Liberty Quarterly Update 21.0.0.1-21.0.0.3

Alasdair Nottingham – Liberty Lead Architect

@nottycode

Agenda

Part 1: 20 Minute Liberty overview

Part 2: Security Update

Part 3: What is new this quarter

Part 4: Q&A

20 minute overview

Open Liberty

6 reasons why Liberty

Lightweight, highly-efficient runtime

CI/CD optimized operational experience

Simple true-to-production developer experience

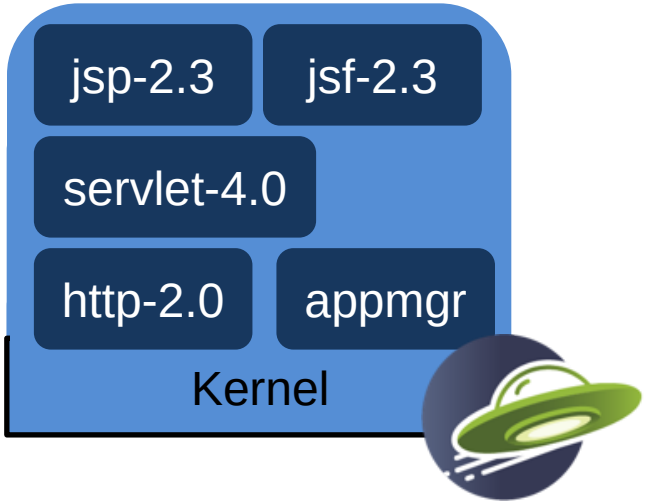
Just enough runtime	80% disk and 56% memory saving
Low operating cost	4x increased density over Tomcat & Spring Boot
Continuous delivery	Zero-effort security fixing & zero technical debt
Zero migration	100% v2v & fixpack migration saving
Kubernetes optimized	Self-tuned optimal perf, production-ready,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

Just Enough Application Server

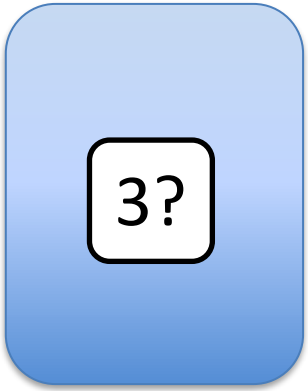
Open Liberty

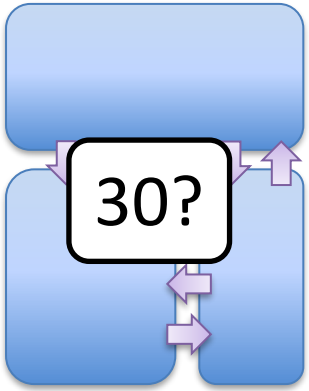
You control which features are loaded into each server instance

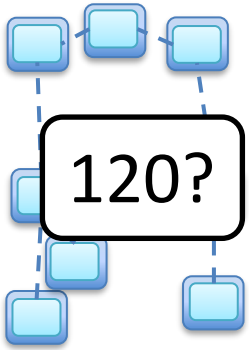

```
<feature>jsf-2.3</feature>
```

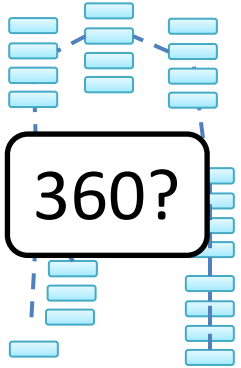


Granularity cost implications







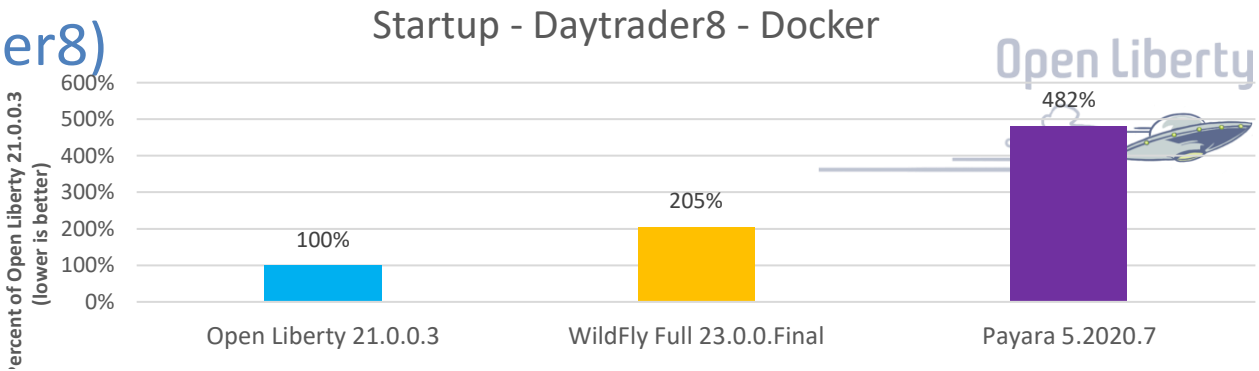


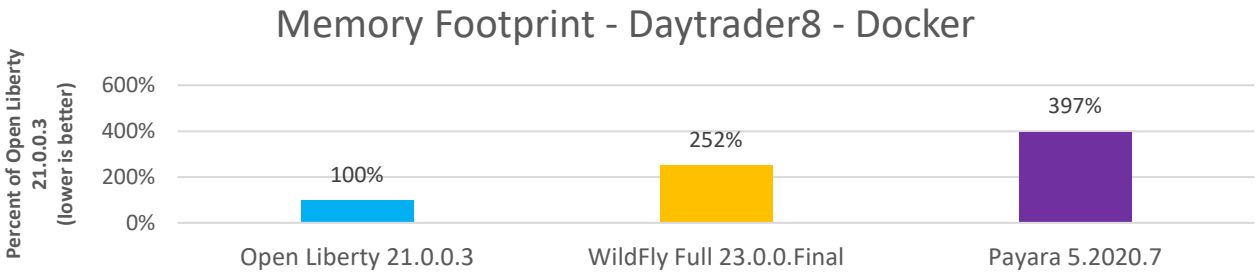
Disk?
Memory?
$$$?

X	X	X	X
200MB	200MB	200MB	200MB
=	=	=	=
600MB	6GB	24GB	72GB

Performance (Daytrader8)

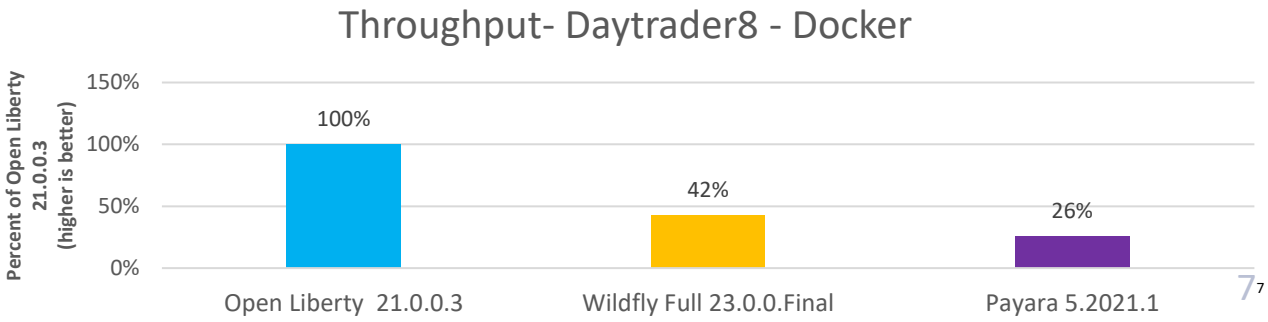
- Comparisons used each application server's Docker image
- Liberty outperforms others on all metrics for EE8 performance (startup time about half, throughput and memory footprint over 50% better)





System Configuration:

SUT: LinTel – Ubuntu 20.04.1 LTS, Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 4 cpus, 4GB RAM.
JDK version distributed with the docker images used for each server instance.



Low Operating Cost

Modernization led to optimized resource usage by **75%**

and reduced infrastructure footprint by **50%**

Major US healthcare provider

Cloud platforms shift responsibilities

Traditional Deployment

- I develop the app
- I give the Ops team a WAR file
- I *occasionally* update app dependencies

App

Server

Java

OS

VM

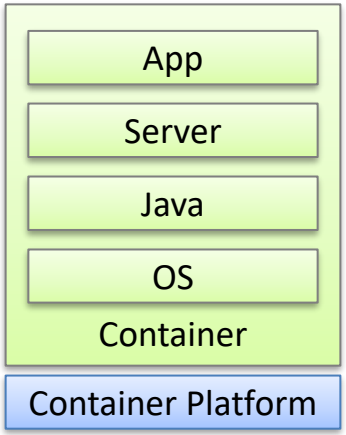
- I deploy the app
- I handle automation
- I monitor the app
- I maintain the infrastructure
- I apply important security fixes
- I plan and execute migrations

Cloud platforms shift responsibilities

Open Liberty

Cloud-native Platform Deployment

- I develop the app
- I *occasionally* update app dependencies
- I deploy the app
- I handle automation
- I monitor the app
- I maintain the infrastructure
- I apply important security fixes
- I plan and execute migrations



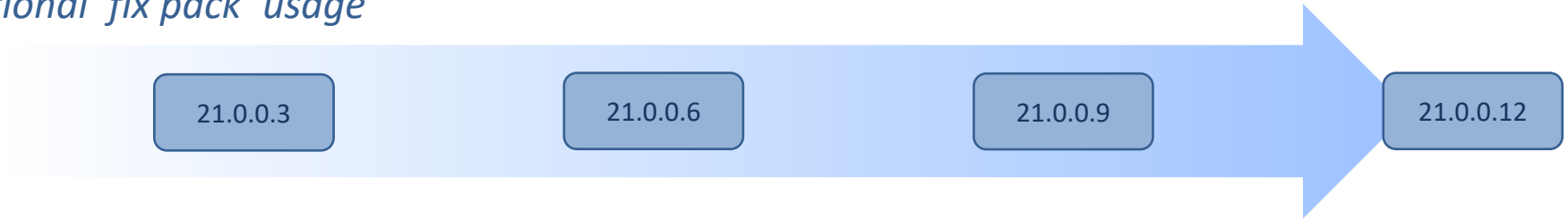
- I manage a cloud platform
- I provide services to the application teams

Liberty Release Cadence

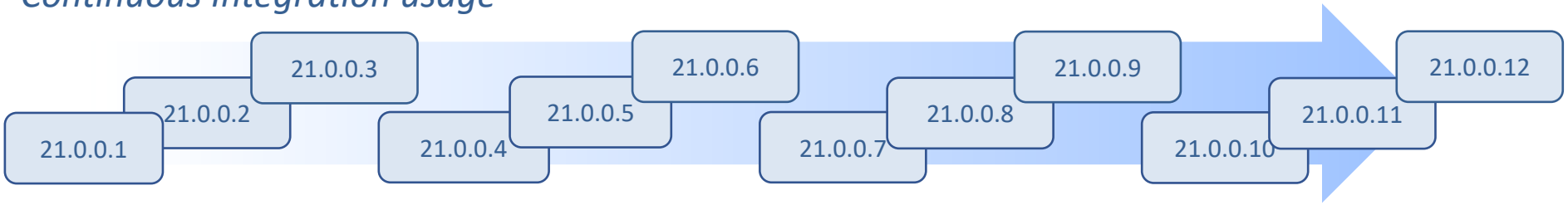
Liberty's 'zero migration' architecture makes picking up a new release simple

Skipping a release does not introduce migration work

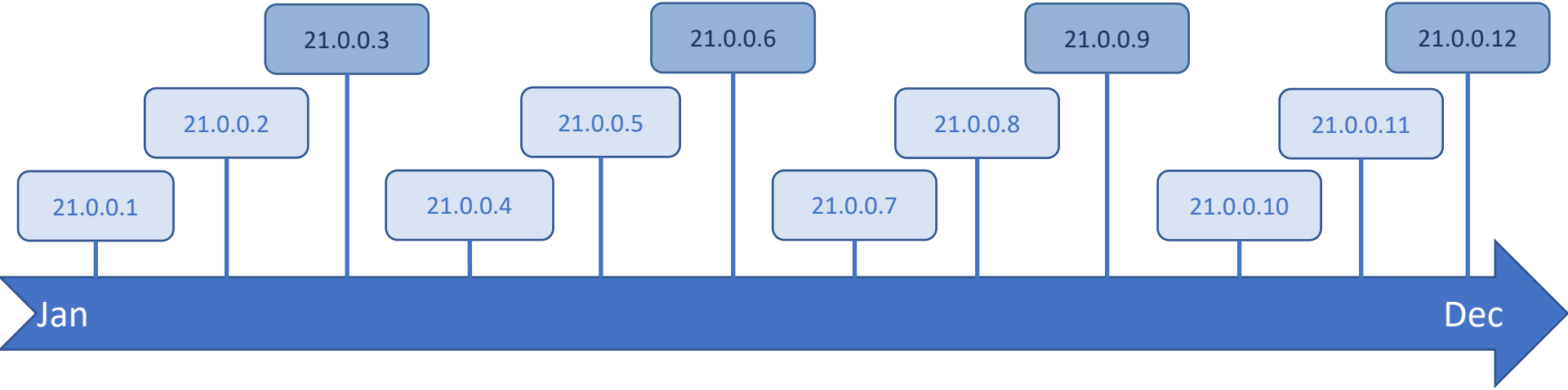
Traditional 'fix pack' usage



Continuous Integration usage



Liberty Release Cadence



	All CD releases	CD releases ending .3 .6 .9 .12
Support Provided	5 years	5 years
iFixes	24 weeks	2 years
Proactive Security iFixes	Most recent	Most recent 2

Zero Migration

- ✓No configuration behavior changes
- ✓No runtime feature behavior changes
- ✓No removals

Stay current with a rebuild
(no app or config changes necessary)

Skipping a release does not introduce additional migration work

Never apply an ifix again

Zero Migration

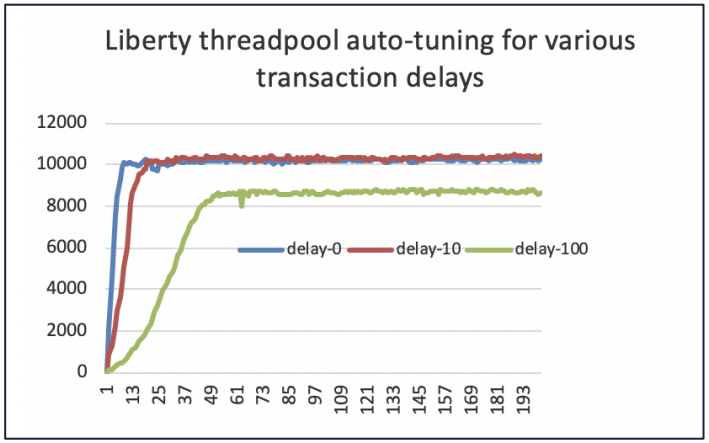
Today we migrated all our Liberty servers config from EE7 to EE8.
This process normally take 18 months in traditional WebSphere, cannot say how many manhours exactly.

Today it took 18 minutes, with Liberty's continuous delivery stream.

In July all apps starting to use EE8 framework.

Henrik Lundström, WAS Systems Administrator, Handelsbanken (Sweden)

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**

Kubernetes Optimized

“You don't have to tune thread pools. Liberty does an outstanding job”

WebSphere Technology Owner
Large health provider

Developer experience

Open Liberty

IDEs

Dev Mode

Repositories

The Central Repository

dockerhub

Build

Maven™

Gradle

APIs

Java EE™

JAKARTA EE

Spring Boot

Testing

JUnit

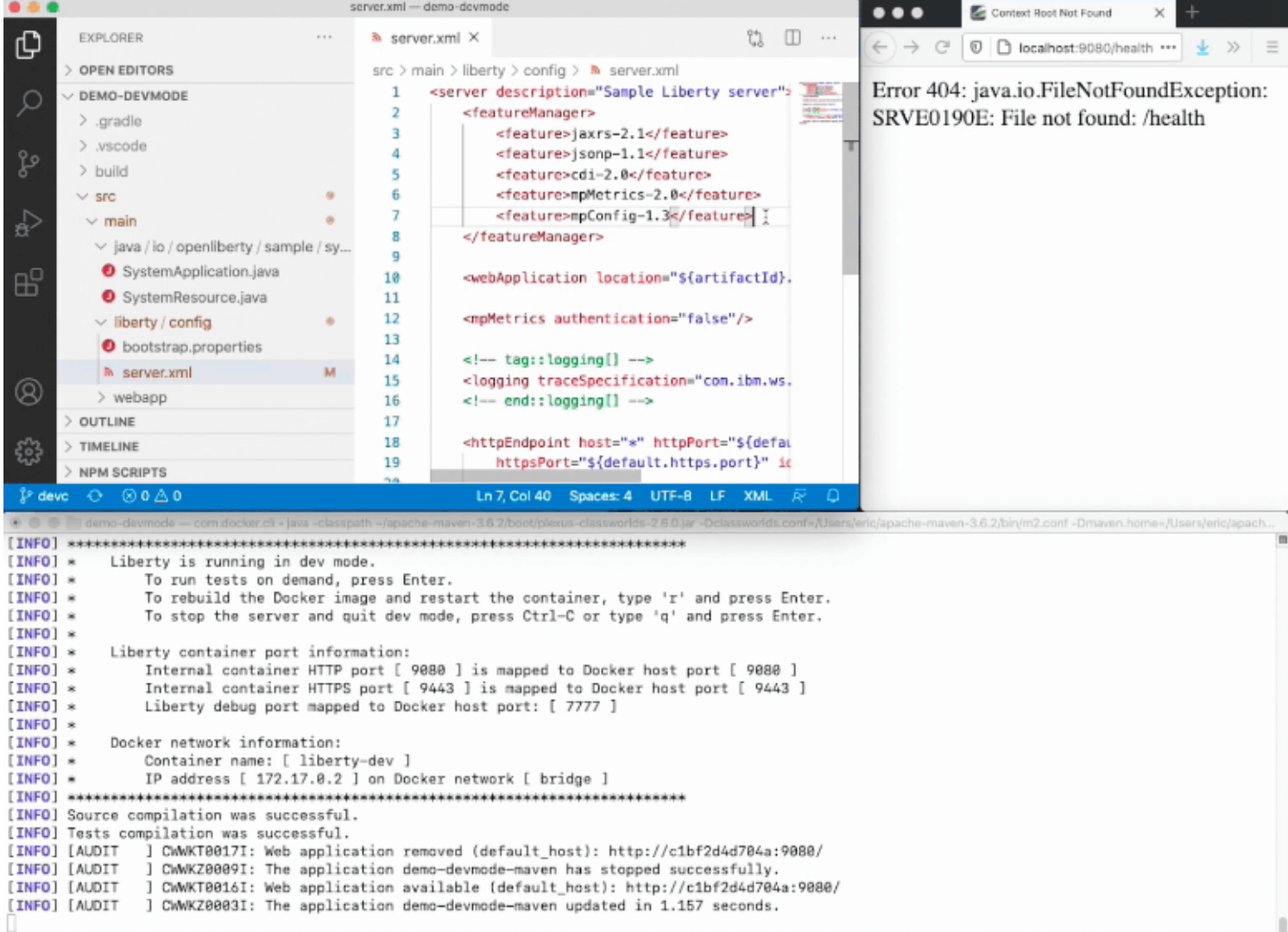
Jesse Gallagher
@Gidgerby

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!

Dev mode in action



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

2021 WebSphere Security Update

Jim Mulvey – April 2021

Agenda

- WebSphere and Open Liberty security, where are we?
- Recent new features and support
- Security future directions

WebSphere and Open Liberty security, where are we?

Open Liberty

Open Liberty & traditional WAS security

Feature / Capability	Liberty	traditional WAS
Minimal ports opened	Yes	No
Federated user registries	Yes	Yes
OAuth, OpenID Connect Client	Yes	Yes
OpenID Connect Provider	Yes	No
LTPA	Yes	Yes
SPNEGO tokens	Yes	Yes
SAML Web SSO	Yes	Yes
JSON web tokens (JWTs)	Yes	Yes
User and group API	Yes (w/SCIM)	Yes (w/VMM)
Auditing	Yes	Yes
Certificate management built-in	No	Yes
Java EE 8 Security 1.0	Yes	No
Multi-realm support	Yes (w/collectives, 1 realm per server)	Yes

Open Liberty & traditional WAS security (2)

Feature / Capability	Liberty	traditional WAS
Local OS registry	Yes (z/OS SAF only)	Yes
Kerberos native support	Limited to use w/SPNEGO	Yes
Kerberos for database connections	Yes (20.0.0.11)	Yes
Fine grained roles	Admin/Reader	Yes
Kerberos for LDAP connections	Beta (21.0.0.2)	Yes (1Q21)
TLS 1.3 support	Yes (19.0.0.3)	Yes (w/ IBM JDK 8 1Q21)
Microprofile JWT (mpJWT 1.2)	Yes (21.0.0.3)	No
Jakarta EE 9 Security 2.0	Beta (21.0.0.3)	No
LetsEncrypt (ACME CA) support	Yes (20.0.0.10)	No
JWE (JSON Web Encryption for JWTs)	Yes (w/mpJWT 1.2 21.0.0,3)	No (pending)
DISA Security Technical Impl Guide (STIG)	No (pending)	Yes
OpenShift OAuth Token support	Yes (20.0.0.9)	No
OpenShift Operator security integration	Yes	No

Recent activity

Recent new features and support

Open Liberty

Recently delivered support: 1Q2020

Security

- tWAS LDAP login using Kerberos
- Liberty mpJWT 1.2
- tWAS TLS 1.3 support with JDK 8
- Jakarta EE 9 Security 2.0

Dev Exp

Foundation

API

Orchestration

Recently delivered support 2Q & 3Q 2020

Security

- Support for OpenShift service account credentials
- Simplify config for RA deployed JAAS Login Modules
- API to retrieve userid from LTPA Token
- Open Liberty operator OpenShift certificate manager and SSO delegation

Dev Exp

- OpenAPI to MP REST Client code gen
- MicroShed Testing support for Kafka
- MicroShed Testing simplified config
- Guide for using MicroProfile Reactive Messaging
- Guide for testing apps using reactive messaging

Foundation

- Automatic http response compression
- Yum/apt-get install support
- featureUtility - provision Liberty features from maven repo
- Control application startup order

API

- MicroProfile 3.3
- Java SE 14
- MicroProfile GraphQL

Orchestration

- Grafana dashboard for metrics
- Log format improvements
- Persistent EJB timer failover in OpenShift

Recently delivered support: 4Q2020

Security

- Support for ACME cert management and LetsEncrypt
- Liberty database login using Kerberos
- Enforce cipher order with TLS

Dev Exp

- Liberty dev mode for containers
- Liberty Maven Tools 3.3
- Liberty Gradle Tools 3.1
- Open Liberty VS Code Tools includes MicroProfile Tools

Foundation

- Support for traditional WebSphere style EJB names

API

Orchestration

- Docker images using minimal kernel
- Red Hat DataGrid for HTTP Session replication

Jakarta EE 9 Beta Liberty 21

Late last year, the Jakarta EE community released the next step in the evolution of the Jakarta EE specification. Version 9.0 of the Jakarta EE specification focuses on renaming packages and namespaces by changing to *jakarta* prefixed packages names in the API and to *the jakarta.ee namespace* in the deployment descriptor files. Additionally, the specification makes optional or removes older functions that have been deprecated or stabilized for many years.

Over the last year, the Open Liberty development team worked to implement the Jakarta EE 9 specification as it was being finalized. To avoid having to duplicate the code just to change the API package names, the team developed an open source byte code transformation technology. With this technology, Jakarta EE 8 compatible code is transformed to work with Jakarta EE 9 APIs by manipulating the byte code to convert class and property reference prefixes from javax to jakarta.

As of the 21.0.0.2-beta release, Open Liberty is the first vendor product to be Jakarta EE Web Profile 9.0 compatible. With the recent 21.0.0.3-beta release, Open Liberty is the first vendor product to be added to the Jakarta EE Platform 9.0 compatibility list. These releases continue to demonstrate Open Liberty's leadership in the Jakarta EE technologies. With the upcoming Jakarta EE 9.1 release that adds Java 11 support to the TCK, I anticipate that Open Liberty will be added to the compatibility list for the 9.1 release when it is available since Open Liberty has supported Java 11 for the last 2 years.

```
For more see: https://openliberty.io/blog/2021/03/05/jakarta-ee-9-compatibility.html
```

Jakarta EE 9 Beta Security 2

Jakarta Security 2.0

The new `appSecurity-4.0` feature has been created to support Jakarta EE 9 and introduces Jakarta Security 2.0 support. The `appSecurityClient-1.0`, `audit-1.0`, `constrainedDelegation-1.0`, `jcaInboundSecurity-1.0`, `jwt-1.0`, `passwordUtilities-1.0` and `spnego-1.0` features have been updated to also support Jakarta EE 9.

The `appSecurity-4.0` feature must be added to `server.xml` when using application security with Jakarta EE 9. The other Liberty security features will automatically adapt to the level of Java EE or Jakarta EE that is already in use, so no change is needed when using them with Jakarta EE 9.

The following features can be included in your server.xml:

```
<featureManager>
      <feature>appSecurity-4.0</feature>
      <feature>appSecurityClient-1.0</feature>
      <feature>audit-1.0</feature>
      <feature>constrainedDelegation-1.0</feature>
      <feature>jcaInboundSecurity-1.0</feature>
      <feature>jwt-1.0</feature>
      <feature>passwordUtilities-1.0</feature>
      <feature>spnego-1.0</feature>
</featureManager>
```

: https://jakarta.ee/specifications/security/2.0/

Additional information about the Jakarta Security 2.0 specification can be found here.

https://openliberty.io/blog/2021/01/26/ee9-messaging-security-21002-beta.html

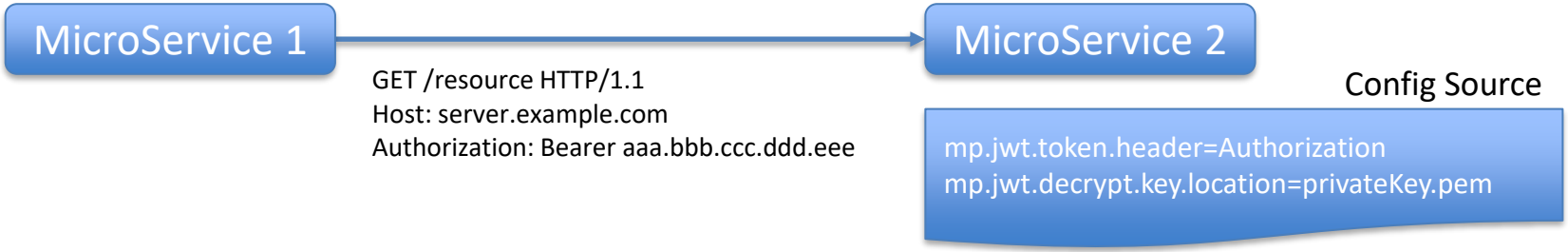
MicroProfile JWT 1.2

```
<feature>mpJwt-1.2</feature>
```

Specification defines how to use a JSON Web Token (JWT) for authenticating and authorizing requests to a service

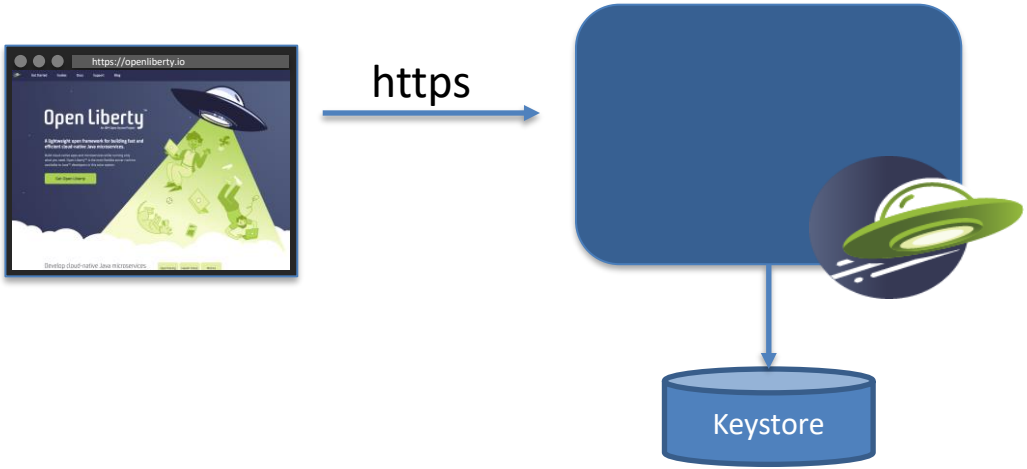
Latest version of the spec introduces:

- New MicroProfile Config properties
- Support for JSON Web Encryption (JWE) tokens
- Support for new signature algorithms

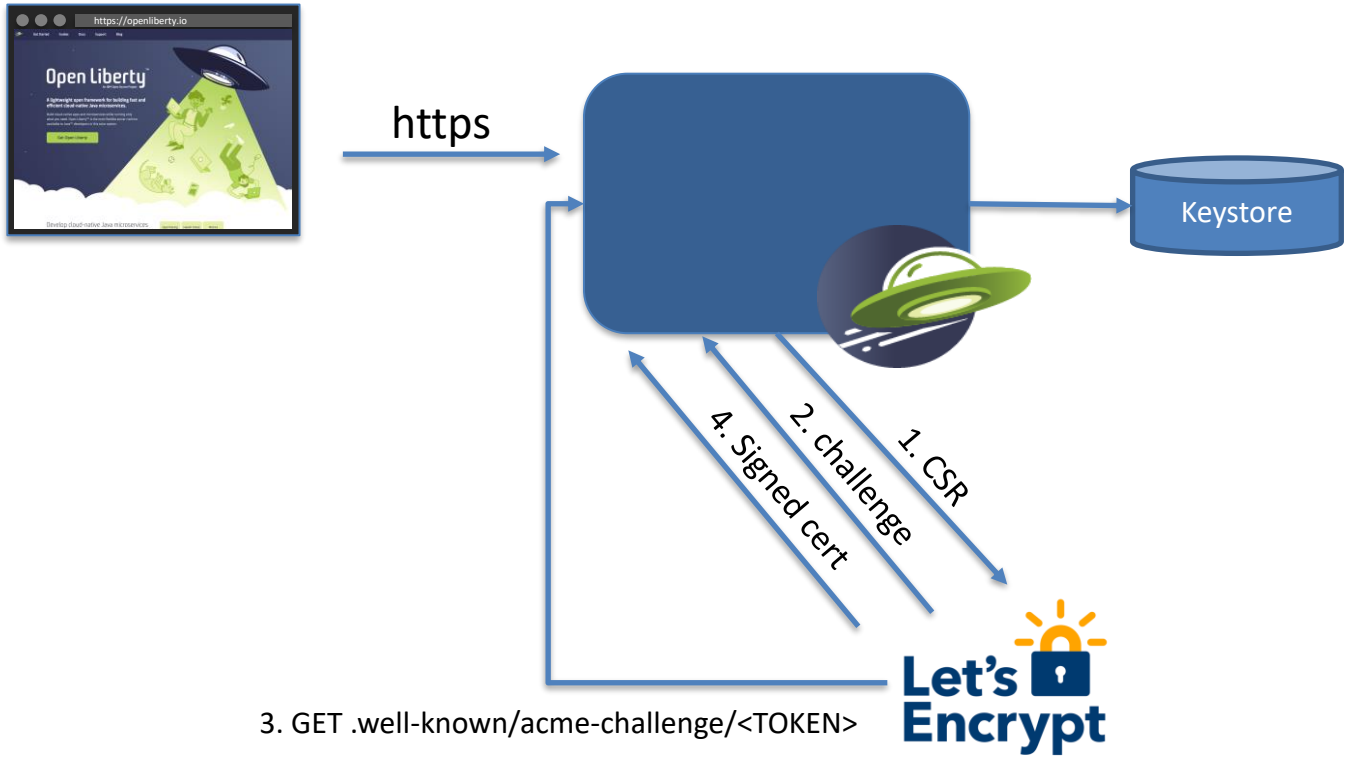


For more see: https://openliberty.io/docs/21.0.0.3/reference/feature/mpJwt-1.2.html

Automatic Certificate Management API



Automatic Certificate Management API



ACME Liberty server config

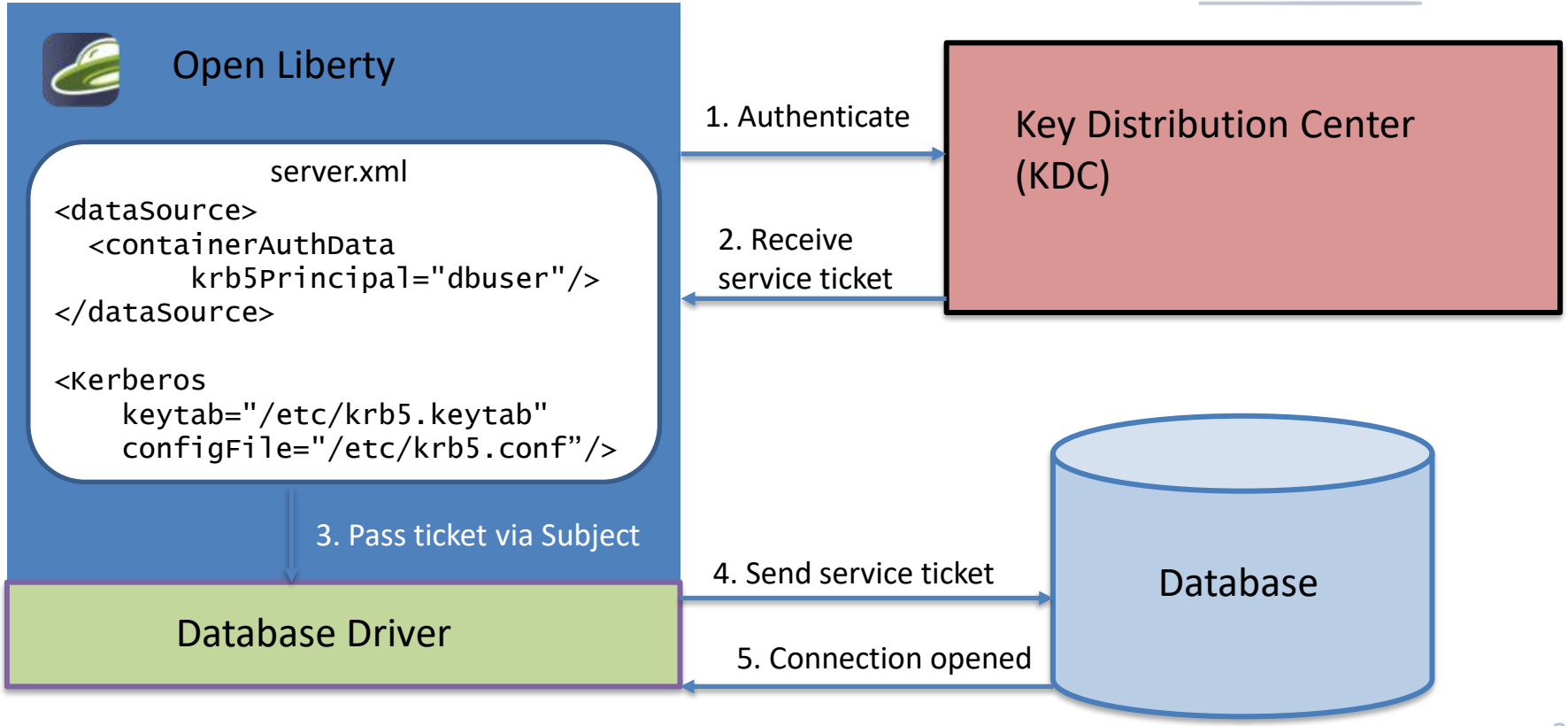
Use with LetsEncrypt

```
<featureManager>
  <feature>acmeCA-1.0</feature>
</featureManager>

<acmeCA directoryURI="https://acme-v02.api.letsencrypt.org/directory"
        renewBeforeExpiration="30d">
    <domain>my_host_name.domain.com</domain>
</acmeCA>
```

Use

```
<featureManager>
  <feature>acmeCA-1.0</feature>
</featureManager>

<acmeCA directoryURI="https://myca.mycompany.com/directory" >
    <domain>my_host_name.mycompany.com</domain>
</acmeCA>
```

Liberty Kerberos authentication to Database



Config for Kerberos Authentication

Keytab Search order

1. server.xml
2. <user.home>/krb5.keytab

Credential Cache Search order

1. server.xml
2. KRB5CCNAME environment variable
3. /tmp/krb5cc_<uid>
4. <user.home>/krb5cc_<user.name>

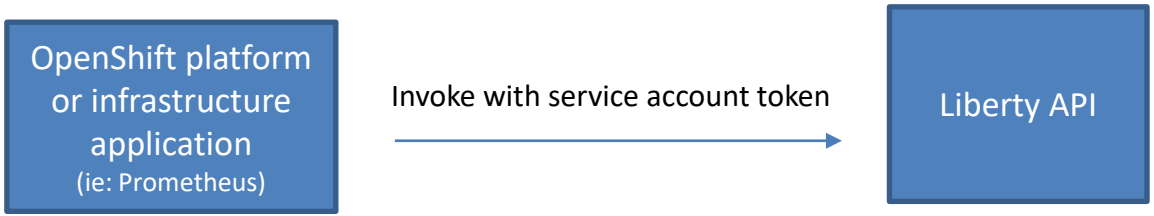
Server.xml :

```
<containerAuthData id="krb"
                   krb5Principal ="dbuser"
                   krb5TicketCache= "/home/user/krb5cc"/>

<kerberos keytab ="/etc/krb5.keytab"/>

<dataSource jndiName="db/myDatabase"
            containerAuthDataRef="krb"
</dataSource>
```

OpenShift Support for Service Account tokens

Open Liberty

In OpenShift environments for service-to-service authentication Service Account(SA) tokens are common-place for authentication flows. Open Liberty now can consume SA tokens and will authenticate with Kubernetes and provide automatic role mapping.



Benefits of SA Tokens

- No real user password required for authentication.
- Not tied to specific user, rather a service.
- Authentication by service account's API token.
- Service account can be revoked or reset without impacting user password.
- Authorization can be done by service account name, or service account project name (namespace).

OpenShift Support for Service Account tokens

Typical configuration for login with “OpenShift service account” when SA token passed

```
<okdServiceLogin
  userValidationApi="https://api.example.com:6443/apis/user.openshift.io/v1/users/~“>
</okdServiceLogin>
```

Minimum configuration (if Liberty is inside the same cluster)

```
<okdServiceLogin />
```

Service Account tokens Liberty usage sample

Service account's user name: `system:serviceaccount:project01:robot`

- SA Token comes into Liberty as an HTTP authorization header
  (ie: Authorization: Bearer <serviceAccountToken here>)

The following API call will be authorized automatically:

```
@GET
@RolesAllowed("project01")
public String returnBook(@PathParam("id") int id) { ...}

The following API call need application-bnd to map role to
openshift project name,  or user in an OpenShift project
@GET
@RolesAllowed("author")
public String returnBook(@PathParam("id") int id) { ...}
```

Server.xml snippet:

```
<application-bnd>
      <security-role name="author">   <!—you only need one of following
           <group name="monitors" access-id="group:<realm name here>/project01 />
           <user name="monitors" access-id="user:<realm name here>/project01:robot />
      </security-role>
</application-bnd>
```

OpenShift Open Liberty Operator & Security

Open Liberty operator watches Liberty pods in OpenShift

Operator capabilities

The Open Liberty Operator has a capability level of five, which means that it has the highest level of enterprise capabilities, including the following capabilities:

- **High availability that's provided by horizontal auto-scaling**

  You can configure horizontal auto-scaling to create and delete instances of your application based on resource consumption. This ability to run multiple instances of your application and auto-scale them means that your application is made highly available.

- **Enhanced deployment management**

  With the Open Liberty Operator, you can more easily manage applications that are deployed to Kubernetes. For example, in the operator deployment file, you can specify an image stream in the `applicationImage` field. Then, after you upload a new container tag for a new version of an application, the operator updates the application on a rolling basis.

- **Automated service binding**

  The operator automates updates of binding information among applications, meaning that it connects applications and maintains information about whether a particular application produces or consumes a service. With this information, the operator automatically handles Kubernetes-level details, including creating and injecting Kubernetes Secrets, so that your applications can connect to required services without interruption.

- **Single sign-on (SSO) delegation**

  With Open Liberty, you can delegate SSO authentication to external providers. The Open Liberty Operator enables easier configuration and management of SSO information for your applications.

https://openliberty.io/docs/21.0.0.3/open-liberty-operator.html#_operator_capabilities

OpenShift Open Liberty Operator & Security (2)

Open Liberty operator watches Liberty pods in OpenShift

- **OpenShift certificate management integration**

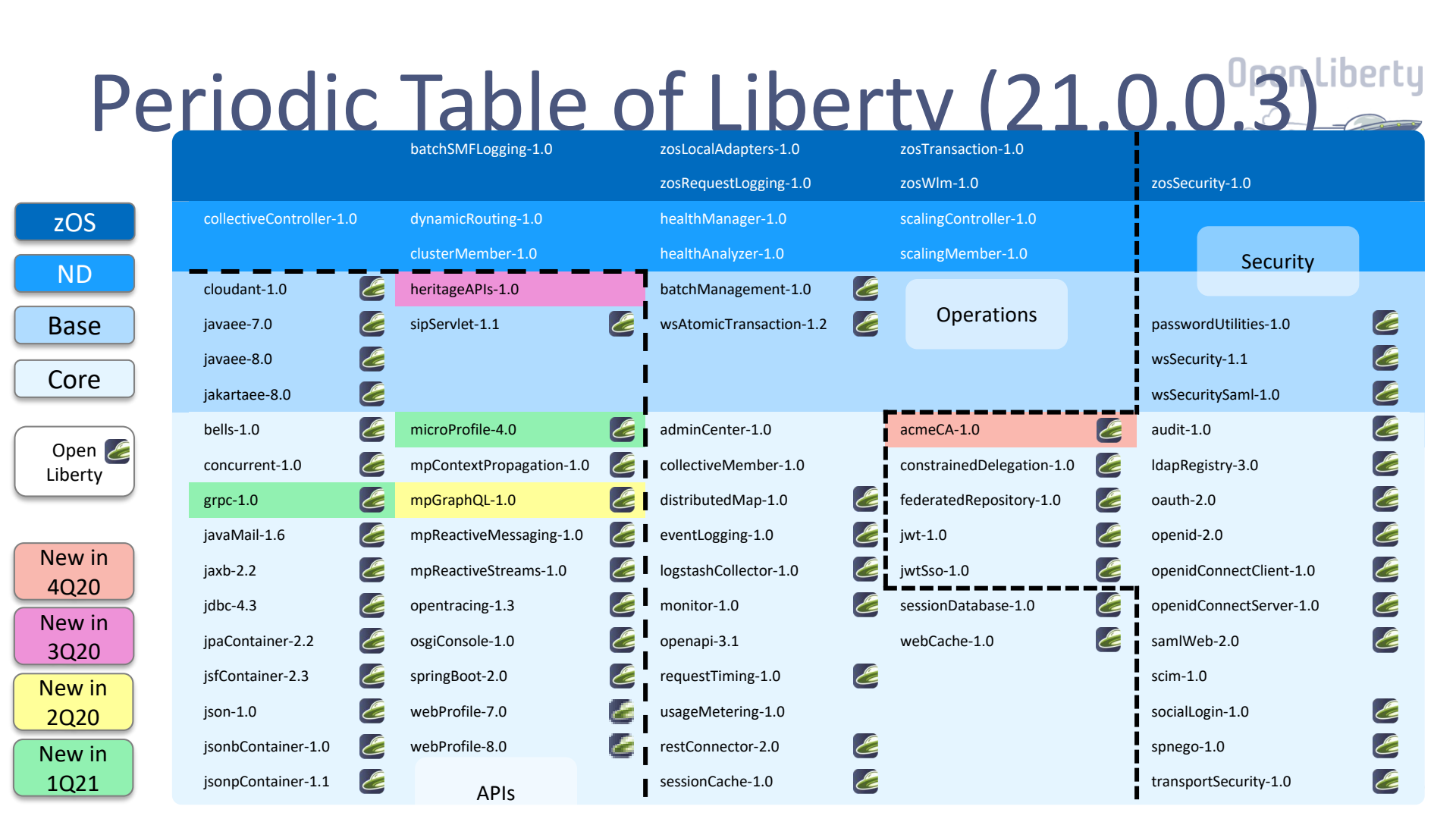
  The operator takes advantage of the cert-manager tool, if it's installed on the Kubernetes cluster. The cert-manager tool allows the operator to automatically provision Transport Layer Security (TLS) certificates for pods and routes. Certificates are mounted into containers from a Kubernetes Secret so that the certificates are automatically refreshed when they're updated.

https://openliberty.io/docs/21.0.0.3/open-liberty-operator.html#_operator_capabilities

Recent Updates

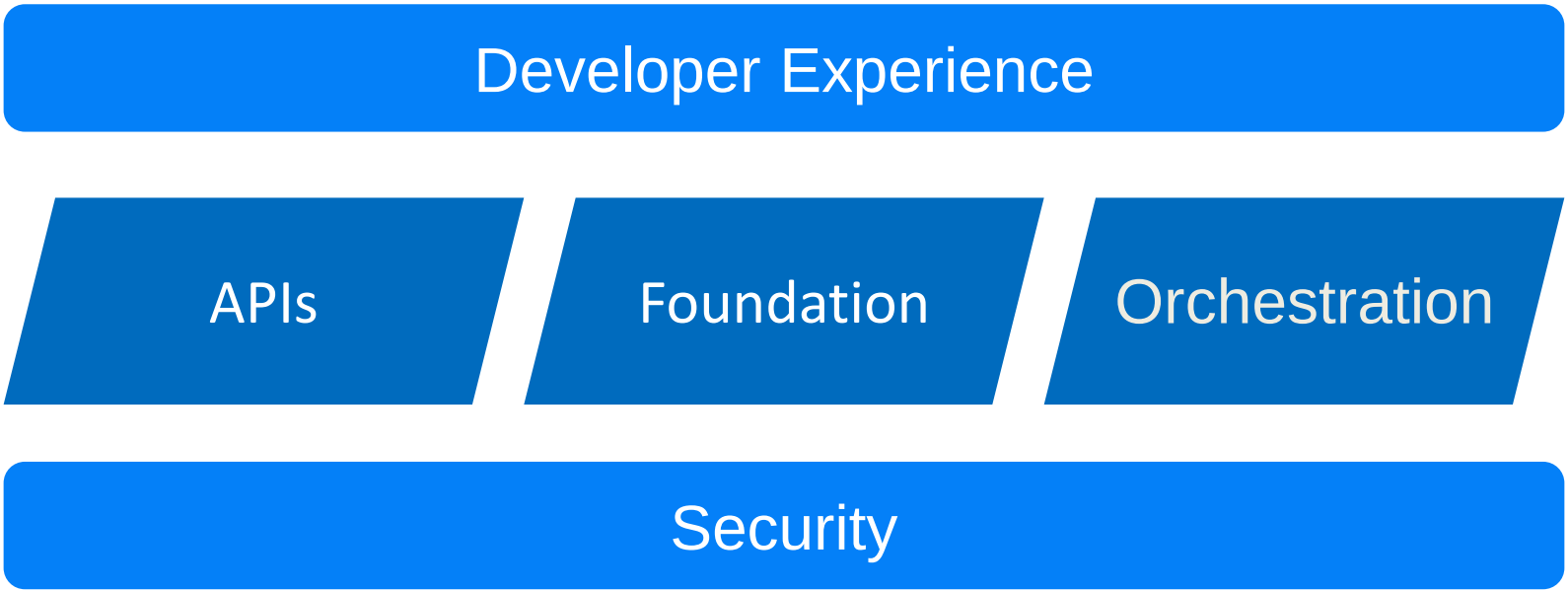
Open Liberty

Periodic Table of Liberty (21.0.0.3)



Focus areas

Open Liberty



Liberty Last Quarter Review

Security

- JSON Web Encryption
- JWT nbf Claim
- WASReqURLOidc domain name validation
- MicroProfile JWT 1.2

Dev Exp

Developer Tools

- Arquillian Liberty Plugin Supports Jakarta EE 9

Guides

- Contract Testing
- Kubernetes Operator
- MongoDB

Migration Tools

- Java SE 15 Migration Advice
- Open Liberty as Migration Target

Foundation

- Override context-root from application.xml
- Trusted Sensitive Header Origin as IP segment

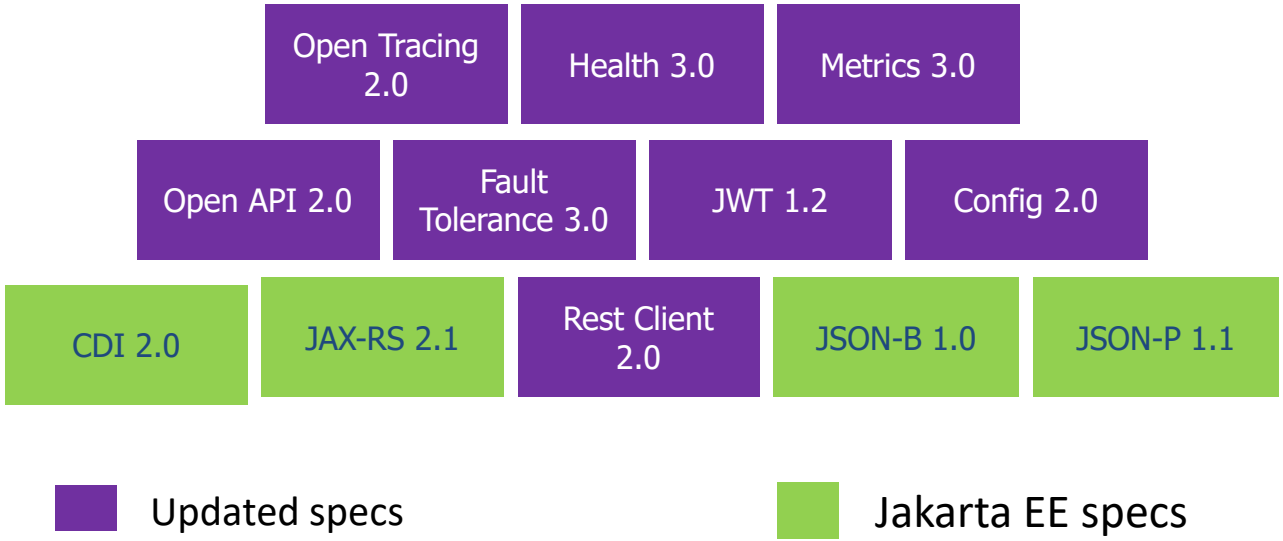
API

- MicroProfile 4.0
- gRPC

Orchestration

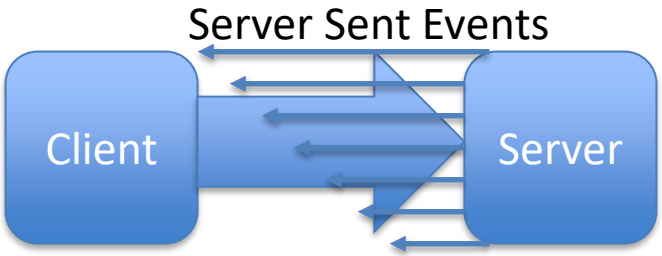
MicroProfile 4.0

Open Liberty



MicroProfile Rest Client - SSE

Open Liberty



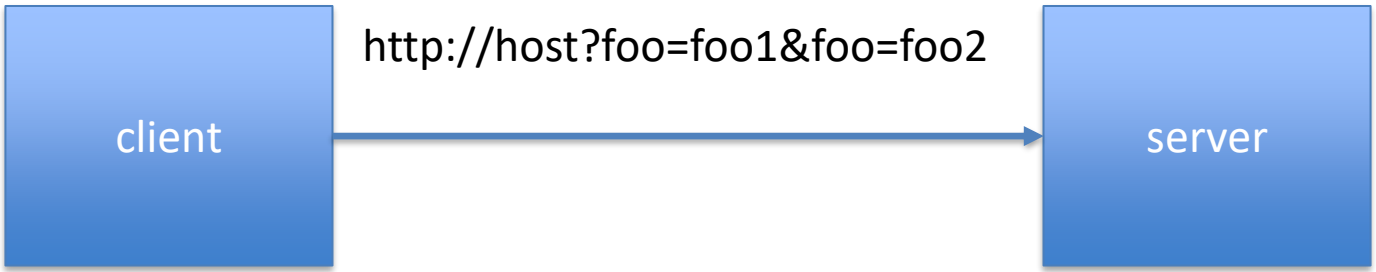

```
public interface SseClient {
  @GET @Path("ssePath")
  @Produces(MediaType.SERVER_SENT_EVENTS)
  Publisher<InboundSseEvent> getEvents();
}
```

```
Publisher<InboundSseEvent)> publisher = Client.getEvents();
publisher.subscribe(new MyInboundSSESubscriber());
```

MicroProfile Rest Client

Open Liberty

```
@GET
@Path(“/”)
@Produces(MediaType.APPLICATION_JSON)
Public Data get(@QueryParam(“foo”)Set<String> foo);
```



MicroProfile Config 2.0

- Variable Replacement

```
server.url=https://${host}:${port}
host=my.server.com
port=443
```

- Config Prefix

```
@ConfigProperties(prefix="server")
public class MyClass {
  @Inject @ConfigProperty("url")
  private String url;
}
```

MicroProfile Config Profiles


```
@ConfigProperties(prefix="server")
public class MyClass {
  @Inject @ConfigProperty("url")
  private String url;
}
```

```
server.url=https://${host}:${port}
%test.host=my.test.server.com
%test.port=9443
%prod.host=my.real.server.com
%prod.port=443
```

```
server.url=https://${host}:${port}
host=my.test.server.com
port=9443
```

```
server.url=https://${host}:${port}
host=my.real.server.com
port=443
```

mp.config.profile=test

test

mp.config.profile=prod

prod

MicroProfile JWT 1.2 - JWE

Base64 URL encoded Header . Base64 URL encoded Payload . Base64 URL encoded Signature

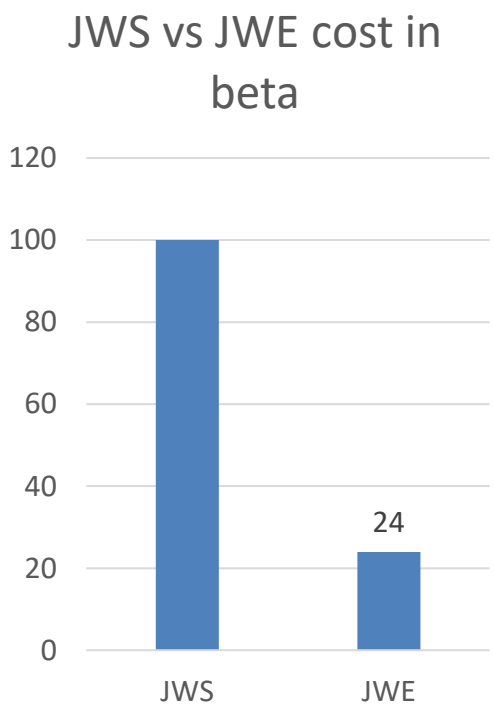
```
{
  "typ":"JWT",
  "alg":"RS256"
}
```

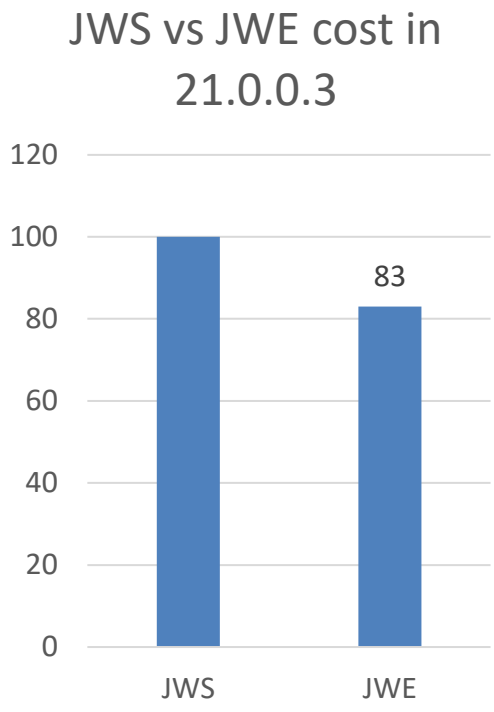
```
{
  "aud":"server",
  "iss":"https://ibm.com/oidc/endpoint/OP",
  "iat":1605792600,
  "exp":1605799800,
  "sub":"jdoe",
  "email":"jdoe@ibm.com"
}
```

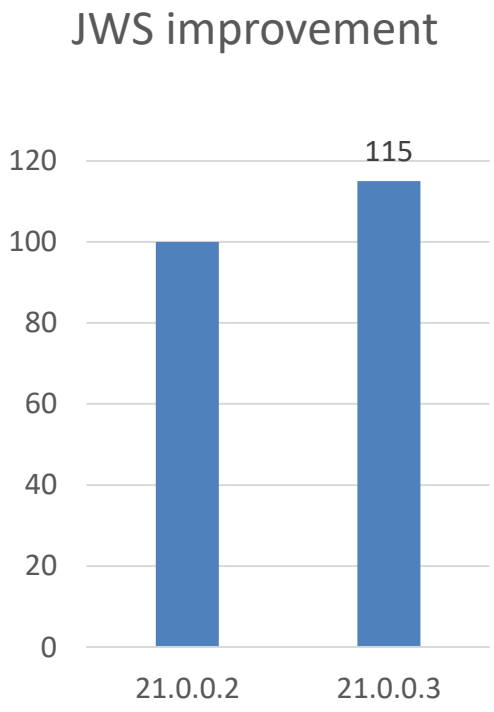
JSON Web Encryption (JWE)

Header . Encrypted key . Initialization vector . Ciphertext (encrypted JWS) . Authentication tag

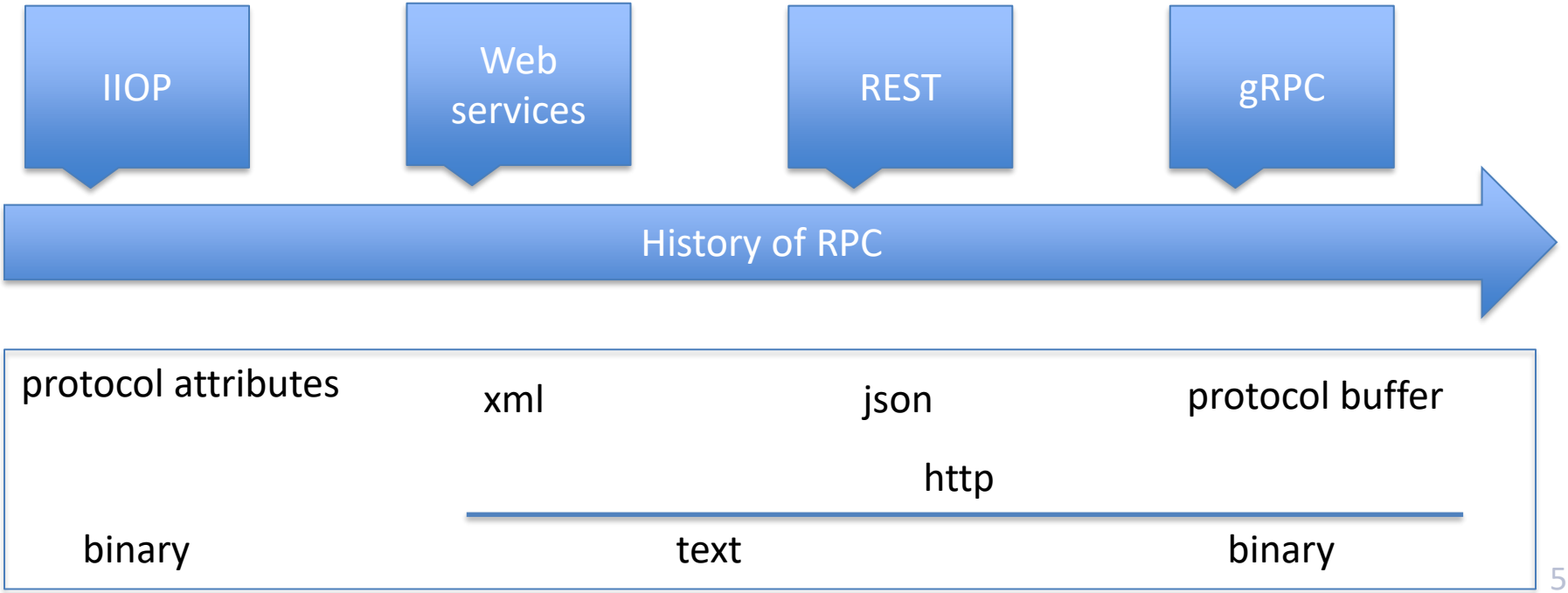
MicroProfile JWT Performance







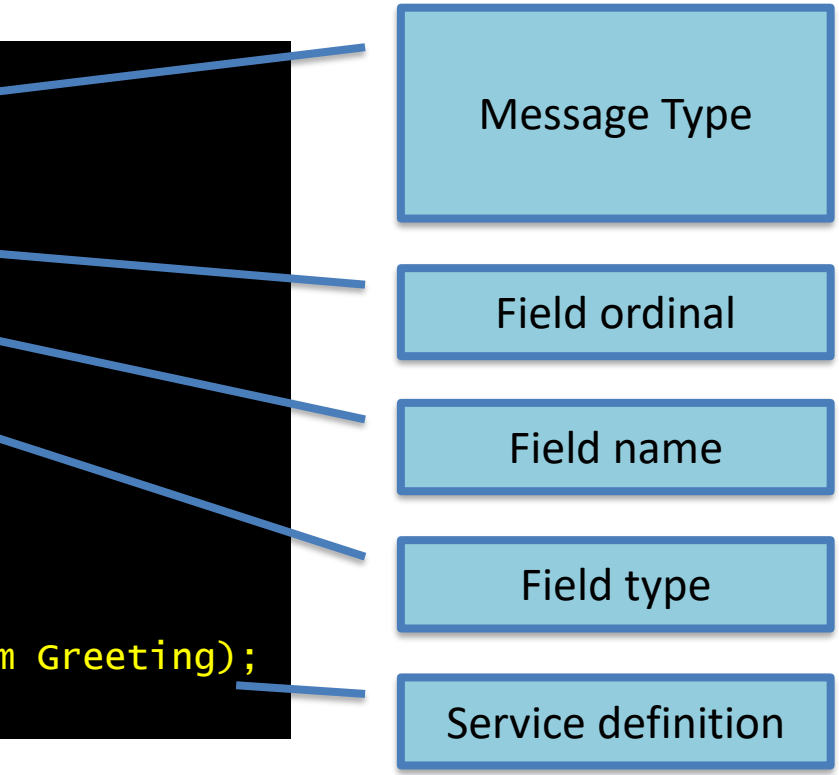
gRPC – What is it



gRPC – Protocol Buffers

Open Liberty

```
syntax = “proto3”

message User {
  string name = 1
  string occupation = 2
  age = 3
}

Message Greeting {
  string message = 1
}

Service GreetingService {
  rpc greetUser (User) returns (stream Greeting);
}
```



gRPC – A service


```
public class GreetingService
             extends GreetingServiceGrpc.GreetingServiceImplBase {

  public void greetUser(User user, StreamObserver<Message> response) {
    String message = “Hello “ + user.getName();
    Greeting greeting = Greeting.newBuilder().setMessage(message).build();
    response.onNext(greeting);

    message = “Bonjour “ + user.getName();
    greeting = Greeting.newBuilder().setMessage(message).build();
    response.onNext(greeting);

    response.onCompleted();
  }
}
```

gRPC - Why

Speed

- gRPC sends less data
- Messages are fast to parse
- Efficient HTTP usage

gRPC - Why

Speed

- gRPC sends less data
- Messages are fast to parse
- Efficient HTTP usage

```
Message Greeting {
  string message = 1
}
```

```
Greeting.newBuilder()
    .setMessage("Hello World")
 .build();
```

gRPC - Why

Speed

- gRPC sends less data
- Messages are fast to parse
- Efficient HTTP usage

```
Message Greeting {
  string message = 1
}
```

```
Greeting.newBuilder()
    .setMessage("Hello World")
 .build();
```

```
{
    message: "Hello World"
}
```

JSON encoding

```
7b 0a 20 20 20 20 6d 65 73 73 61 67
65 3a 20 22 48 65 6c 6c 6f 20 57 6f 72
6c 64 22 0a 7d 0a
```

JSON on the wire

```
12 11 [48 65 6c 6c 6f 20 57 67 72 6c
64]
```

protocol buffer on the wire

gRPC - Why


```
Message Greeting {
  string message = 1
  string message2 = 2
}
```

```
Greeting.newBuilder()
    .setMessage("Hello World")
    .setMessage("Hello World")
 .build();
```

```
{
    message: "Hello World"
    message2: "Hello World"
}
```

JSON encoding

7b 0a 20 20 20 20 6d 65 73 73 61 67
65 3a 20 22 48 65 6c 6c 6f 20 57 6f 72
6c 64 22 0a 20 20 20 20 6d 65 73 73
61 67 65 32 3a 20 22 48 65 6c 6c 6f 20
57 6f 72 6c 64 22 0a 7d 0a

JSON on the wire

12 11 [48 65 6c 6c 6f 20 57 67 72 6c
64] 13 11 [48 65 6c 6c 6f 20 57 67 72
6c 64]

protocol buffer on the wire


Jakarta EE 9
In beta
Open Liberty

Jakarta EE 9 Final Release

JAKARTA™ EE

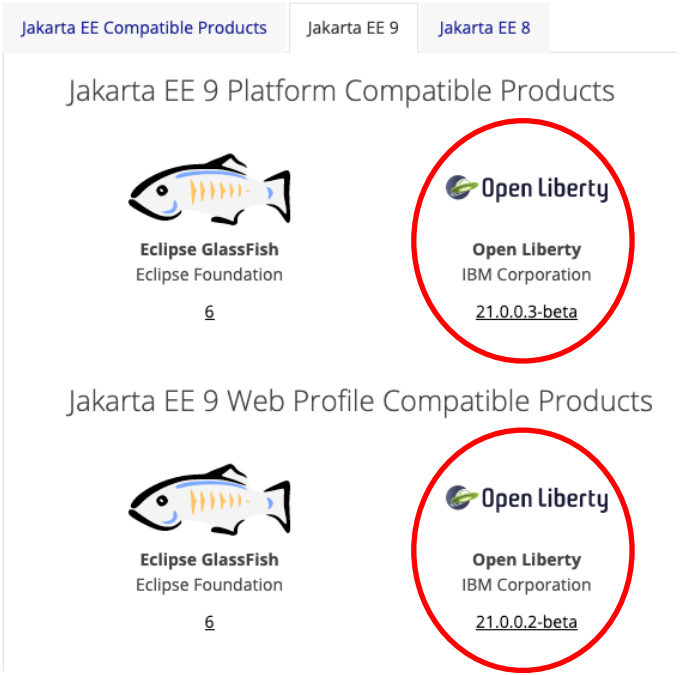
Jakarta EE 9 – Nov 20, 2020

- Final Specifications for ALL Projects
  - https://jakarta.ee/specifications/
- Final APIs for ALL Projects
  - https://mvnrepository.com/artifact/jakarta
- Final TCKs for ALL Projects
  - https://download.eclipse.org/jakartaee/
- Final Compatible Implementation(s)
  - Glassfish 6.0.0 for Platform Specification Certification
  - https://jakarta.ee/compatibility/

JakartaOne LiveStream – Dec 08, 2020

- https://jakartaone.org/2020/
- Formal announcement of Jakarta EE 9

Jakarta EE 9 Compatible Products



https://jakarta.ee/compatibility/#tab-9

- First Vendor Compatible Implementation!
- Working through plans for GA

Labs, Questions

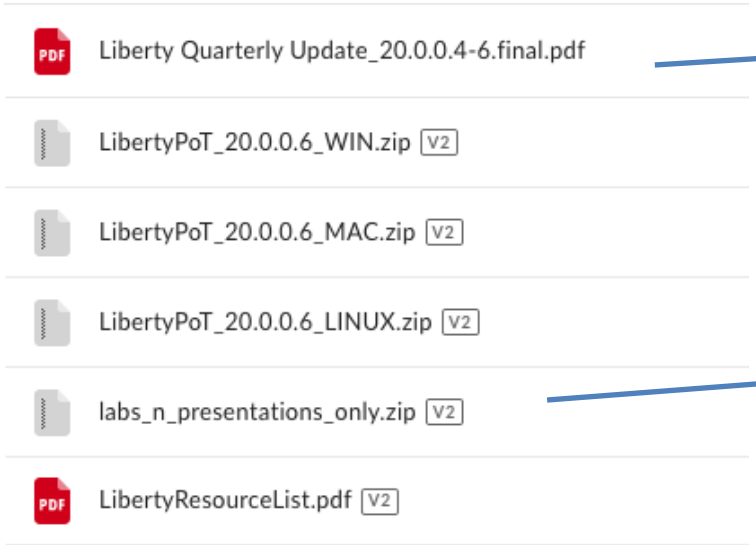
Open Liberty

WebSphere Liberty Virtual POT

Download the operating system specific content zip file from either

https://ibm.box.com/WASLibertyVPoT (fast - about 10 minute download)

https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/pot/ (slower but firewall friendly – about 1 hour download)



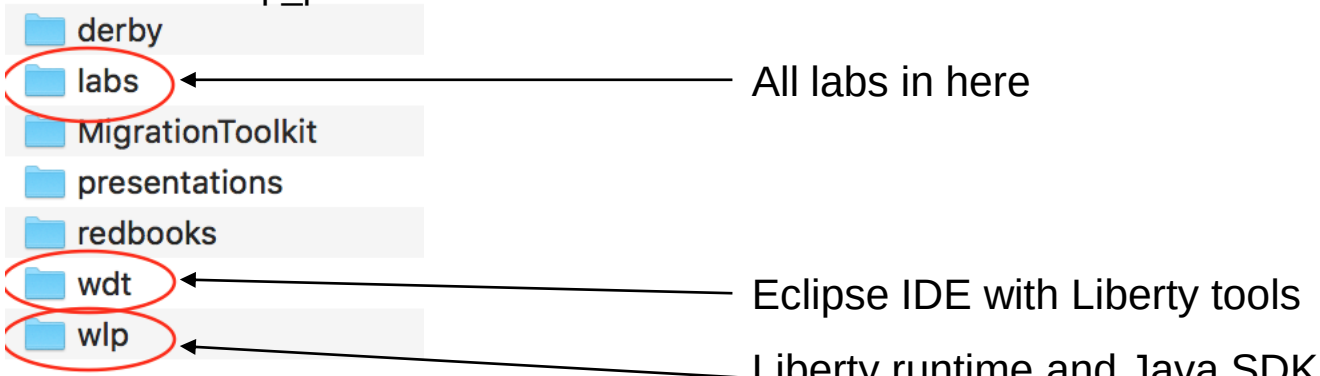
charts only

charts & lab instructions only

WebSphere Liberty Virtual POT

Unzip to C:\wlp_pot

- Note: You can unzip to anywhere you wish but the lab instructions assume the unzip location is C:\wlp_pot



Follow labs\gettingStarted\0_setup_**20180105**\setup.pdf

Then choose any labs you want to do

Open Liberty Guides

- Hands-on learning in ~20 minutes
- 52 guides
  - MicroProfile & Jakarta EE
  - Open Shift, Docker, Kubernetes Istio
- Latest Guides
  - *Authenticating users through social media providers*
  - *Deploying microservices to OpenShift by using Kubernetes Operators*

https://openliberty.io/guides

Resources

Useful Liberty Links

- Why choose Liberty for Microservices: https://ibm.biz/6ReasonsWhyLiberty
- Choosing the right Java runtime: https://ibm.biz/ChooseJavaRuntime
- How to approach application modernization: https://ibm.biz/ModernizeJavaApps
- Open Liberty: https://www.openliberty.io
- Open Liberty Guides: https://www.openliberty.io/guides

Programming API Links

- Eclipse MicroProfile: https://microprofile.io
- Jakarta EE: https://jakarta.ee

Support Links

- Java support dates: http://www.ibm.com/developerworks/java/jdk/lifecycle
- Single Stream Continuous Delivery: https://www-01.ibm.com/support/docview.wss?uid=ibm10869798
- Container Support Policy: https://www.ibm.com/support/pages/container-deployment-support-policy-websphere-liberty

Migration Tools

- IBM Transformation Advisor http://ibm.biz/cloudta
- WebSphere Binary Migration Toolkit: http://ibm.biz/WAMT4AppBinaries

Resources

Red Hat UBI images

- https://hub.docker.com/r/ibmcom/websphere-liberty
- https://hub.docker.com/r/openliberty/open-liberty

Ubuntu images

- https://hub.docker.com/_/websphere-liberty
- https://hub.docker.com/_/open-liberty

IBM Container Registry images

- https://cloud.ibm.com/docs/Registry?topic=RegistryImages-ibmliberty

Configuration/build files in github

- https://github.com/WASdev/ci.docker
- https://github.com/OpenLiberty/ci.docker

Next Quarterly Update

Liberty 21.0.0.1-3 Update

~~Session#1: April 14, 2021 from 1-3pm ET - http://ibm.biz/Liberty-Apr14~~

Session#2: April 21, 2021 from 9-11am ET - http://ibm.biz/Liberty-Apr21

Liberty 21.0.0.4-6 Update

Session#1: July 21, 2021 from 1-3pm ET - http://ibm.biz/Liberty-Jul21

Session#2: Aug 4, 2021 from 9-11am ET - http://ibm.biz/Liberty-Aug04

WebSphere Customer Advisory Board

All Customers and Business Partners welcome

http://ibm.biz/WebSphereAdvisoryBoard
email: claudiab@us.ibm.com

OPEN invitation

Join 230+ other members

Recordings/charts:

ibm.biz/WASCABCommunityResources

NEW

Monthly sessions for Business Partners and in the IST timezone

Possible engagement levels – no commitment needed:

1. Fly on the wall – NEW**
2. Stay ahead of the curve: more time commitment
3. Close the gap: quarterly involvement
4. At your own pace: impact longer term goals

What you get out of it:

- ✓ **Direct access** to architects, developers, team leads during sessions
- ✓ **Insight** into roadmaps, pre-announce insider tips (under Feedback agreement)
- ✓ Directly **influence** deliverables
- ✓ **Gain** insights from other customers
- ✓ **Bonus**: special sessions at conferences
- ✓ **Bonus**: Free Cloud assessment

Questions?

http://stackoverflow.com/questions/tagged/websphere-liberty

alasdair@ibm.com

Thank You

Your Feedback is Important

Open Liberty