
Open Liberty
Cloud-native Java innovations that matter with Open Liberty
For JUG Philippines

Docs Get Started Support Fork the code

openliberty.io

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.

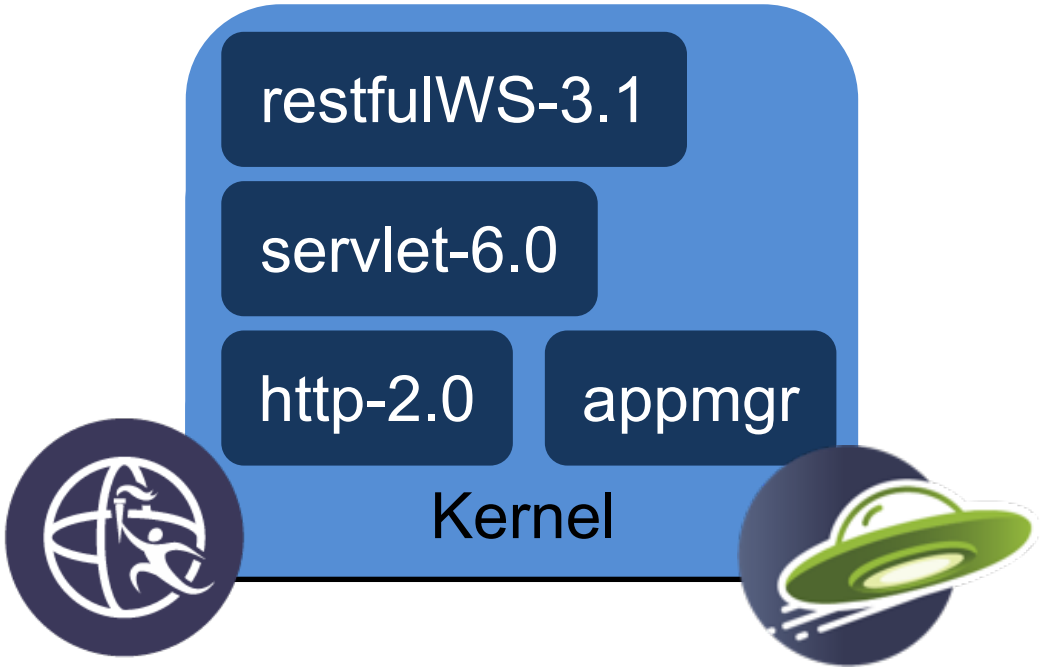
Get Open Liberty

Just Enough Application Runtime

With a *traditional application server*, the full API stack as well as administration and operations features are loaded in each server instance

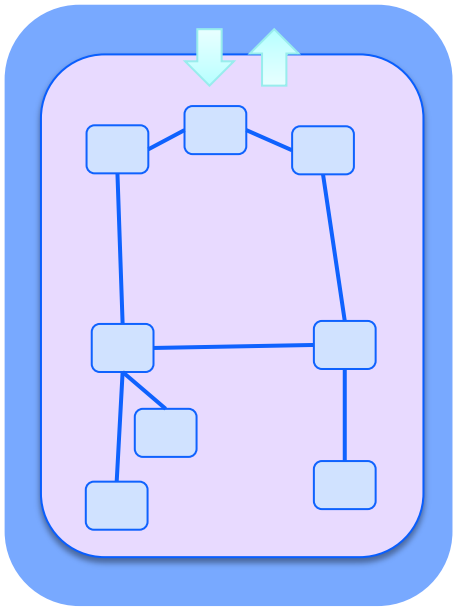
With *Liberty*, you control which features are loaded into each runtime instance

```
<feature>restfulWS-3.1</feature>
```

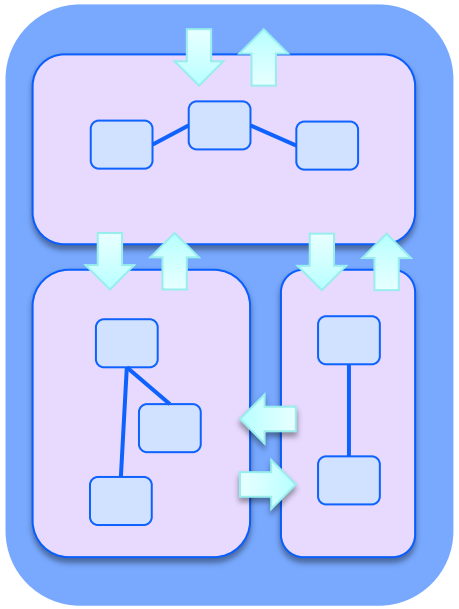


Enabling Spectrum of Architecture Styles

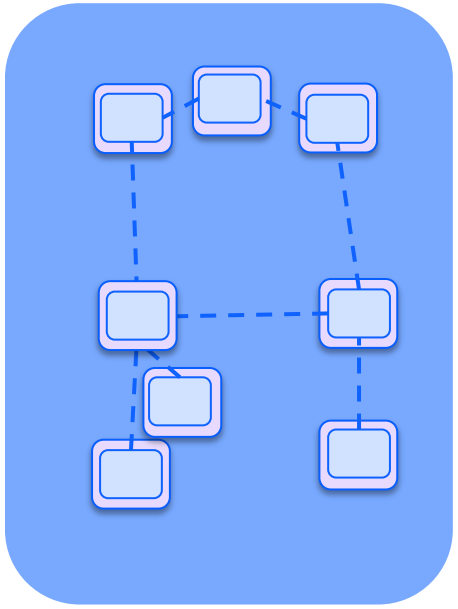
Open Liberty



Monolith



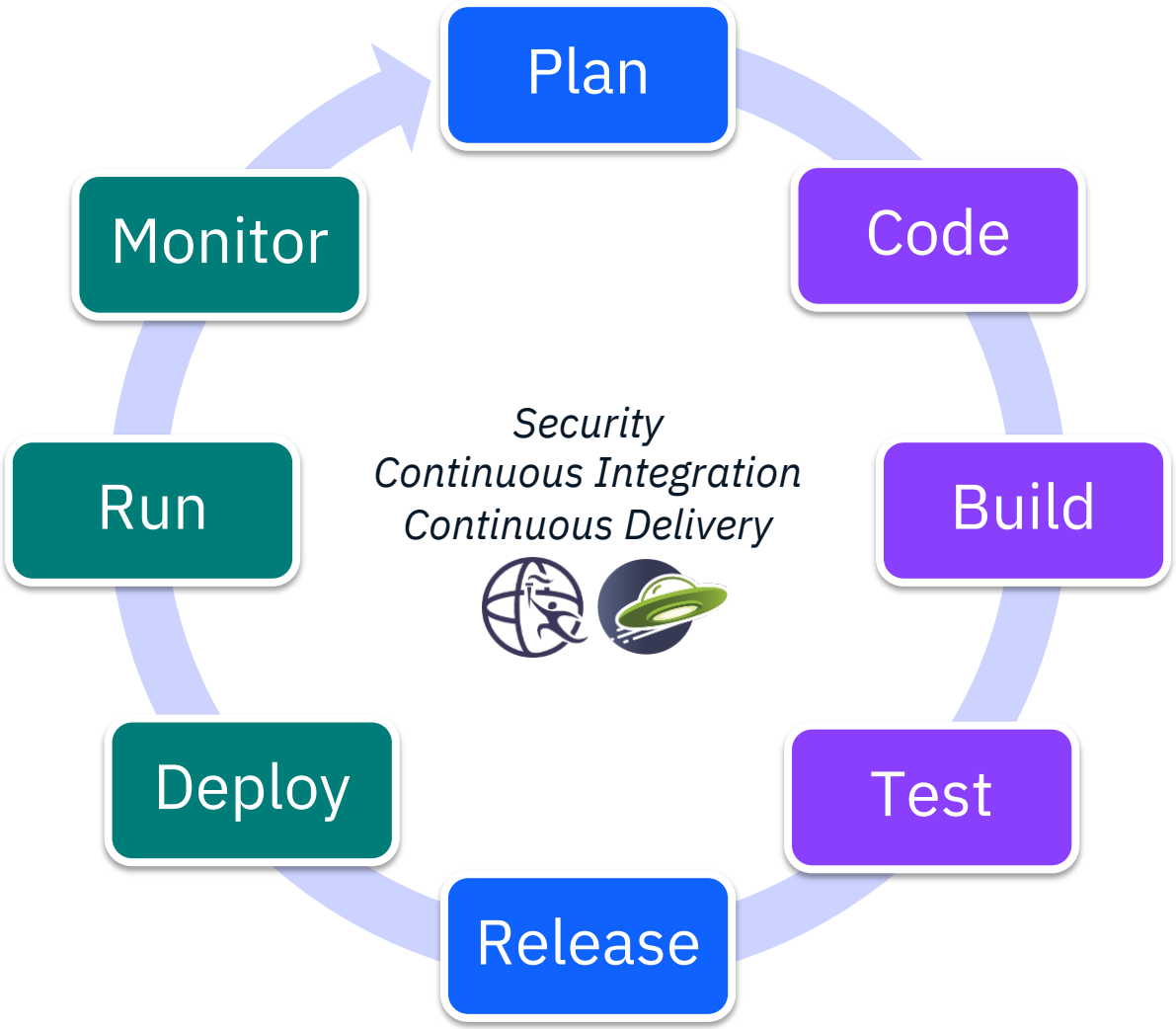
“Macroservices”



Microservices

Cloud-native Delivery

An end-to-end cloud-native DevOps delivery experience with Liberty



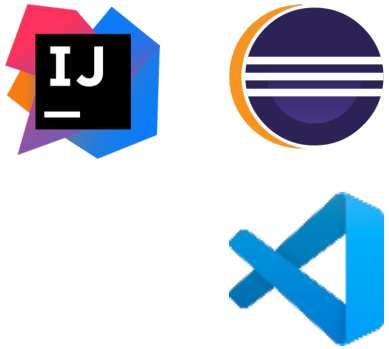
Liberty Tools

Rapid iterative development in your IDE of choice

Code

New — start.openliberty.io

Code — Code and config assist
Dev Mode



Build — Maven and Gradle plugins
Cloud Native Buildpack, Source-2-image, Dockerfile

Test — From Unit Testing to Contract Testing
Integration Testing in Containers

Debug — Hot code replace

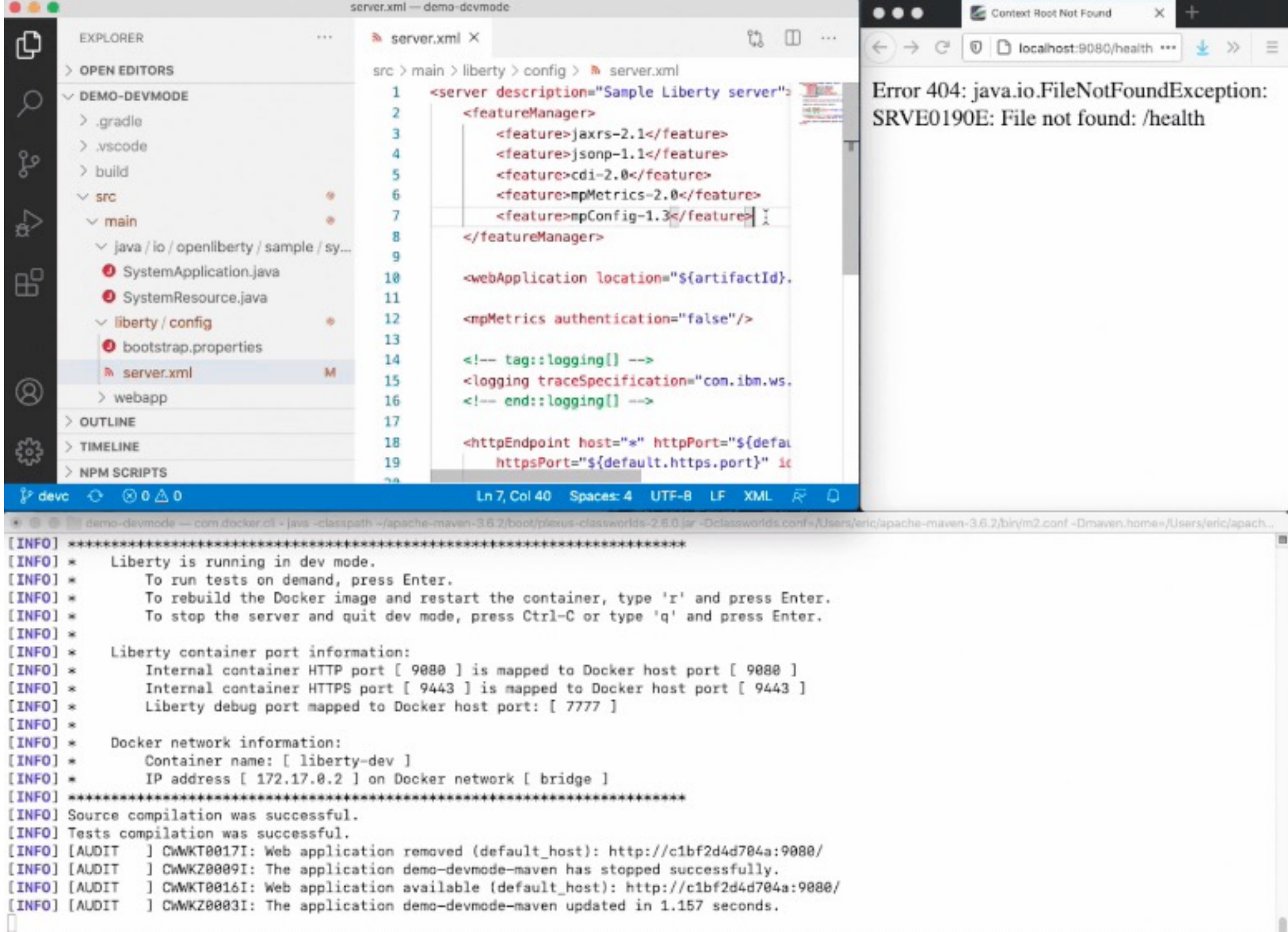
Dev Mode

- No rebuild
- No redeploy
- No install
- No restart

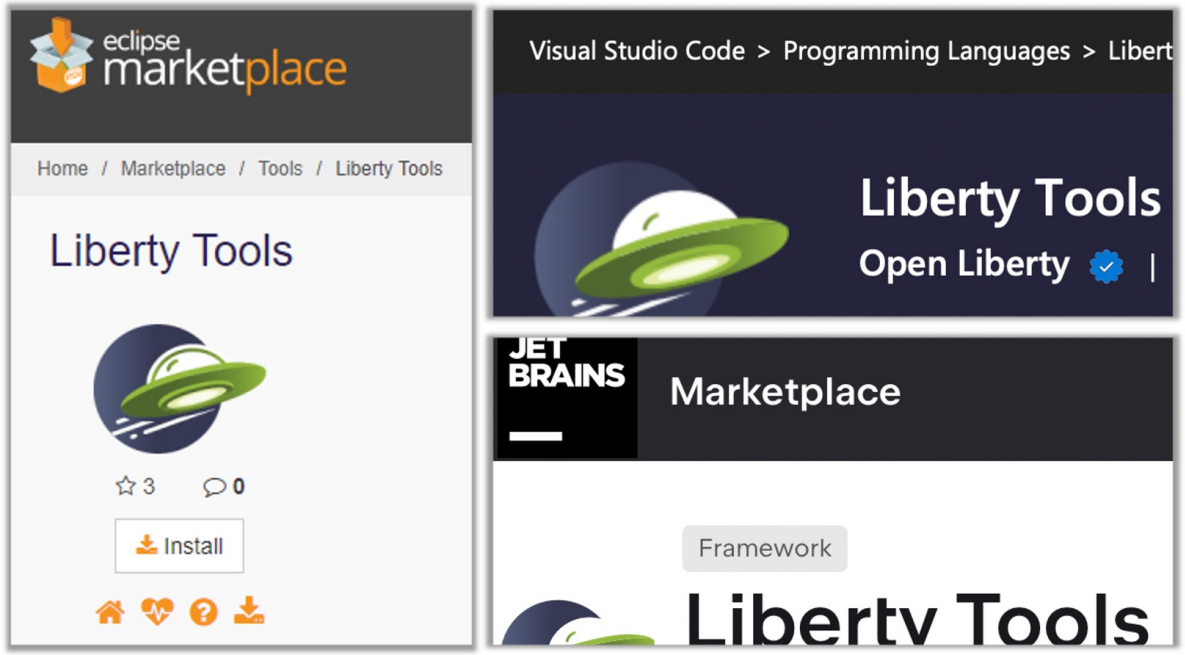
- Just code!

- In containers too

Code



Liberty Tools



- Fast iterative development with Liberty dev mode
- Coding & editing assistance
  - Jakarta EE (9.x and later) & MicroProfile (3.x and later) APIs
  - Liberty configuration files
- Consistent experience across IDEs
- Available in Eclipse IDE, IntelliJ IDEA, and Visual Studio Code marketplaces

APIs

Code

Microservice and Enterprise cloud-native APIs free from vendor lock-in

- Build new open cloud-native microservices with MicroProfile, leveraging existing Java EE/Jakarta EE skills and assets

+

- Modernize existing Java EE applications to cloud-native through Jakarta EE and MicroProfile

JAKARTA EE

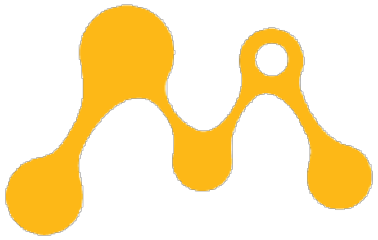
Fully Open Source Cloud Native Java

2023 Devies Award & 2019 Duke's Choice Winner

- The evolution and innovation of enterprise Java technologies under an open, vendor-neutral process
- Delivers open source Full Platform and Web Profile specifications, related Technology Compatibility Kits (TCKs), and compatible implementations

Jakarta EE

Open Liberty

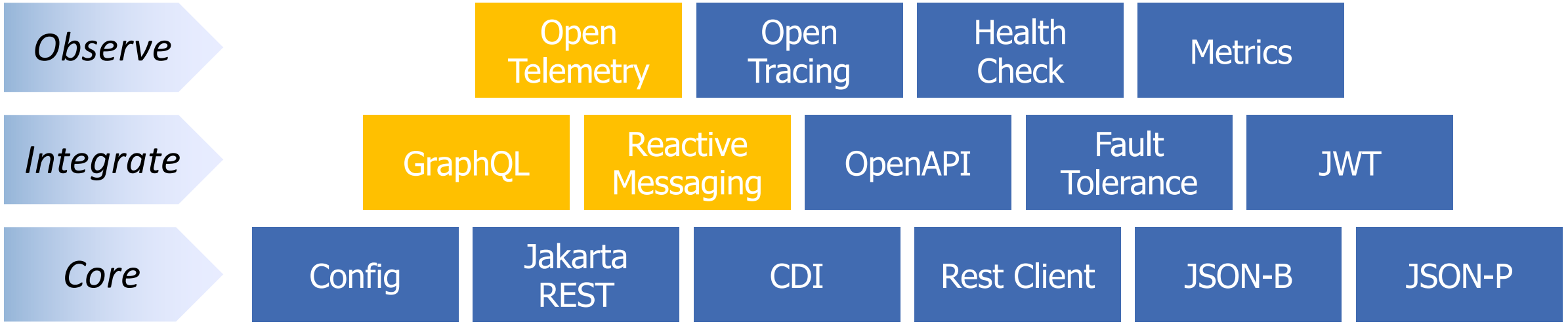


MICROPROFILE™

OPTIMIZING ENTERPRISE JAVA

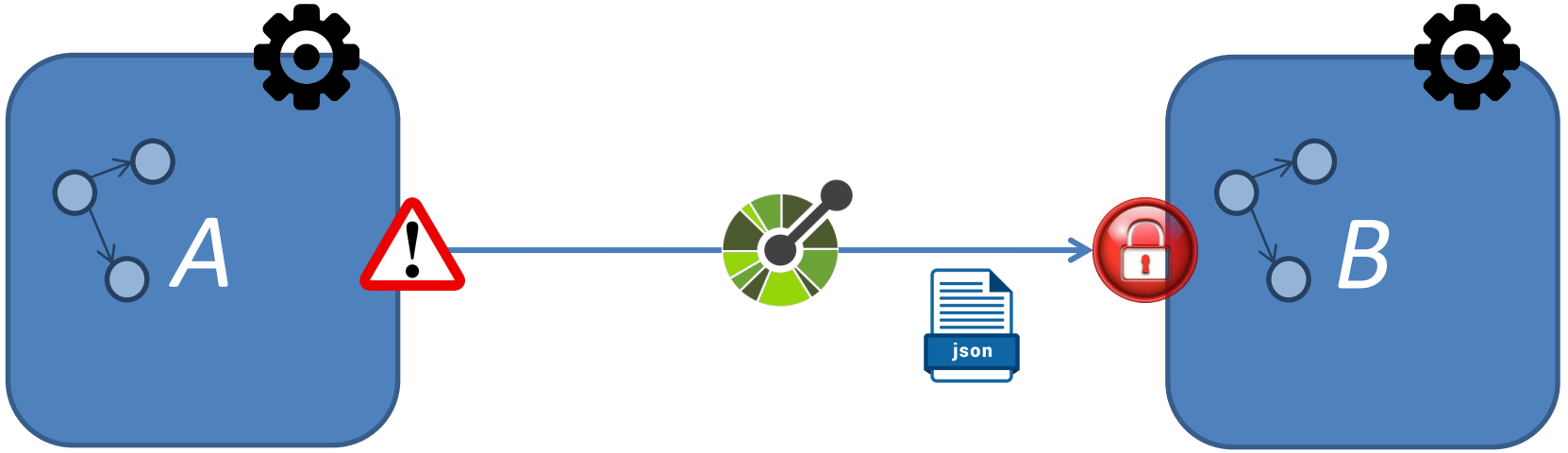
Open Liberty

https://microprofile.io/



Open cloud-native Java APIs for microservices

MicroProfile Config




```
@Inject
@ConfigProperty(name="host",
                defaultValue="foo.com")
Private Optional<String> host;
```

microprofile-config.properties

host=abc.com

MicroProfile Metrics


```
@POST
@Metered
public Response orderCoffee(@Valid @NotNull CoffeeOrder order)
{
  ...
}
```

Build

Build

Friction-free, right-size application and container build

Application Build

- Maven and Gradle Plugins
- All Liberty artefacts released to Maven Central

Package	Size on Disk	Memory
Java EE 8 / Jakarta EE 8 + MicroProfile 3.3	121MB	165MB
MicroProfile 3.3	59MB	113MB
Servlet 4.0	24MB	72MB

Right-size runtime

Container Build

- Leading container build approaches – Dockerfile, Cloud Native Buildpack, Source-2-Image
- Certified Liberty images released to IBM Container Registry

```
FROM icr.io/appcafe/open-liberty:kernel-slim-java8-openj9-ubi

COPY --chown=1001:0 server.xml /config/
RUN features.sh
COPY --chown=1001:0 target/*.war /config/apps
RUN configure.sh
```

Production-ready right-size containers

Testing

Test

Ensuring quality targets met before release

- Unit testing low-level methods/functions testing JUnit
- Integration testing of application components in Liberty with Arquillian
- In-container integration testing with Testcontainers
- Consumer contract testing with PACT

JUnit

Arquillian

Testcontainers

PACT

Developer Experience

Open Liberty

IDEs

Dev Mode

Repositories

The Central Repository

Build

Maven™

Gradle

APIs

Testing

JUnit

PACTS

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!

Liberty Operators

Addressing the Kubernetes skills gap

Deploy

- Insulate from Kubernetes complexities
- Automate common tasks: deploy, scale, upgrade, dump gather
- Security capabilities out-of-the-box
- Reduce configuration by up to 80%

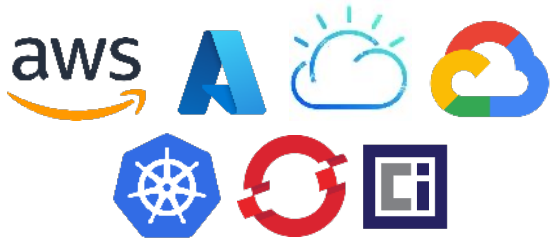
```yaml
apiVersion: liberty.websphere.ibm.com/v1
kind: WebSphereLibertyApplication
metadata:
  name: liberty-cloud-demo
spec:
  license:
    accept: false
    edition: IBM WebSphere Application Server
    productEntitlementSource: Standalone
    metric: Processor Value Unit (PVU)
  replicas: 3
  applicationImage: liberty-cloud-demo:1.0
  pullPolicy: Always
  expose: true
  storage:
    size: 2Gi
    mountPath: "/logs"
```

Cloud Deployments

Deploy

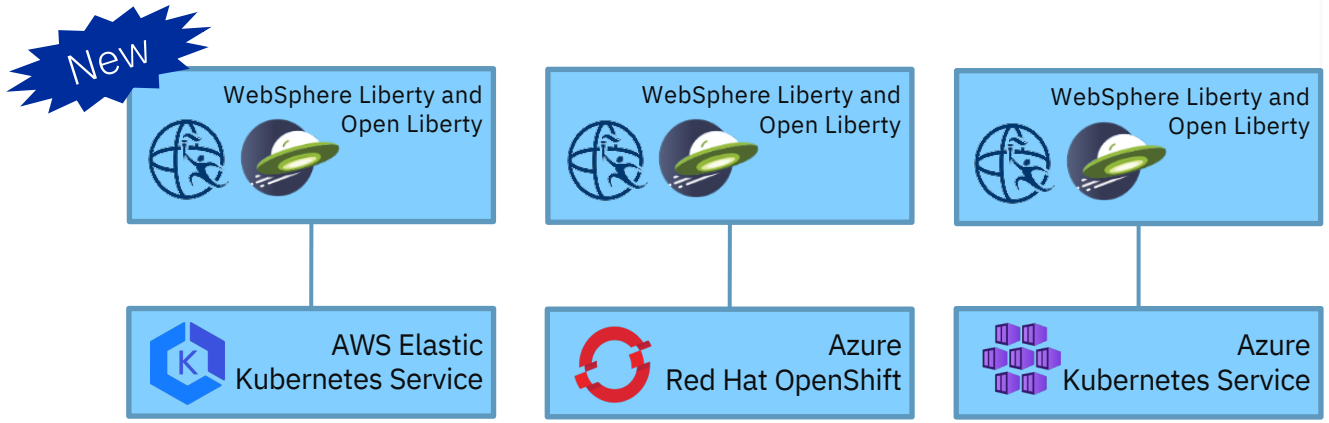
Support on leading clouds

- Liberty supported on all leading Cloud Virtual Machine, Kubernetes and OpenShift Environments
- Azure, AWS, IBM Cloud, Google Cloud, … (Bring-Your-Own-License)
- **New:** AWS ECS Fargate CaaS



Simplified setup

- Marketplace and Partner Solution options simplify and accelerate setup in AWS and Azure at no extra cost
  - Provision or re-use Cluster
  - Provision or re-use container registry
  - Networking and load-balancing
  - Operator install and configuration
  - Application deployment



https://docs.microsoft.com/en-us/azure/developer/java/ee/websphere-family

Configuration

Deploy

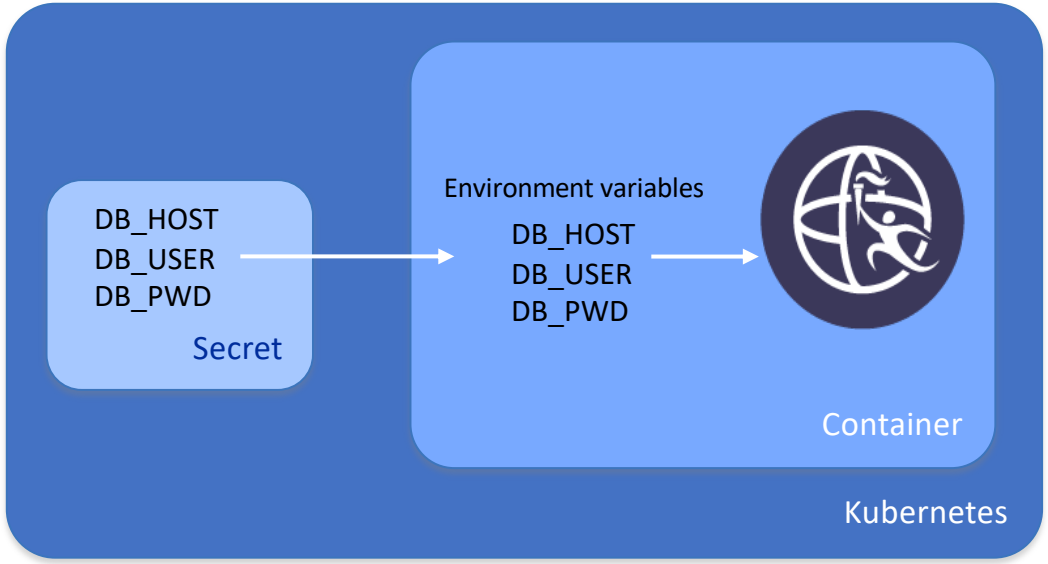
Good configuration management key to CI/CD success

Goals

- Build containers once [1]
- Configure for different environments [2]

Liberty

- Simple, modular, templatizable configuration
- Versionable in source control
- Kubernetes ConfigMaps and Secrets can be mapped to:
  - Environment variables
  - Files – e.g. bootstrap.properties, configDropins/overrides/prod.xml




```
<properties.db2.jcc databaseName="ordersdb"
  serverName="${DB_HOST}"
  user="${DB_USER}" password="${DB_PWD}" />
<variable name="DB_HOST" defaultValue="localhost" />
```

[1] https://12factor.net/dev-prod-parity

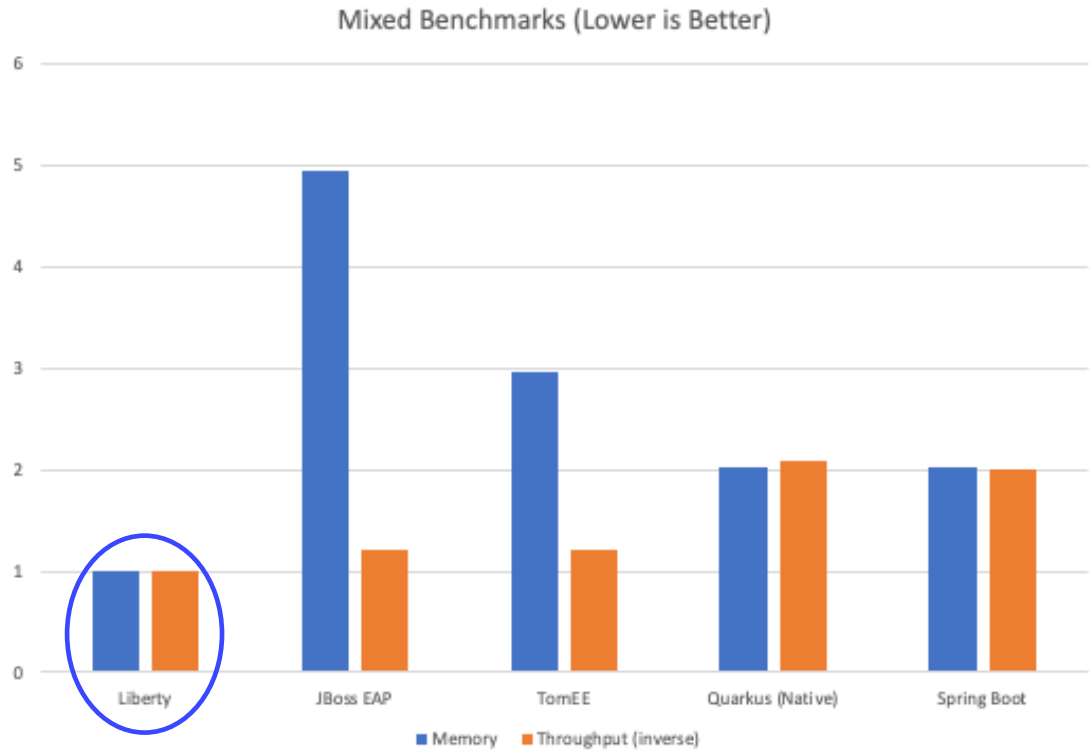
[2] https://12factor.net/config

Performance

Run

Help reduce costs and achieve sustainability targets

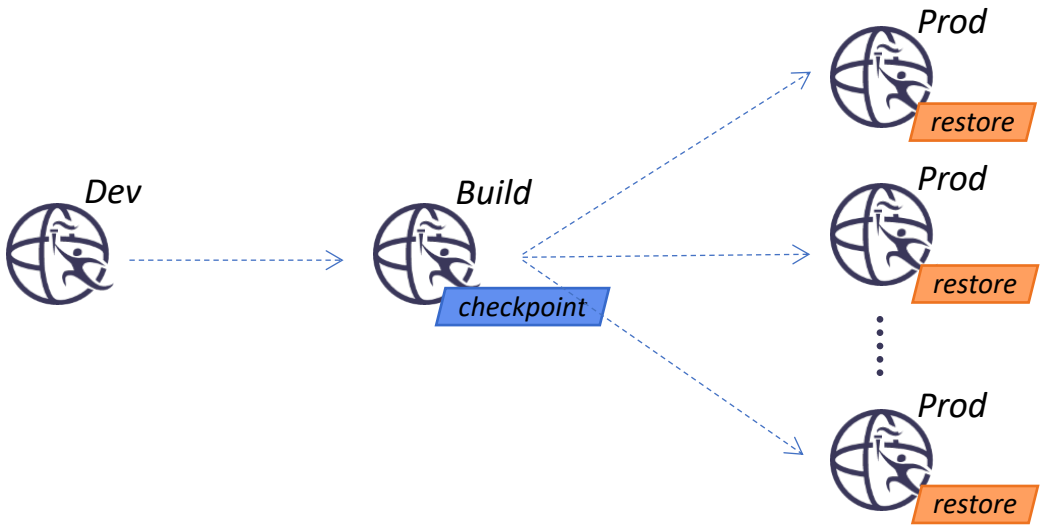
- Better throughput and lower memory mean less hardware for the same workloads
- Less hardware means lower infrastructure and license costs
- Less hardware means lower environmental impact

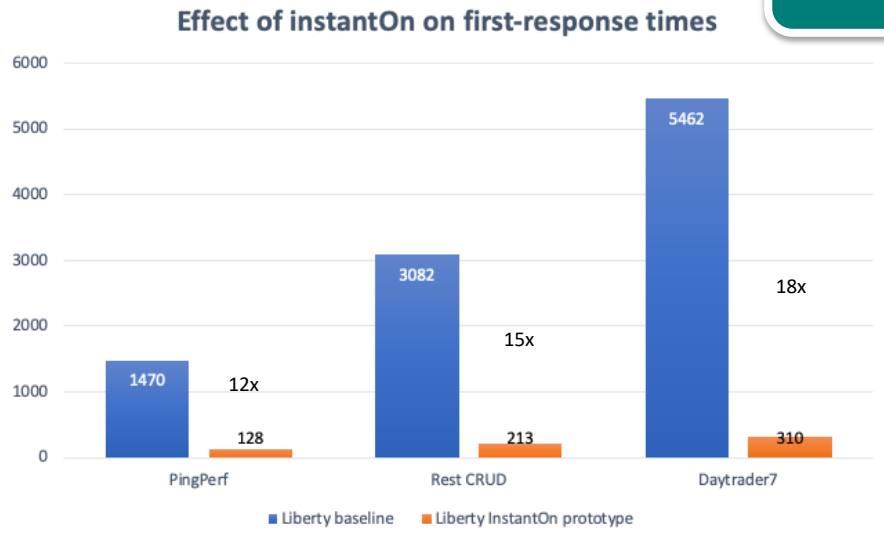


InstantOn without compromise

Run

- Start applications in milliseconds
- Ideal for serverless
- Up to 18x faster
- With all the benefits of the JVM and none of the compromises of Native Image





Characteristics	Semeru InstantOn	Semeru JVM	Graal Native
Full Java support	Yes	Yes	No
'Instant on'	Yes	No	Yes
High throughput	Yes	Yes	No
Low memory (under load)	Yes	Yes	No
Dev-prod parity	Yes	Yes	No

What is Liberty InstantOn?

Liberty InstantOn is a holistic solution that provides fast startup without compromise in containers

- Use checkpoint / restore
  - CRIU Project - https://criu.org
  - Linux technology used to freeze a running process state
  - Checkpoint state used to restore an application
- Select point in startup to freeze an application
  - Restore application and run from the point where checkpoint was taken

- Liberty InstantOn feature requires support for checkpoint / restore to be added to the JDK layer for Liberty to exploit
- IBM Semeru Runtimes JDK is used by default in Liberty containers
- Semeru InstantOn: Refers to the checkpoint / restore support in the IBM Semeru Runtimes JDK
- Collaborative effort between Java framework (Liberty) and JDK (Semeru) to **create production-ready InstantOn feature that is now generally available**

Is Native Image the answer?

Static AOT compile to native image (GraalVM Native Image)

PROS	CONS
• Small footprint • Fast start-up • Great for short-lived apps	• Limited JVM (static vs dynamic compiler optimization, limited GC) • Limited Java features (dynamic, APIs) • Moving target (frameworks, ecosystem, standards) • Dev/Prod parity (devs use compiler?)

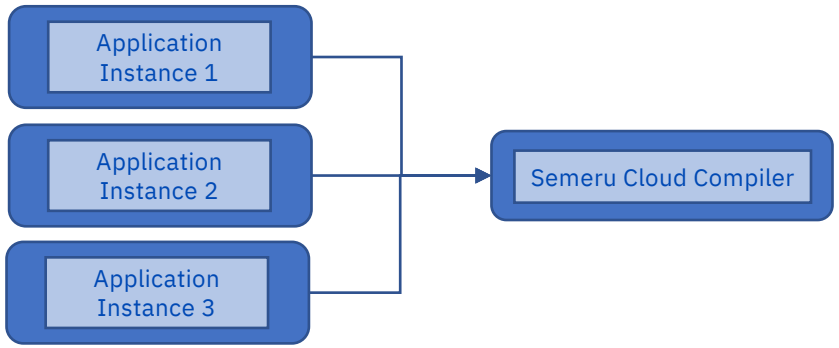
24

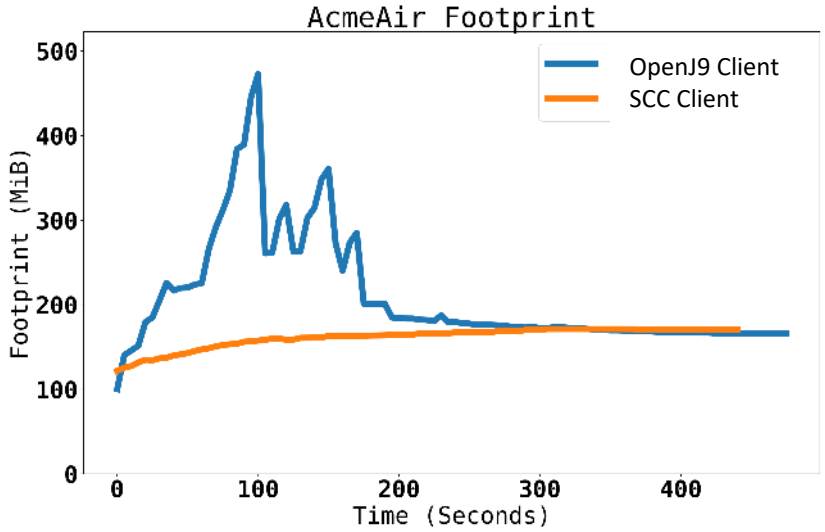
Cloud Optimization

Run

Optimizing memory for Kubernetes deployments

- Offloads costly JIT compilation to separate server
- Dramatically reduces peak memory usage
- Future: Simple Kubernetes enablement through Liberty Operators





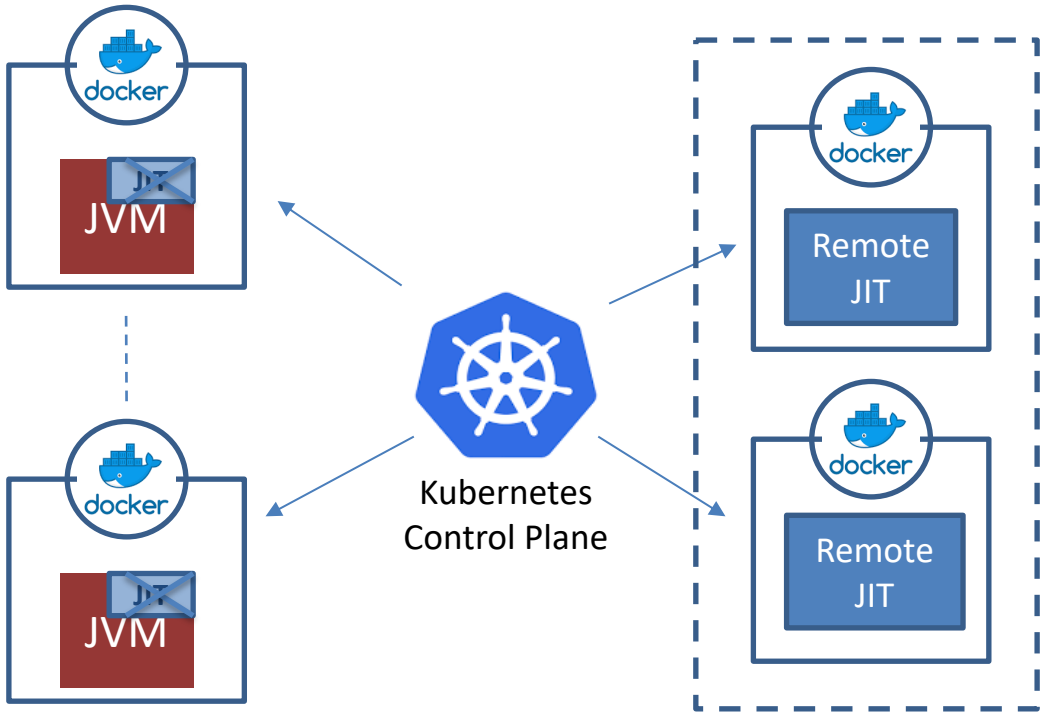
Service	Memory limit w/o SCC Server	Memory limit with SCC Server	Saving
Auth	1,050 MB	750 MB	300MB
Booking	3,300 MB	2,400 MB	900MB
Customer	1,650 MB	1,050 MB	600MB
Flight	2,250 MB	1,250 MB	1,000 MB
Main	600 MB	450 MB	150MB
Total	8,850 MB	5,900 MB	2,950 MB

Semeru Cloud Compiler: JIT-as-a-Service

Open Liberty

Decouple the JIT compiler from the JVM and let it run as an independent process

Offload JIT compilation to remote process



Treat JIT compilation as a cloud service

- Auto-managed by orchestrator
- A mono-to-micro solution
- Local JIT still available

JITServer Advantages for JVM Clients

Provisioning

Easier to size; only consider the needs of the application

Performance

Improved ramp-up time due to JITServer supplying extra CPU power when the JVM needs it the most.

Reduced CPU consumption with JITServer AOT cache

Resiliency

If the JITServer crashes, the JVM can continue to run and compile with its local JIT

Cost

Reduced memory consumption means increased application density and reduced operational cost.

Efficient auto-scaling – only pay for what you need/use.

27

Observe/Day-2

The right information in the right place at the right time

Observability

- All four categories of observability enabled through Liberty runtime, developer instrumentation and leading observability tools

Day-2 Operations

- Day-2 problem determination enabled through Liberty Operators – dumps, traces

Monitor

Logging

- JSON logging (logs, trace, ffdc, access logs, audit logs)
- Integration with java logging API (JUL)
- LogRecordContext (add custom fields to log records)
- LogstashCollector feature, Kibana Dashboards
- Works with common log aggregators (Splunk, Humio, LogDNA, Elastic Stack, etc)

Metrics

- Built-in JVM and Liberty metrics
- App metrics using MicroProfile
- Metrics aggregation and dashboarding with Prometheus and Grafana

Tracing

- Distributed tracing using MicroProfile
- Built-in JAX-RS instrumentation
- App instrumentation using MicroProfile
- Zipkin, Jaeger to aggregate/visualize
- OpenTelemetry under development

Health

- Kube Health endpoint using MicroProfile
- Startup, Liveness, Readiness for different lifecycle states
- App health checks using MicroProfile

CI/CD

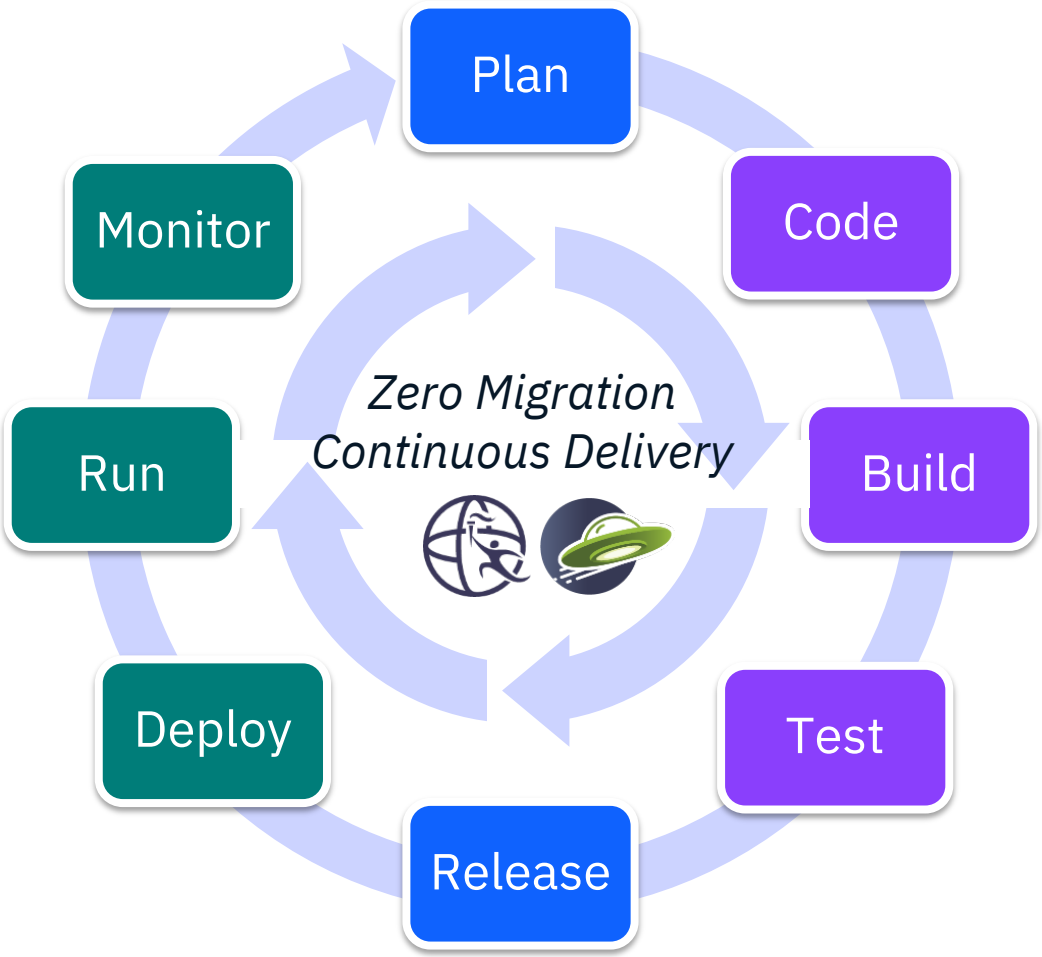
Seamless currency to eliminate technical debt and stay secure

Zero Migration makes staying current easy

- No configuration or runtime behavior changes

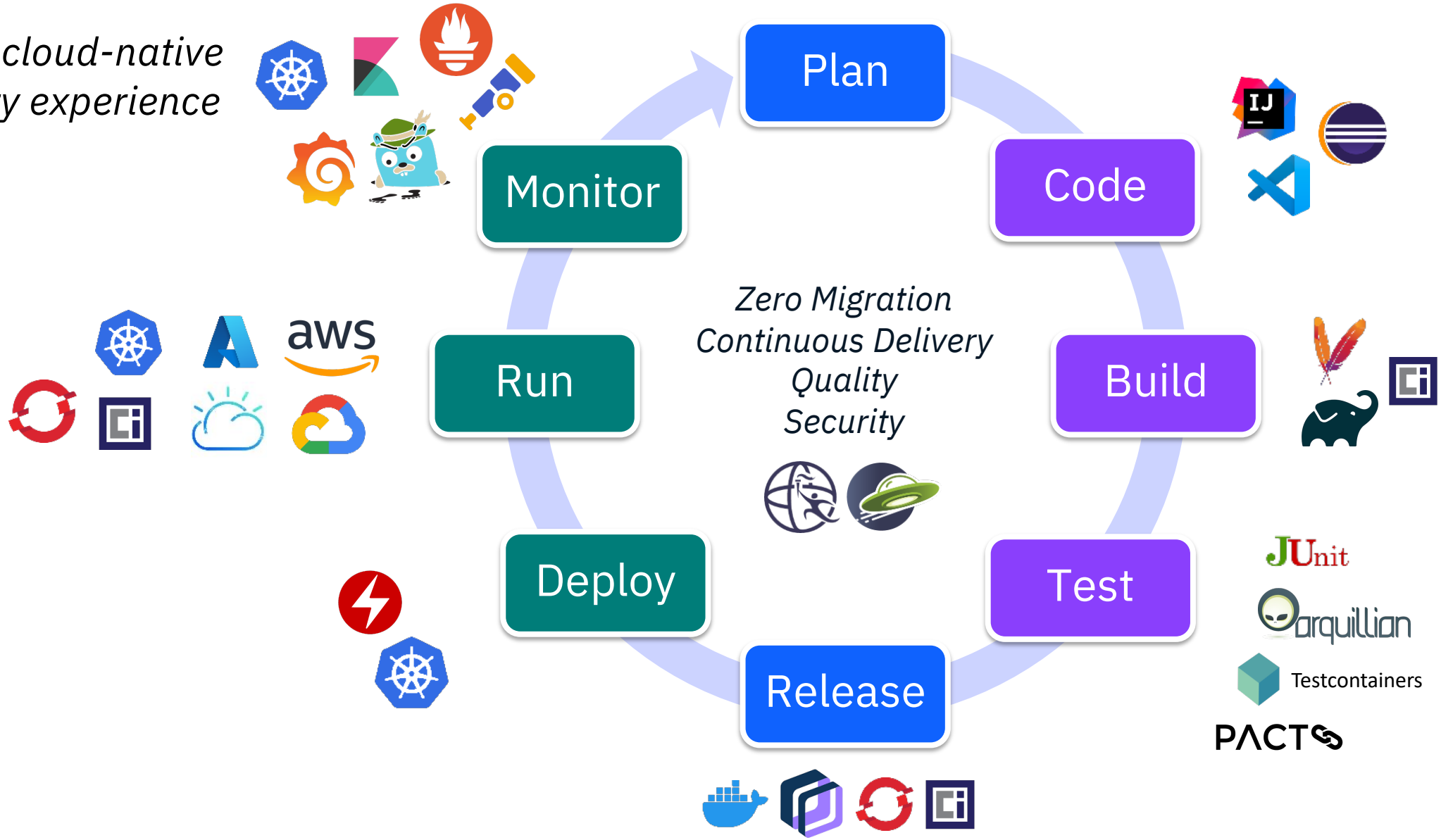
Continuous Delivery gives frequent and reliable access to the latest features and fixes

- Full releases every 4 weeks
- Quarterly LTS releases



Cloud-native Liberty

An end-to-end cloud-native DevOps delivery experience with Liberty



Demos

Open Liberty

Give Liberty a Try!

Docs Get Started Support Fork the code

openliberty.io

@OpenLibertyIO

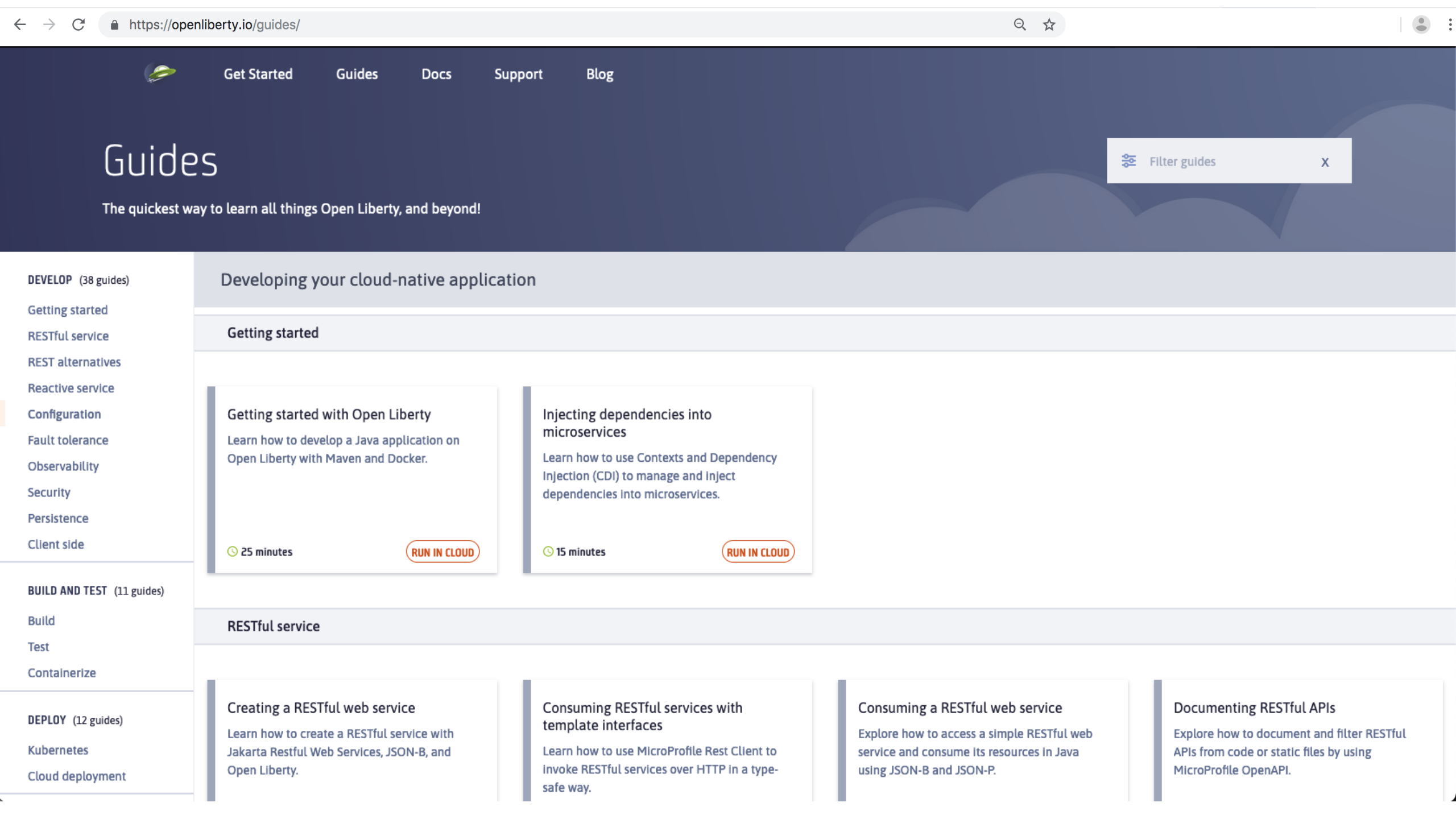
Open Liberty

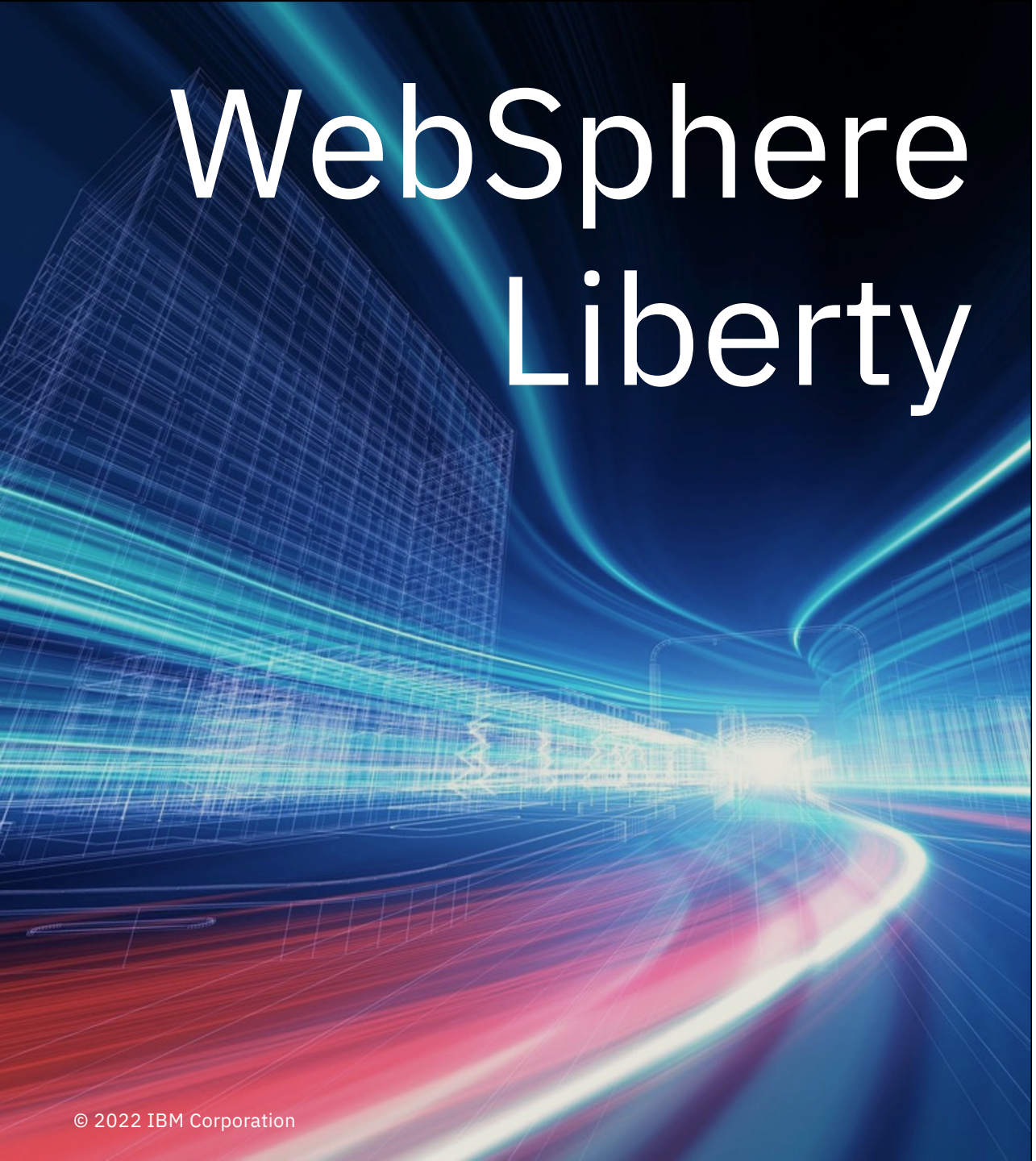
An IBM Open Source Project

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


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 (38 guides)
Getting started
RESTful service
REST alternatives
Reactive service
Configuration
Fault tolerance
Observability
Security
Persistence
Client side
BUILD AND TEST (11 guides)
Build
Test
Containerize
DEPLOY (12 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
RUN IN CLOUD
Injecting dependencies into microservices
Learn how to use Contexts and Dependency Injection (CDI) to manage and inject dependencies into microservices.
15 minutes
RUN IN CLOUD
RESTful service
Creating a RESTful web service
Learn how to create a RESTful service with Jakarta Restful Web Services, 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.



A world-leading, open source application runtime for new cloud-native and modernized workloads. Lightweight, efficient and simple to use, enabling businesses to reduce costs and increase agility. Fully supported by IBM.

Built on open source

Open Liberty

https://openliberty.io/

Thank You

YeeKang.Chang@ibm.com

@yeekangc