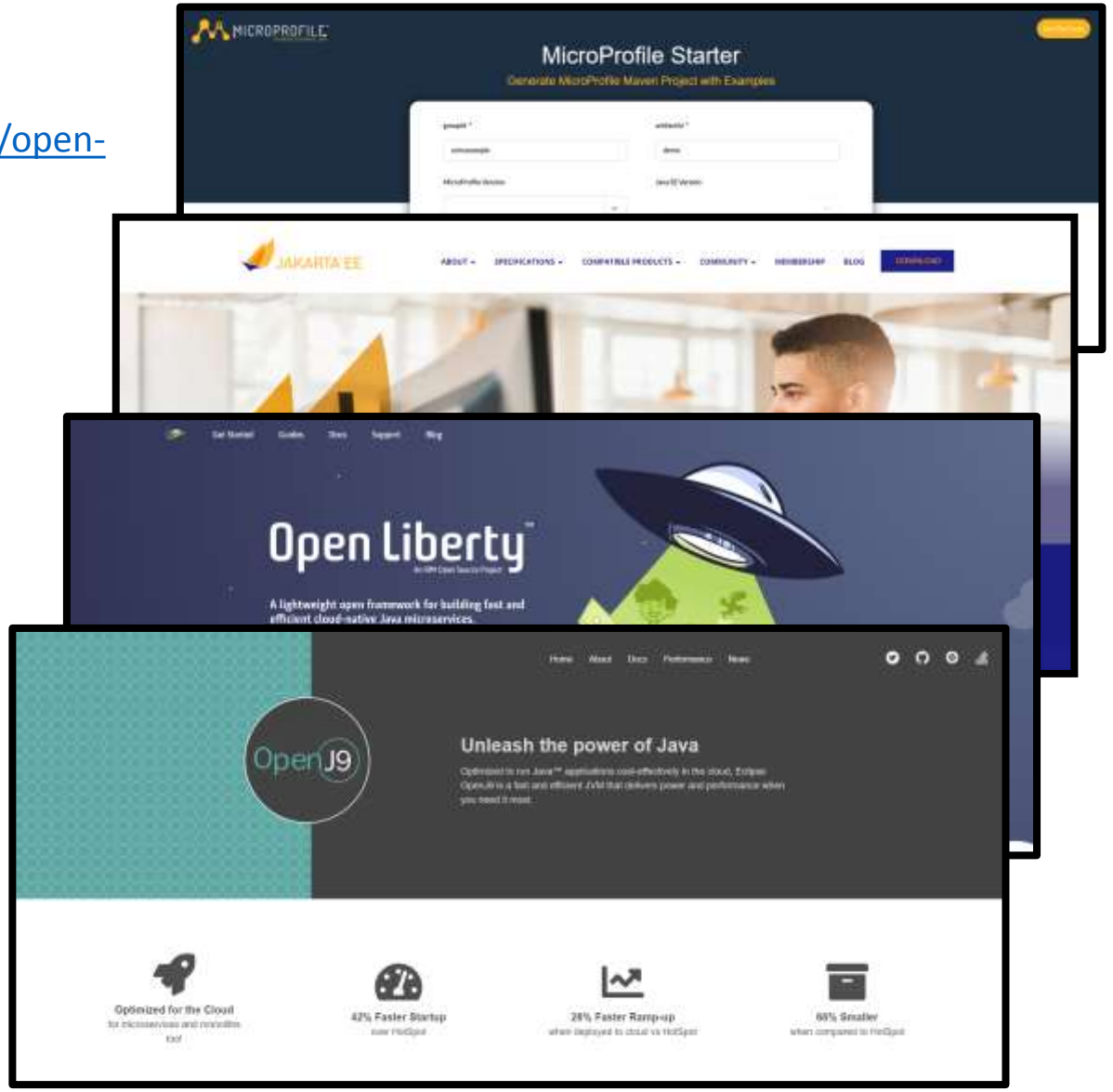
- Open Liberty Site: https://openliberty.io/
- Open Liberty Guides: https://openliberty.io/guides
- Open Liberty Docs: https://openliberty.io/docs

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

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