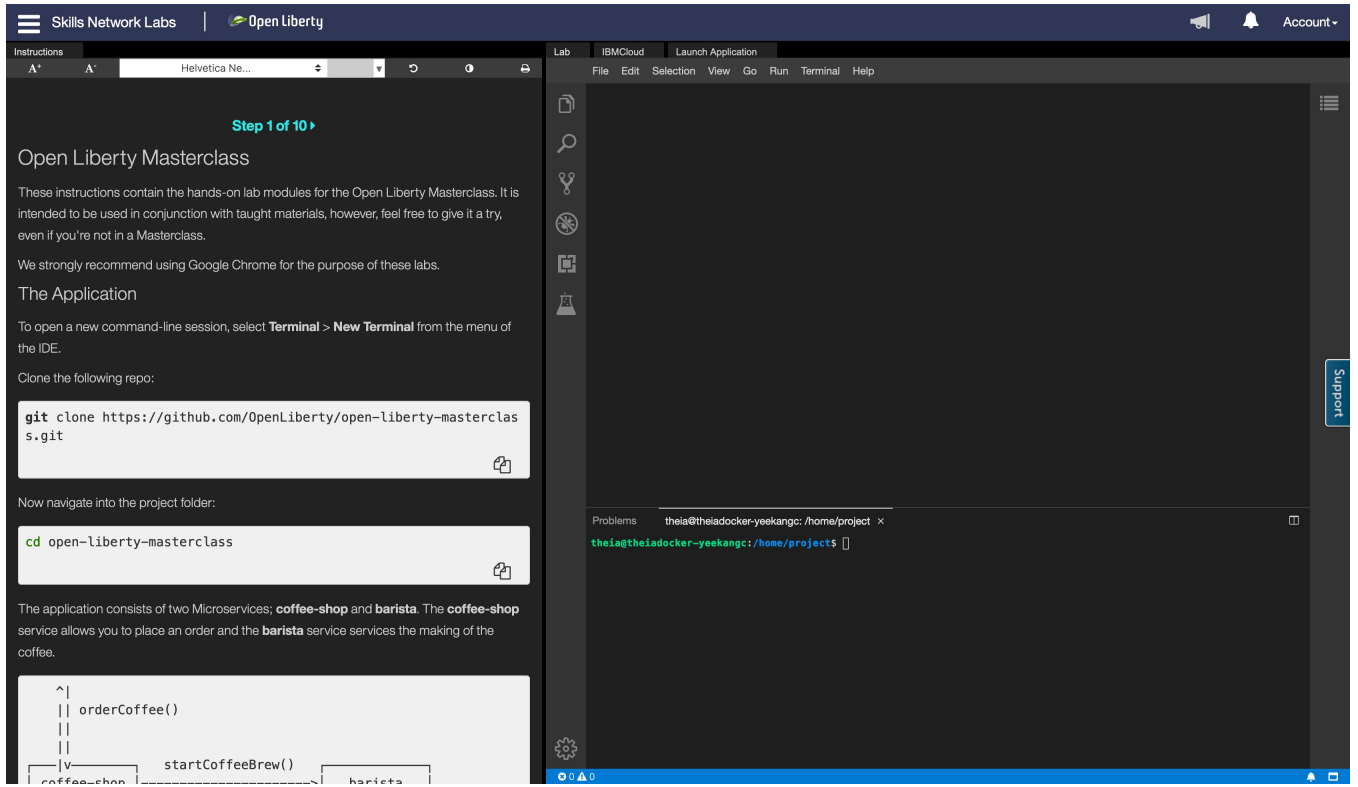
Open Liberty

Masterclass: A Technical Deep Dive on Liberty

Ready to get your hands dirty?

Online exercises:
https://ibm.biz/Liberty Masterclass

Hosted environment with an editor and a terminal that you can try online through your browser



Ready to get your hands dirty?

The Masterclass is *open sourced* at:
https://github.com/OpenLiberty/open-liberty-masterclass

Follow the instructions in the ”Before you begin” section of the README.md to:

- Install Pre-requisites
- Prime Maven and Docker Caches

Mastersclass hands-on content

Background

Open Liberty

6 reasons why Liberty

Lightweight, highly-efficient runtime

CI/CD optimized operational experience

Simple true-to-production developer experience

Just enough runtime	80% disk and 56% memory saving
Low operating cost	4x increased density over Tomcat & Spring Boot
Continuous delivery	Zero-effort security fixing & zero technical debt
Zero migration	100% v2v & fixpack migration saving
Kubernetes optimized	Self-tuned optimal perf, production-ready, kube-native
Developer experience	Container & kube-native experience, rapid inner loop

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
- Jakarta EE 9 – first vendor product to certify


Docs
Get Started
Support
Fork the code
openliberty.io
Launched September 2017
Open Liberty
An IBM Open Source Project
A lightweight open framework for building fast and efficient cloud-native Java microservices.
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.
Get Open Liberty

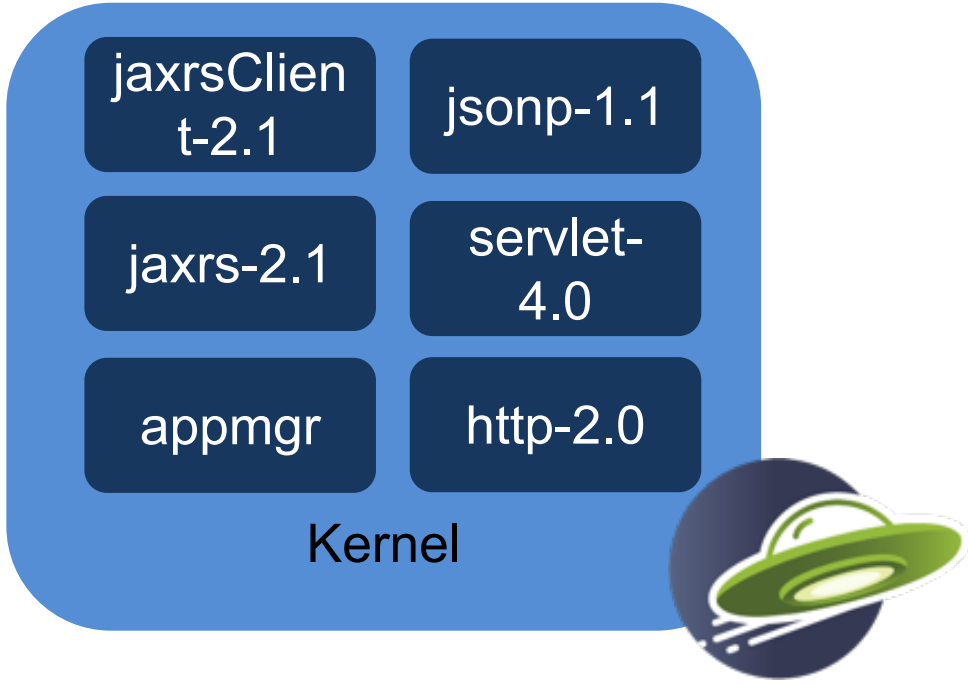
Liberty Architecture

Just Enough Application Server

Open Liberty

- You control which features are loaded into each server instance


```
<feature>jaxrs-2.1</feature>
```



Liberty Zero Migration

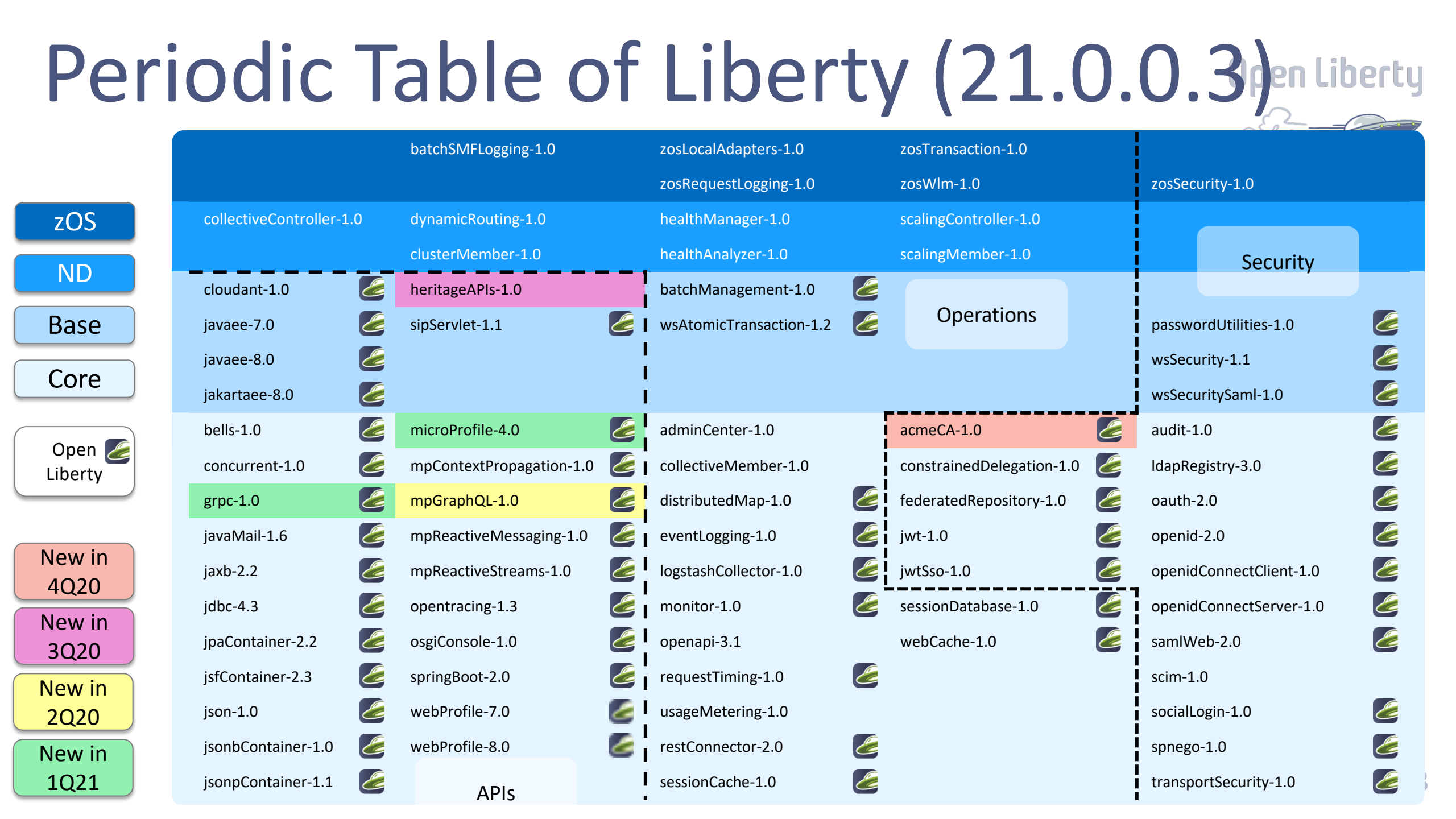
Open Liberty

- 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 15, 11, 8
  - AdoptOpenJDK
  - IBM
  - OpenJDK
  - Oracle

Features

Open Liberty

Periodic Table of Liberty (21.0.0.3)



Build

Module 1

Support for Maven and Gradle

- Manage full server and application lifecycle through the two most popular build technologies.
- Maven
  - liberty-maven-plugin
  - liberty-archetype-*
  - Docs & Source: https://github.com/OpenLiberty/ci.maven
- Gradle
  - liberty-gradle-plugin
  - Docs & Source: https://github.com/OpenLiberty/ci.gradle
- All the artefacts you need in Maven Central
  - https://search.maven.org/search?q=io.openliberty
  - https://search.maven.org/search?q=com.ibm.websphere

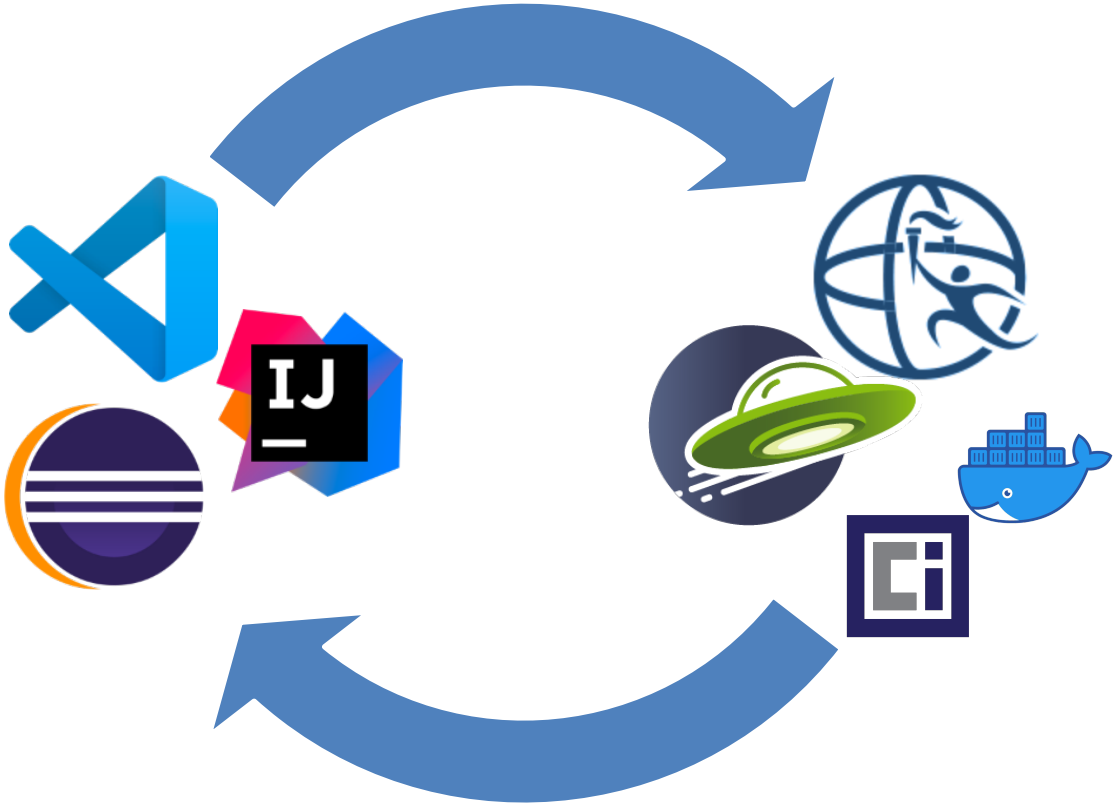
Dev Mode

Module 2

dev mode

Open Liberty

- Boosts developer productivity
- Immediate feedback for code and config changes
- No re-build necessary
- Hot deployment, testing and debugging
- In containers for near-production testing



mvn liberty:dev

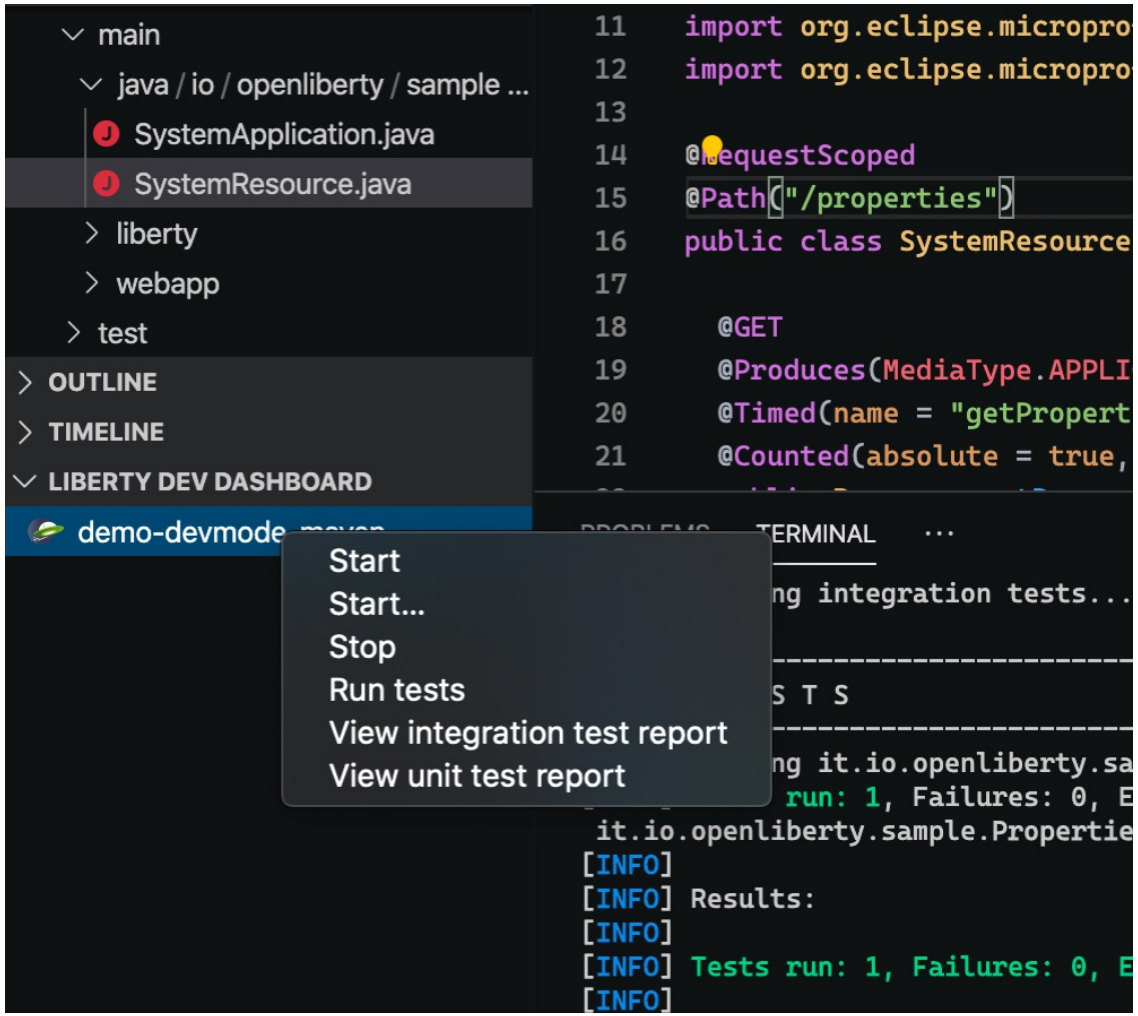
mvn liberty:devc

gradle libertyDev

gradle libertyDevc

Tools

- Open Liberty Tools for VS Code, IntelliJ & Eclipse
  - Start/Stop server
  - Run tests
  - View test reports
- MicroProfile Starter
  - Create a new custom Microservice project within the IDE
- MicroProfile REST Client Generator
  - Generate a REST client based on an OpenAPI definition



Developer experience

Open Liberty

IDEs

Dev Mode

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!

Application APIs

Module 2

Java EE 7

appClientSupport-1.0
batch-1.0
concurrent-1.0
ejb-3.2
ejbHome-3.2
ejbPersistentTimer-3.2
ejbRemote-3.2
j2eeManagement-1.1
javaMail-1.5
jacc-1.5
jaspic-1.1
jaxb-2.2
jaxws-2.2
jca-1.7
jms-2.0
managedBeans-1.0
mdb-3.2
wasJmsClient-2.0
webProfile-7.0
wmqJmsClient-2.0

appSecurity-2.0
beanValidation-1.1
cdi-1.2
ejbLite-3.2
el-3.0
jaxrs-2.0
jaxrsClient-2.0
jdbc-4.2
jndi-1.0
jpa-2.1
jsonp-1.0
jsf-2.2
jsp-2.3
managedBeans-1.0
servlet-3.1
ssl-1.0
websocket-1.1

Jakarta EE or Java EE 8

appClientSupport-1.0	ejbHome-3.2	jacc-1.5	managedBeans-1.0
appSecurityClient-1.0	ejbPersistentTimer-3.2	jaxb-2.2	mdb-3.2
batch-1.0	ejbRemote-3.2	jaxws-2.2	wasJmsClient-2.0
concurrent-1.0	j2eeManagement-1.1	jca-1.7	webProfile-8.0
ejb-3.2	javaMail-1.6	jms-2.0	wmqJmsClient-2.0

appSecurity-3.0	jaxrs-2.1	jsonp-1.1	websocket-1.1
beanValidation-2.0	jaxrsClient-2.1	jsf-2.3	
cdi-2.0	jdbc-4.2	jsp-2.3	
ejbLite-3.2	jndi-1.0	managedBeans-1.0	
el-3.0	jpa-2.2	servlet-4.0	
jaspic-1.1	jsonb-1.0	ssl-1.0	

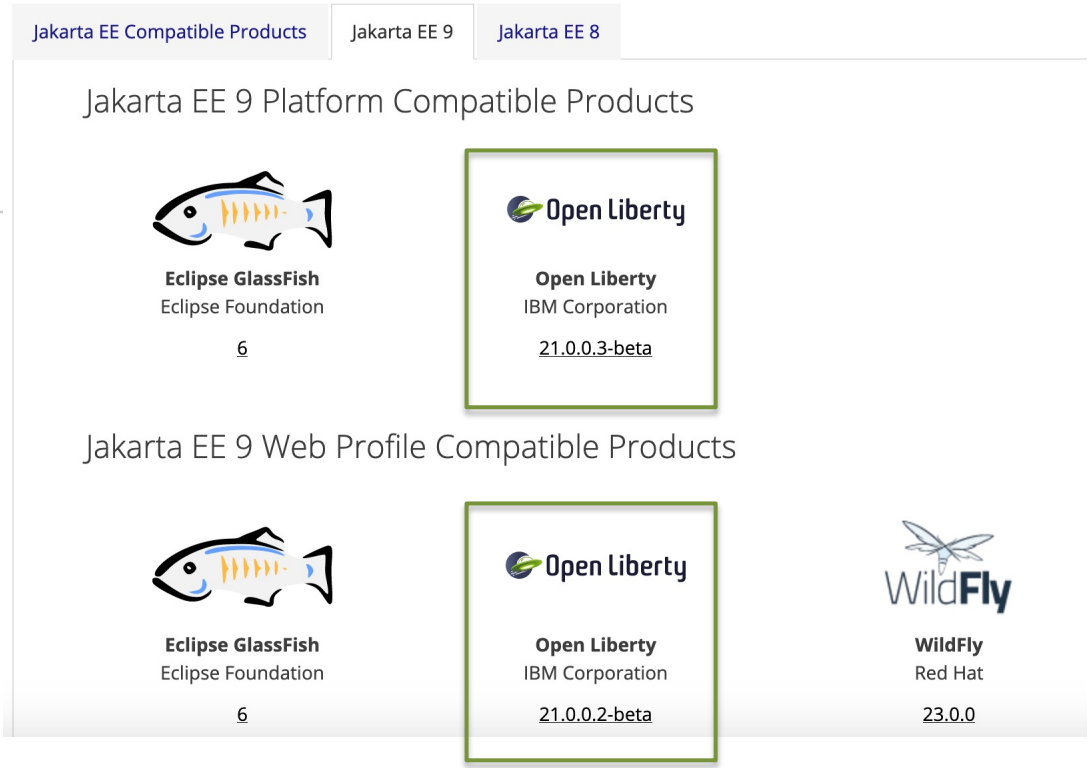
Open Liberty beta is Jakarta EE 9 compatible

Open Liberty beta releases have satisified the Jakarta EE 9 TCK and have been added to the Jakarta EE website

Jared Anderson on Mar 5, 2021

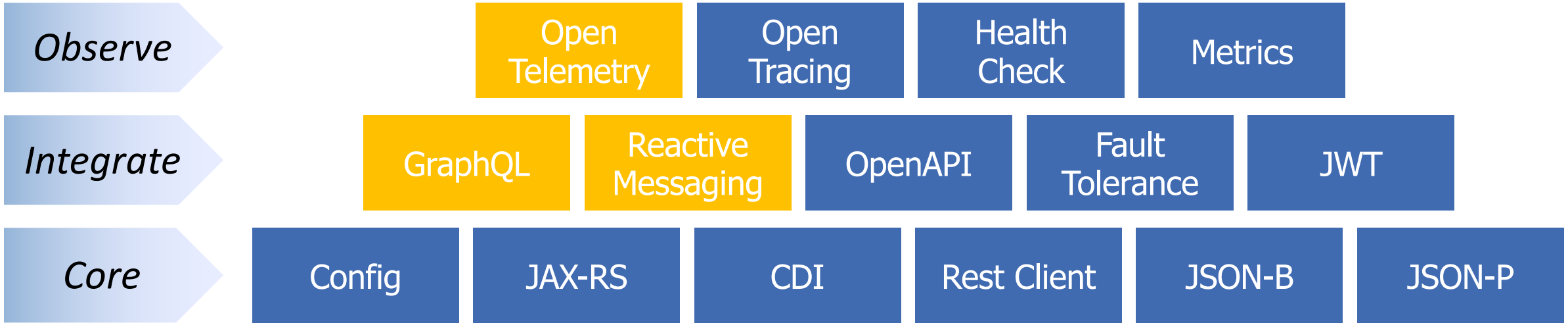
announcements, Java EE, beta, Jakarta EE, community, ...

`javax.* -> jakarta.*`



https://openliberty.io/blog/2021/03/05/jakarta-ee-9-compatibility.html
https://jakarta.ee/compatibility/#tab-9

https://microprofile.io/



Open cloud-native Java APIs

MicroProfile

- Builds on Java EE
- By the Java EE community
- Open Source at Eclipse
- Multiple Implementations

```
@Path(“/”)
public class RestEE {

  @GET
  @Counter
  @Traced
  public String hello() {
    return "Hello MicroProfile!!";
  }
}
```

Server Configuration

Module 4

Simple Config

Open Liberty

```
<server>
  <featureManager>
    <feature>jaxrs-2.1</feature>
  </featureManager>

  <webApplication location="myweb.war" contextRoot="/" />

  <applicationManager autoExpand="true"/>
</server>
```

server.xml

```
-Xmx1g
-Dsystem.prop=value
```

jvm.options

```
WLP_OUTPUT_DIR=/usr/wlp-out/
```

server.env

27

Composing Config


```
<server>
  <httpEndpoint id="defaultHttpEndpoint" host="${host}"
                httpPort="${http}"
                httpsPort="${https}"/>
</server>
```

configDropins/defaults/common-http.xml

```
<server>
  <include location="https://myHost/ports.xml"/>
  <variable name="host" value="${my.host}"/>
  <variable name="http" value="${my.host.http}"/>
  <variable name="https" value="${my.host.https}"/>
</server>
```

configDropins/overrides/ports.xml

28

Externalizing Configuration

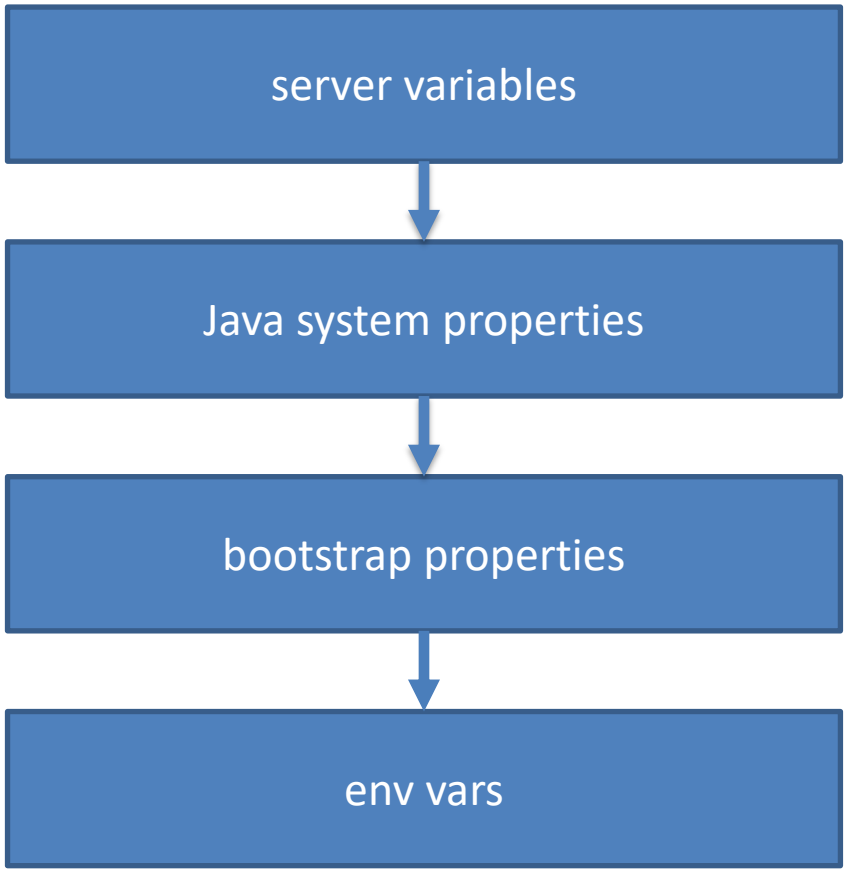
Module 5

Open Liberty

Variable Resolution

Open Liberty

- Liberty variable resolution happens top to bottom



Externalizing Config


```
<server>
  <httpEndpoint id="defaultHttpEndpoint" host="*"
                httpPort="${env.SERVER_HTTP_PORT}"
                httpsPort="${env.SERVER_HTTPS_PORT}"/>
</server>
```

configDropins/defaults/common-http.xml

31

MicroProfile

- Inject Configuration
- Sourced from
  - Properties file
  - Environment
  - Java system properties

```
@Path(“/”)
public class RestEE {

  @Inject
  @ConfigProperty(“name”)
  private String name;

  @GET
  @Counter
  @Traced
  public String hello() {
    return "Hello " + name;
  }
}
```

Testing

Module 6

Testing

- Use maven-surefire-plugin to run unit test
  - Use CDI dependency injection to make your code easy to mock & unit test
- Use maven-failsafe-plugin to run integration test
- Use Liberty Arquillian integration to run tests on a Liberty server
- Use MicroShed Testing for system testing in Containers (Module 8)

Docker

Module 7

Open Liberty

Liberty in Containers

app and config

+

liberty and java

+

os config

+

=

portable container

Hosted Cloud

- IBM Cloud Kubernetes Service
- Azure Kubernetes Service
- Google Kubernetes Engine
- Amazon Elastic Kubernetes Service
- Jelastic

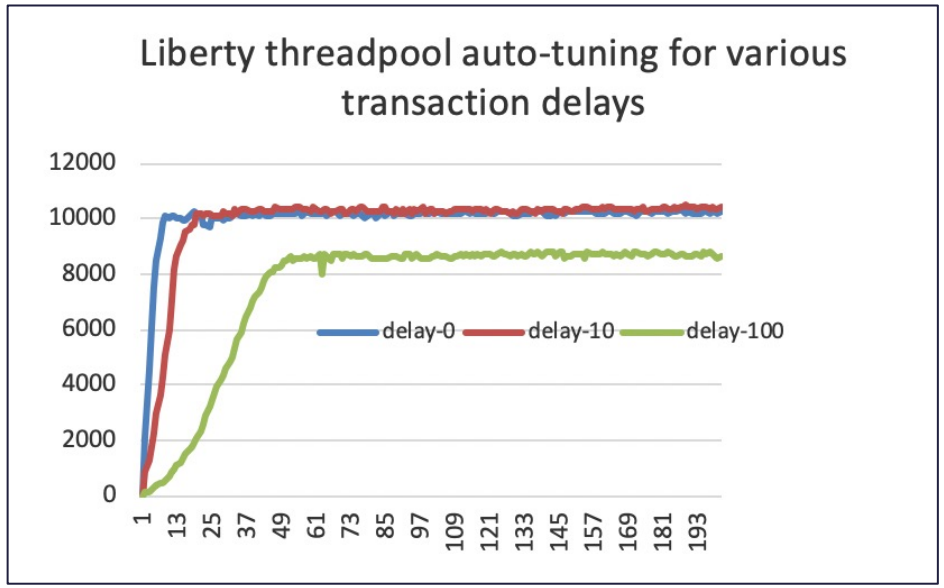
Private Cloud

- Red Hat Open Shift Container Platform
- Pivotal Kubernetes Service
- Pivotal Cloud Foundry

```
FROM open-liberty
ADD myapp.war /config/dropins/myapp.war
```

Making the most of Docker

Kubernetes optimized



Open Liberty Operator
provided by IBM

Deploy and manage applications running on Open Liberty

docker hub

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

Open Liberty

- **Deliver faster** without costly tuning exercises
- Get **optimal performance** even as the environment changes

- **Simple Operator-based deploy** and day-2 operations experience
- Supported **production-ready images**
- **APIs** for Kubernetes integration
- Container-based **usage tracking**

Testing in Containers

Module 8

Open Liberty

MicroShed Testing

Open Liberty

- 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)
- https://microshed.github.io/

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

Support Licensing

Module 9

How to get Support

Open Liberty

WebSphere

z/OS
ND
Base

Core

WebSphere Hybrid Edition

IBM Integrated Application Runtimes

Java:

- WebSphere
- Liberty
- MicroProfile
- Jakarta EE
- OpenJ9

Cloud Foundry Migration Runtime

Transformation Advisor

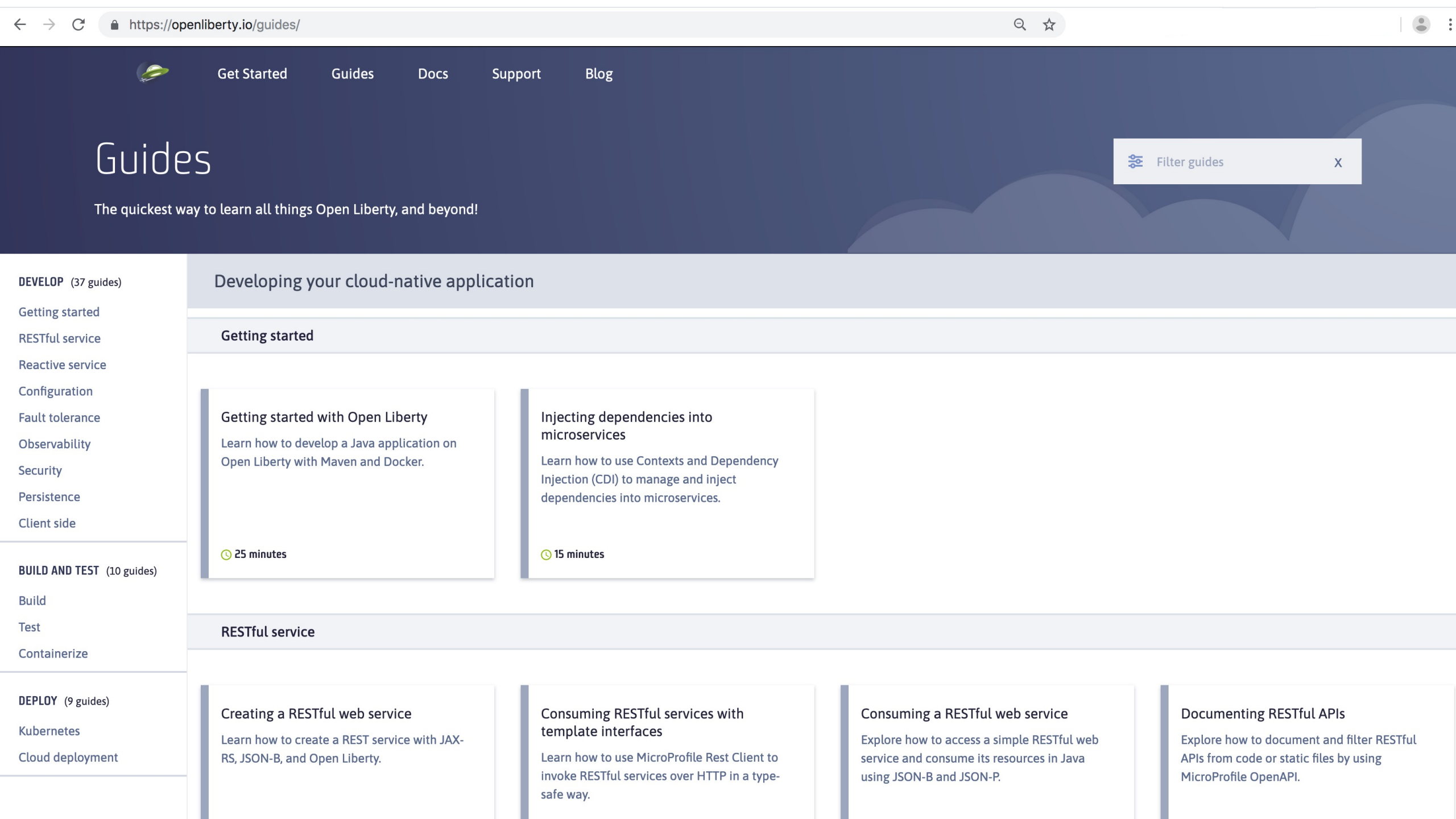
Mono2Micro

Red Hat OpenShift

Wrap-up

Open Liberty

Mastersclass hands-on content


https://openliberty.io/guides/
Get Started
Guides
Docs
Support
Blog
Guides
The quickest way to learn all things Open Liberty, and beyond!
Filter guides
DEVELOP (37 guides)
Getting started
RESTful service
Reactive service
Configuration
Fault tolerance
Observability
Security
Persistence
Client side
BUILD AND TEST (10 guides)
Build
Test
Containerize
DEPLOY (9 guides)
Kubernetes
Cloud deployment
Developing your cloud-native application
Getting started
Getting started with Open Liberty
Learn how to develop a Java application on Open Liberty with Maven and Docker.
25 minutes
Injecting dependencies into microservices
Learn how to use Contexts and Dependency Injection (CDI) to manage and inject dependencies into microservices.
15 minutes
RESTful service
Creating a RESTful web service
Learn how to create a REST service with JAX-RS, JSON-B, and Open Liberty.
Consuming RESTful services with template interfaces
Learn how to use MicroProfile Rest Client to invoke RESTful services over HTTP in a type-safe way.
Consuming a RESTful web service
Explore how to access a simple RESTful web service and consume its resources in Java using JSON-B and JSON-P.
Documenting RESTful APIs
Explore how to document and filter RESTful APIs from code or static files by using MicroProfile OpenAPI.

Thank You

Open Liberty