Open Liberty 2020

Release Notes for Open Liberty 20.0.0.11 on Red Hat OpenShift Container Platform

Release Notes for Open Liberty 2020 on Red Hat OpenShift Container Platform

Last Updated: 2020-10-23

Open Liberty 2020 Release Notes for Open Liberty 20.0.0.11 on Red Hat OpenShift Container Platform

Release Notes for Open Liberty 2020 on Red Hat OpenShift Container Platform

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in Open Liberty 2020 on Red Hat OpenShift Container Platform release.

Table of Contents

CHAPTER 1. FEATURES

With Open Liberty 20.0.0.11 you can now make use of the Kerberos authentication protocol to secure your JDBC data sources. The Open Liberty Grafana dashboard has also been updated to be able to visualize MicroProfile Metrics data from Thanos data sources.

In Open Liberty 20.0.0.11:

- Kerberos authentication for JDBC data sources
- Grafana dashboard support for Thanos data source
- Significant bugs fixed in this release

1.1. RUN YOUR APPS USING 20.0.0.11

If you're using Maven, here are the coordinates:

```
<dependency>
    <groupId>io.openliberty</groupId>
    <artifactId>openliberty-runtime</artifactId>
    <version>20.0.0.11</version>
    <type>zip</type>
</dependency>
```

Or for Gradle:

```
dependencies {
    libertyRuntime group: 'io.openliberty', name: 'openliberty-runtime', version: '[20.0.0.11,)'
}
```

Or if you're using Docker:

```
FROM open-liberty
```

1.1.1. Kerberos authentication for JDBC data sources

Kerberos is a network authentication protocol in which a client and server authenticate by communicating with a Key Distribution Center (KDC). Kerberos authentication may be used for JDBC data sources that are backed by one of the following databases:

- IBM DB2
- Oracle Database
- Microsoft SQLServer
- PostgreSQL

Kerberos authentication in Liberty builds on top of the Kerberos Login Module and JGSS API provided by the JDK, which in turn builds on top of the Kerberos OS libraries for the specific system being used.

The **kerberos** configuration element in **server.xml** provides system-wide configuration options for the Liberty server. For example:

```
  <kerberos keytab="${server.config.dir}/security/krb5.keytab"
configFile="${server.config.dir}/security/krb5.conf"/>
```

Below is an example of the Kerberos protocol being used to secure a data source:

```
  <featureManager>
    <feature>jdbc-4.2</feature>
  </featureManager>

  <!-- optional config: This is only needed if you need to customize the location of keytab or krb5.conf -
->
  <kerberos keytab="${server.config.dir}/security/krb5.keytab"
configFile="${server.config.dir}/security/krb5.conf"/>

  <authData id="myKerberosAuth" krb5Principal="krbUser"/>

  <library id="db2DriverLib">
    <fileset dir="${server.config.dir}/db2"/>
  </library>

  <dataSource jndiName="jdbc/krb/basic" containerAuthDataRef="myKerberosAuth">
    <jdbcDriver libraryRef="db2DriverLib"/>
    <properties.db2.jcc databaseName="${DB2_DBNAME}" serverName="${DB2_HOSTNAME}"
portNumber="${DB2_PORT}"/>
  </dataSource>
```

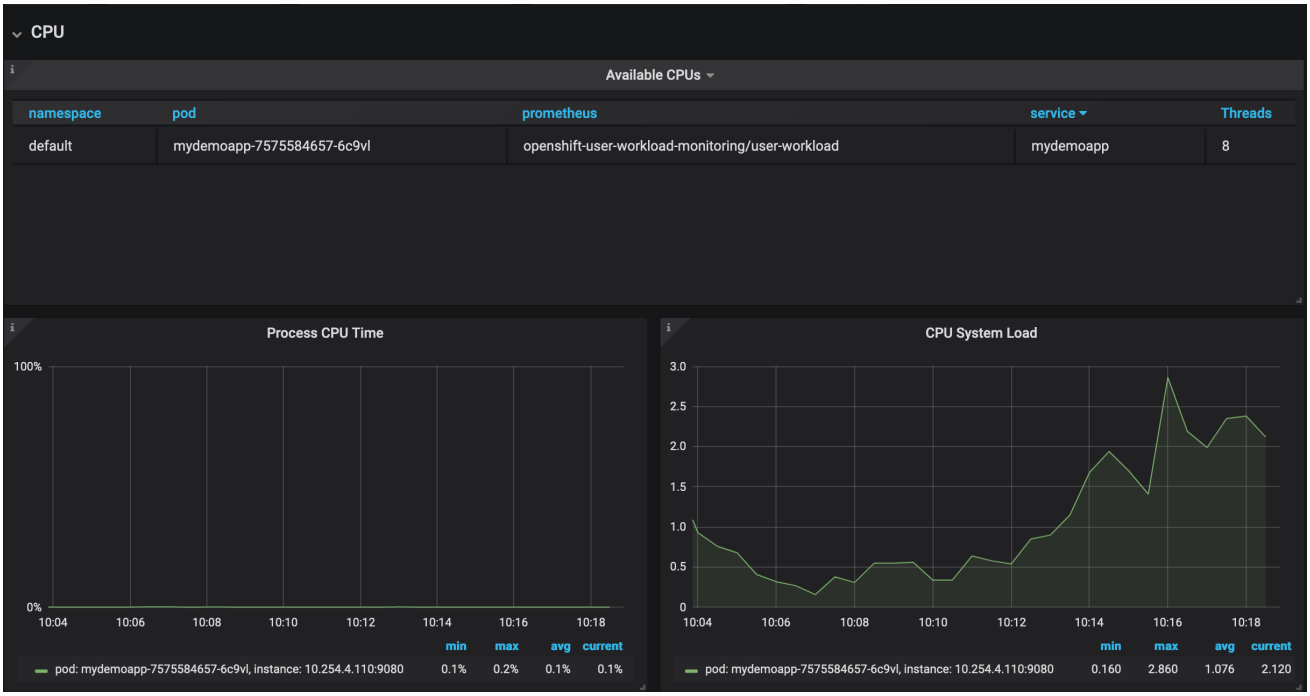
Before this release, it was technically possible to use Kerberos authentication with JDBC data sources, but the configuration was complex and un-documented. There was also previously a lack of connection pooling support when using Kerberos authentication for data sources.

1.2. GRAFANA DASHBOARD SUPPORT FOR THANOS DATA SOURCE

With Open Liberty 20.0.0.11 the Open Liberty Grafana dashboard, which visualizes data from the MicroProfile Metrics feature, is now able to visualize data from a Thanos data source.

The Grafana dashboard provides a wide range of time-series visualizations of MicroProfile Metrics data such as CPU, Servlet, Connection Pool, and Garbage Collection metrics. It is powered by a Prometheus data source which is configured to ingest data from the **/metrics** endpoint of one or more Liberty servers, enabling you to view performance metrics in near real-time.

Previously, support has only been provided for visualizing metrics data with Grafana on Open Liberty servers that used Prometheus as the data source. In Kubernetes environments, such as the Red Hat OpenShift Container Platform, you can use Thanos to query and store metrics data from multiple clusters. The previous Grafana Dashboards would not work when Thanos was set as the data source, whereas the new Grafana dashboard allows users using Thanos as the data source to display metrics data.



For more information:

- Open Liberty Grafana dashboards
- Creating your own custom visualizations with Prometheus
- Thanos in Red Hat OpenShift Container Platform

1.3. SIGNIFICANT BUGS FIXED IN THIS RELEASE

We've spent some time fixing bugs in 20.0.0.11, including the following issues:

- Server.xml config sources would not respect the "config_ordinal" property
  The MicroProfile Config specification allows for the ordinal of a built in config source to be changed by defining a property named **config_ordinal** which is read from that config source. However, built in config sources for **server.xml** variables and **<appProperties>** would not change their ordinal if a property with **config_ordinal** is set. This has now been corrected and built-in config sources will now update their ordinal when **config_ordinal** is set.

- Wrong month returned when OffsetDateTime is used via EclipseLink 2.7
  Previously, a flaw existed within EclipseLink's implementation regarding the coversion of certain java.util and java.time types. Conversions between **Calendar** and **LocalDate**/**LocalDateTime**/**OffsetDateTime** did not account for the numeric indexing differences representing calendar months, resulting in inaccurate and invalid month conversions. This behaviour has now been corrected for Open Liberty 20.0.0.11 and **OffsetDateTime** should now return a correct value consistently.

- Unable to Override JAX-RS SecurityContext in ContainerRequestFilter
  Previously, you could not override the default **SecurityContext** in a **ContainerRequestFilter**. This means that it was not possible to override behavior of **SecurityContext** methods, which is especially useful for custom behavior with JSR 250 security annotations . Starting in 20.0.0.11, you are now able to override the **SecurityContext** in a **ContainerRequestFilter** with **@Priority(AUTHORIZATION)** via **ContainerRequestContext.setSecurityContext()**. This allows you greater flexibility to control authorization of your JAX-RS application with JSR 250 annotations.

- Open Liberty Java security function did not grant default permissions from the JDK's "java.policy" file to applications
  Included with any JDK is a **java.policy** file that allows Java code to access various system-specific actions, for example the ability to read system properties (i.e. **os.name**). Open Liberty would previously return an **AccessControlException** when attempting to use the **System.getProperty()** method to read the system properties that were permitted to be read. This behavior required application developers to grant these permissions in their applications' **permissions.xml** file or in the **server.xml** unnecessarily. This has now been fixed by ensuring that all of the **java.policy** permissions are imported into all code sources.

- HTTP/1.1 and HTTP/2 behave differently when using a non-standard HTTP method
  Since release 18.0.0.2, Open Liberty has included full HTTP/2 support via Servlet 4.0. In past Open Liberty versions, the HTTP **PATCH** method, or any non-standard HTTP method, would return a **HTTP 501 Not Implemented** error when using HTTP/2. This has now been updated to allow the **PATCH** and other non-standard HTTP methods to be used with both HTTP/1.1 and HTTP/2 inside Open Liberty 20.0.0.11.

CHAPTER 2. RESOLVED ISSUES

See the Open Liberty 20.0.0.11 issues that were resolved for this release .

CHAPTER 3. FIXED CVES

For a list of CVEs that were fixed in Open Liberty 20.0.0.11, see security vulnerabilities.

CHAPTER 4. KNOWN ISSUES

See the list of issues that were found but not fixed during the development of 20.0.0.11 .