



Open Liberty

Masterclass: Hands-on Liberty for Developers

Ready to get your hands dirty?



Online exercises: <https://ibm.biz/OpenLibertyMasterclass>

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



- [The Application](#)
- [Module 1: Build](#)
- [Module 2: Dev Mode](#)
- [Module 3: Application APIs](#)
- [Module 4: Server Configuration](#)
- [Module 5: Externalizing Configuration](#)
- [Module 6: Integration Testing](#)
- [Module 7: Docker](#)
 - [Overriding Dev Server Configuration](#)
- [Module 8: Testing in Containers](#)
- [Module 9: Support Licensing](#)
- [Conclusion](#)

Brief History



Liberty Goals



- Efficient
- Simple to use
- Consistency
- Just enough runtime
- End of migration
- Agile-ready



Liberty

- 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



[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



Fit for purpose

- You control which features are loaded into each server instance



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



Liberty Zero Migration



- Zero config migration
 - Write once, run forever
- Zero migration for apps
 - No behavior changes in existing features
 - New behaviors in new features
- Choose your Java
 - Java 12, 11, 8
 - AdoptOpenJDK
 - IBM
 - OpenJDK
 - Oracle



Features


Open Liberty

Periodic Table of Liberty (20.0.0.3)

batchSMFLogging-1.0		zosLocalAdapters-1.0		zosTransaction-1.0		zosSecurity-1.0			
		zosRequestLogging-1.0		zosWlm-1.0					
collectiveController-1.0		dynamicRouting-1.0		healthManager-1.0		scalingController-1.0			
		clusterMember-1.0		healthAnalyzer-1.0		scalingMember-1.0			
cloudant-1.0		sipServlet-1.0		batchManagement-1.0		passwordUtilities-1.0		Security	
javaee-7.0				wsAtomicTransaction-1.2		wsSecurity-1.1			
javaee-8.0						wsSecuritySaml-1.0			
jakartaee-8.0									
bellS-1.0		mpContextPropagation-1.0		adminCenter-1.0		constrainedDelegation-1.0		audit-1.0	
concurrent-1.0		mpReactiveMessaging-1.0		collectiveMember-1.0		federatedRepository-1.0		ldapRegistry-3.0	
javaMail-1.6		mpReactiveStreams-1.0		distributedMap-1.0		jwt-1.0		oauth-2.0	
jaxb-2.2		opentracing-1.3		eventLogging-1.0		jwtSso-1.0		openid-2.0	
jdbc-4.3		osgiConsole-1.0		logstashCollector-1.0		sessionDatabase-1.0		openidConnectClient-1.0	
jpaContainer-2.2		springBoot-2.0		monitor-1.0		webCache-1.0		openidConnectServer-1.0	
jsfContainer-2.3		webProfile-7.0		openapi-3.1				samlWeb-2.0	
json-1.0		webProfile-8.0		requestTiming-1.0				scim-1.0	
jsonbContainer-1.0				usageMetering-1.0				socialLogin-1.0	
jsonpContainer-1.1				restConnector-2.0				spnego-1.0	
microProfile-3.2				sessionCache-1.0				transportSecurity-1.0	

Periodic Table of Liberty (20.0.0.3)

ZOS

ND

Base

Core

Open Liberty

New in 4Q19

New in 3Q19

New in 2Q19

New in 1Q20

		batchSMFLogging-1.0			
collectiveController-1.0	dynamicRouting-1.0	appClientSupport-1.0	ejbHome-3.2	jacc-1.5	managedBeans-1.0
clusterMember-1.0		appSecurityClient-1.0	ejbPersistentTimer-3.2	jaxb-2.2	mdb-3.2
cloudant-1.0	sipServlet-1.0	batch-1.0	ejbRemote-3.2	jaxws-2.2	wasJmsClient-2.0
javaee-7.0		concurrent-1.0	j2eeManagement-1.1	jca-1.7	webProfile-8.0
javaee-8.0		ejb-3.2	javaMail-1.6	jms-2.0	wmqJmsClient-2.0
jakartaee-8.0		mpContextPropagation-1.0			
bells-1.0	mpReactiveMessaging-1.0	appSecurity-3.0	jaxrs-2.1	jsonp-1.1	websocket-1.1
concurrent-1.0	mpReactiveStreams-1.0	beanValidation-2.0	jaxrsClient-2.1	jsf-2.3	
javaMail-1.6		cdi-2.0	jdbc-4.2	jsp-2.3	
jaxb-2.2	opentracing-1.3	ejbLite-3.2	jndi-1.0	managedBeans-1.0	
jdbc-4.3	osgiConsole-1.0	el-3.0	jpa-2.2	servlet-4.0	
jpaContainer-2.2	springBoot-2.0	jaspic-1.1	jsonb-1.0	ssl-1.0	
jsfContainer-2.3	webProfile-7.0				
json-1.0	webProfile-8.0	cdi-2.0	jsonb-1.0	mpMetrics-2.2	mpOpenTracing-1.3
jsonbContainer-1.0		jaxrs-2.1	mpConfig-1.3	mpJwt-1.1	mpRestClient-1.3
jsonpContainer-1.1		jsonp-1.1	mpFaultTolerance-2.0	mpOpenAPI-1.1	mpHealth-2.1
microProfile-3.2					
APIs					

Periodic Table of Liberty (20.0.0.3)

ZOS

ND

Base

Core

Open Liberty

New in 4Q19

New in 3Q19

New in 2Q19

New in 1Q20

		batchSMFLogging-1.0			
collectiveController-1.0	dynamicRouting-1.0	appClientSupport-1.0	ejbPersistentTimer-3.2	jaspic-1.1	managedBeans-1.0
cloudant-1.0	clusterMember-1.0	batch-1.0	ejbRemote-3.2	jaxb-2.2	mdb-3.2
javaee-7.0	sipServlet	concurrent-1.0	j2eeManagement-1.1	jaxws-2.2	wasJmsClient-2.0
javaee-8.0		ejb-3.2	javaMail-1.5	jca-1.7	webProfile-7.0
jakartaee-8.0		ejbHome-3.2	jacc-1.5	jms-2.0	wmqJmsClient-2.0
bells-1.0	mpContextPropagation-	appSecurity-2.0	jaxrsClient-2.0	jsp-2.3	wsSecurity-1.1
concurrent-1.0	mpReactiveMessaging-	beanValidation-1.1	jdbc-4.2	managedBeans-1.0	wsSaml-1.0
javaMail-1.6	mpReactiveStreams-1.0	cdi-1.2	jndi-1.0	servlet-3.1	wsdl-3.0
jaxb-2.2	opentracing-1.3	ejbLite-3.2	jpa-2.1	ssl-1.0	wsdl-3.0
jdbc-4.3	osgiConsole-1.0	el-3.0	jsonp-1.0	websocket-1.1	connectClient-1.0
jpaContainer-2.2	springBoot-2.0	jaxrs-2.0	jsf-2.2		connectServer-1.0
jsfContainer-2.3	webProfile-7.0				
json-1.0	webProfile-8.0	cdi-2.0	jsonb-1.0	mpMetrics-2.2	mpOpenTracing-1.3
jsonbContainer-1.0		jaxrs-2.1	mpConfig-1.3	mpJwt-1.1	mpRestClient-1.3
jsonpContainer-1.1		jsonp-1.1	mpFaultTolerance-2.0	mpOpenAPI-1.1	mpHealth-2.1
microProfile-3.2	APIs				

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

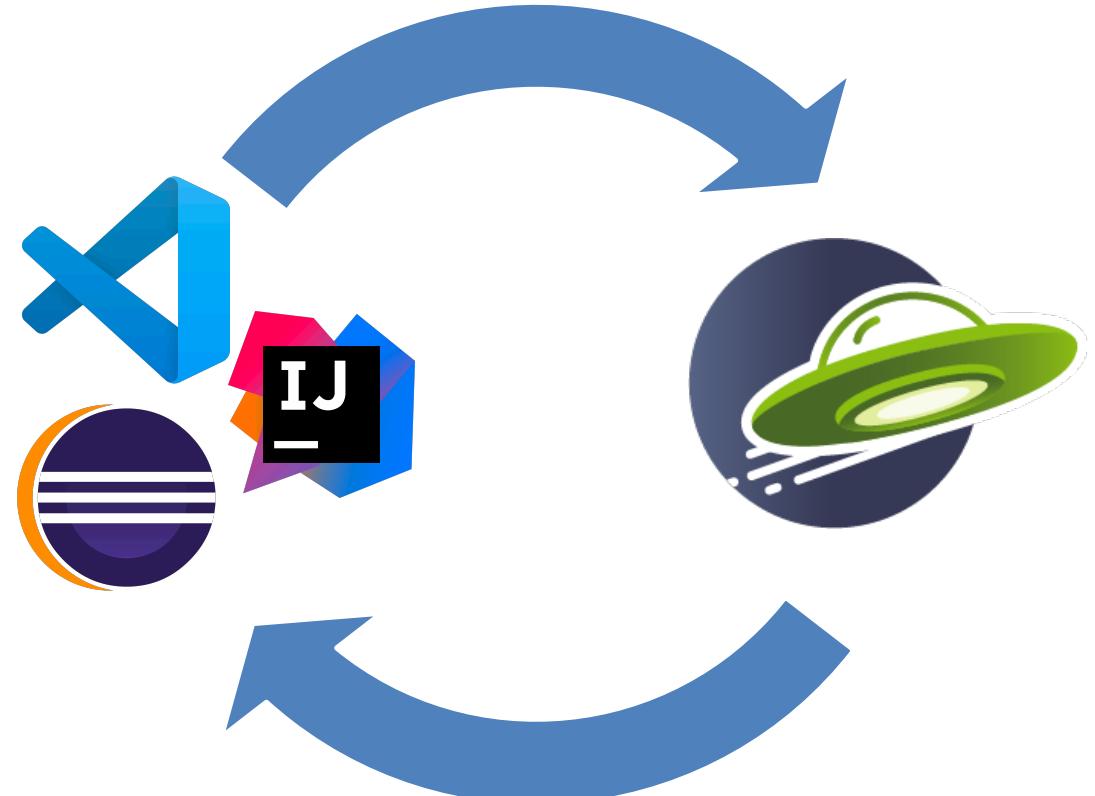


dev mode



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

mvn liberty:dev
gradle libertyDev



Open Liberty Tools for VS Code



- Launch dev mode easily to build with Liberty from popular editor
- Auto detect Liberty projects
- View unit and integration test reports

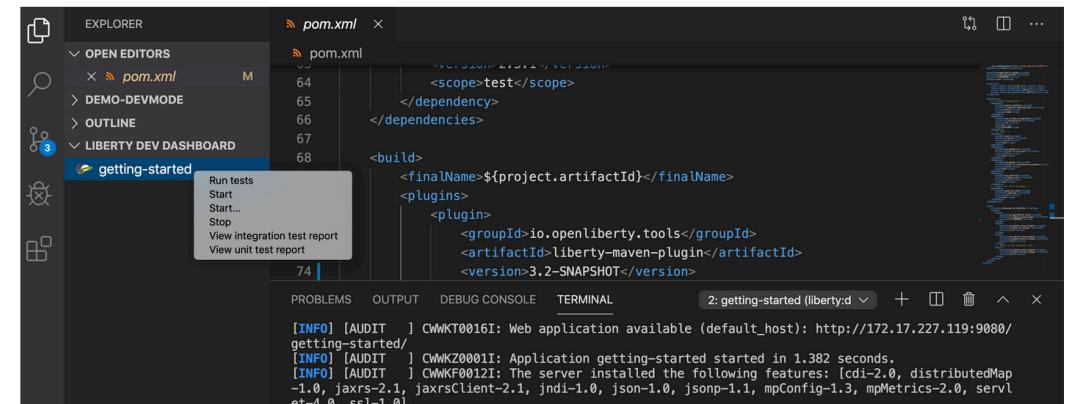


Overview Q & A Rating & Review

Open Liberty Tools for VS Code

Visual Studio Marketplace v0.1.5 license EPL 2.0

A VS Code extension for Open Liberty. The extension will detect your Liberty Maven or Liberty Gradle project if it detects the `io.openliberty.tools:liberty-maven-plugin` in the `pom.xml` or `io.openliberty.tools:liberty-gradle-plugin` in the `build.gradle`. Through the Liberty Dev Dashboard, you can start, stop, or interact with Liberty dev mode on all available [Liberty Maven](#) or [Liberty Gradle](#) projects in your workspace.



Application APIs

Module 2



Java EE 7



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

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

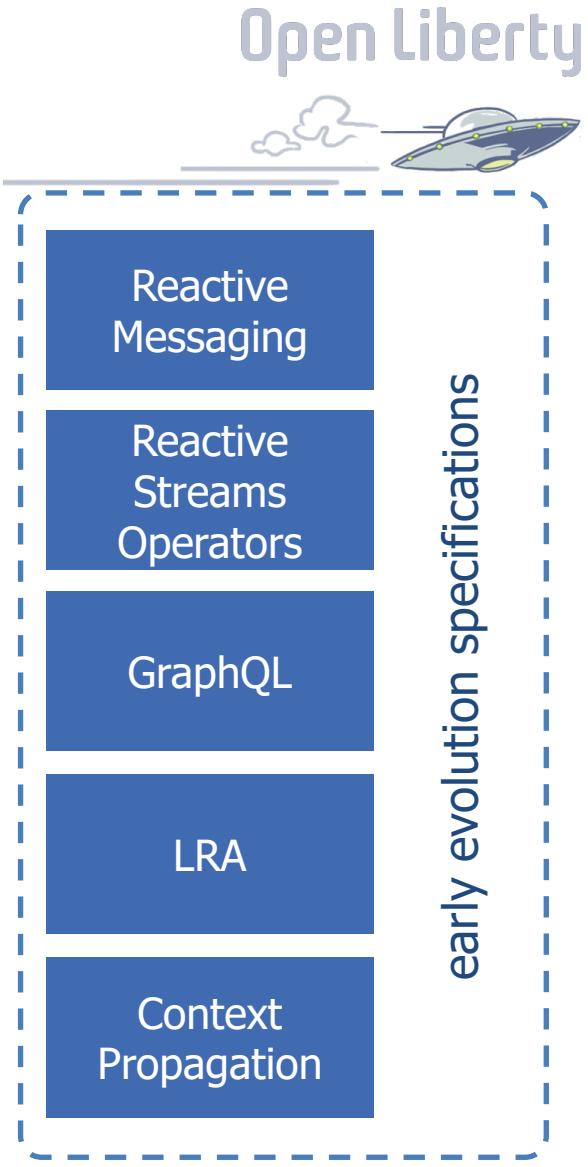
Jakarta EE or Java EE 8



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	

Eclipse MicroProfile



Eclipse 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



```
<server>
  <featureManager>
    <feature>jsp-2.3</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

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

Externalizing Configuration

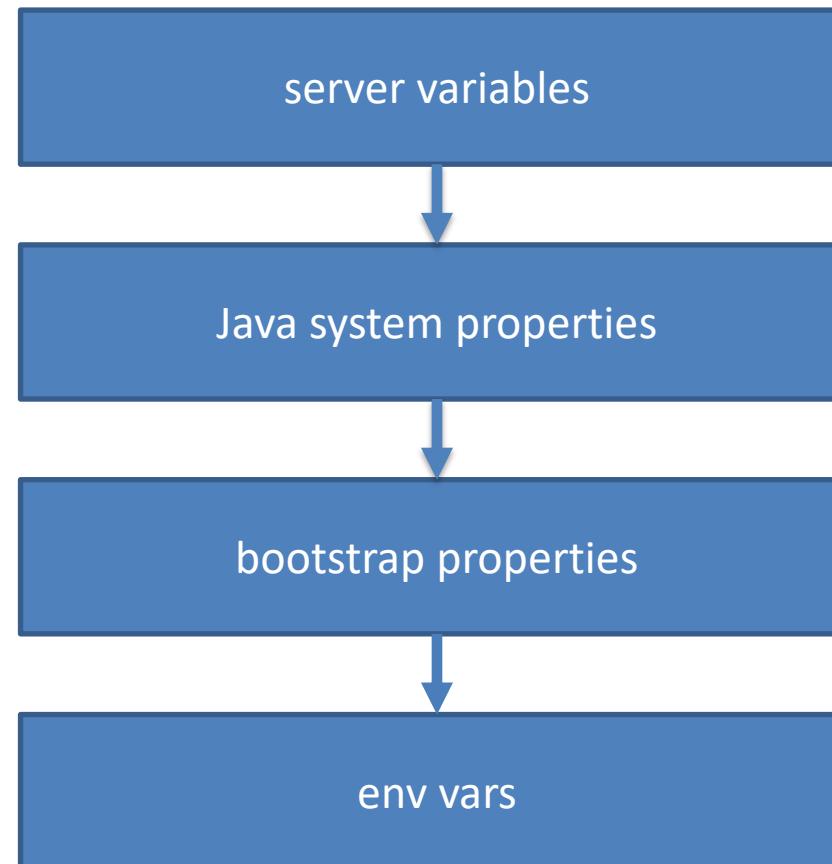
Module 5



Variable Resolution



- 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

Eclipse 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



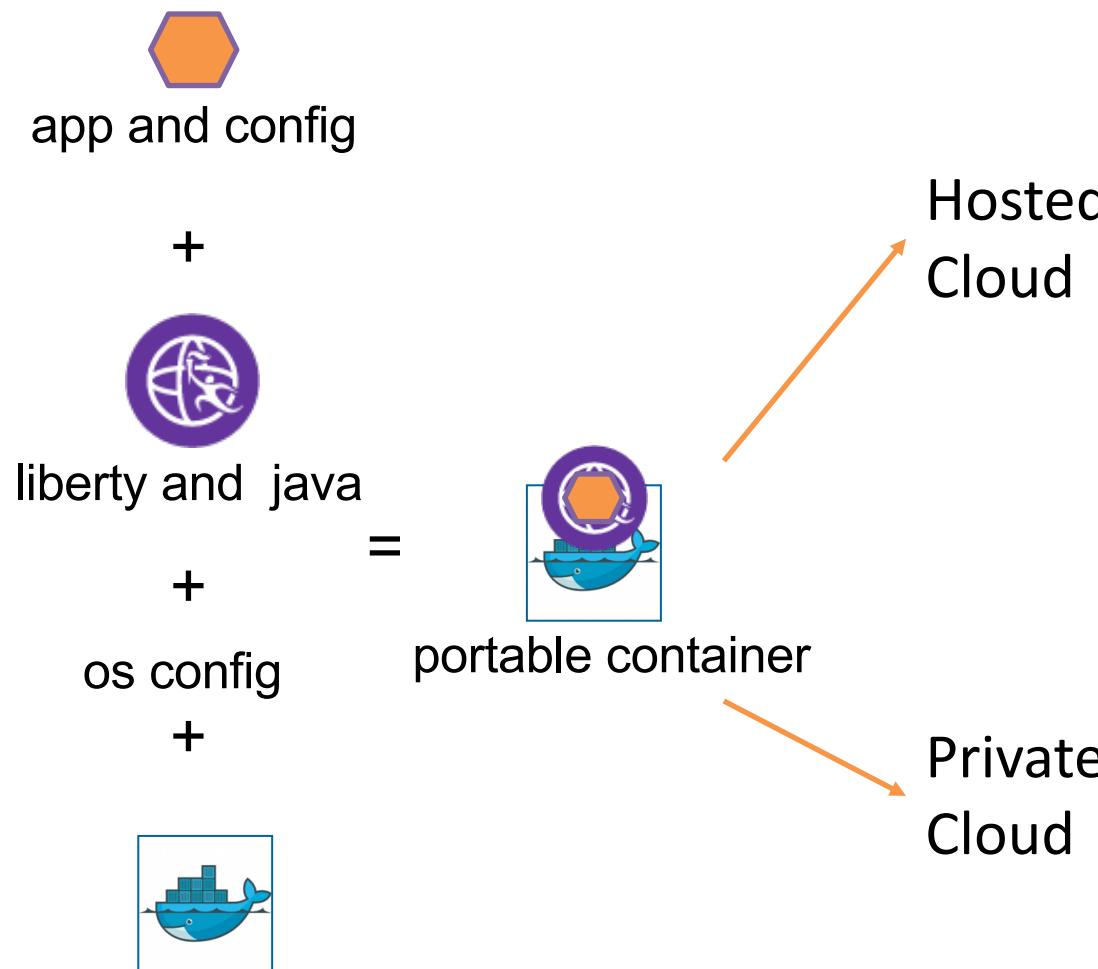
- Use maven-surefire-plugin to run unit test
 - Use CDI dependency injection to make your code easy to mock & unit test
- Use maven-failsafe-plugin to run integration test
- Use Liberty Arquillian integration to run tests on a Liberty server
- Use MicroShed Testing for system testing in Containers (Module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

Docker

Module 7



Liberty in Containers



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

- 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



Testing in Containers

Module 8



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)
- <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;

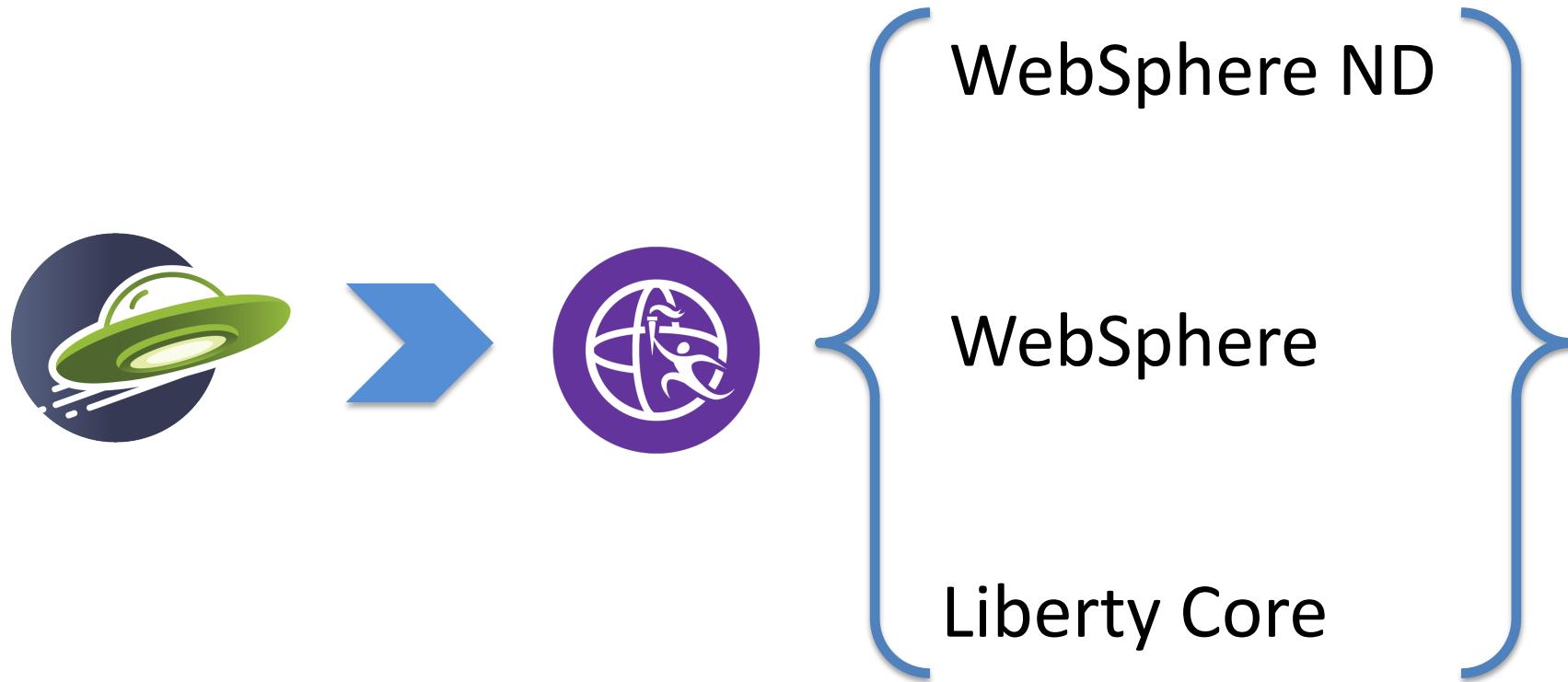
    // A test method like any other
    @Test
    public void testMyService() {
        ...
    }
}
```

Support Licensing

Module 9



Buy WebSphere Liberty



Monthly VPC
(List Price)
US\$341

US\$91.20

US\$45

Licensing Options

You are not alone...

Whether you're an enterprise or a smaller crew, there's Open Liberty support for all

IBM Cloud Pak for
Applications

I'm an IBM customer wanting to
modernize to OpenShift.

IBM Support

WebSphere
Application Server

I just want support for Open
Liberty.

IBM Support

Red Hat Runtimes

I'm a Red Hat customer wanting
to deploy into OpenShift.

Red Hat Support

Wrap-up



Mastersclass hands-on content

- [Open Liberty Masterclass](#)
 - [Table of Contents](#)
 - [Before you begin](#)
 - [Install Pre-requisites](#)
 - [Prime Maven and Docker Caches](#)
 - [The Application](#)
 - [Module 1: Build](#)
 - [Module 2: Dev Mode](#)
 - [Module 3: Application APIs](#)
 - [Module 4: Server Configuration](#)
 - [Module 5: Externalizing Configuration](#)
 - [Module 6: Integration Testing](#)
 - [Module 7: Docker](#)
 - [Overriding Dev Server Configuration](#)
 - [Module 8: Support Licensing](#)
 - [Conclusion](#)



Guides

The quickest way to learn all things Open Liberty, and beyond!

Filter guides

DEVELOP (29 guides)

[Getting started](#)[RESTful service](#)[Reactive service](#)[Configuration](#)[Fault tolerance](#)[Observability](#)[Security](#)[Persistence](#)[Client side](#)

BUILD AND TEST (8 guides)

[Build](#)[Test](#)[Containerize](#)

DEPLOY (8 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

