



JVM on ARM from sensor to cloud

Dmitry Chuyko
JVM Team

Who we are

Dmitry Chuyko

 [@dchuyko](https://twitter.com/dchuyko)

BELL[®]SOFT

Liberica JDK– verified OpenJDK binary

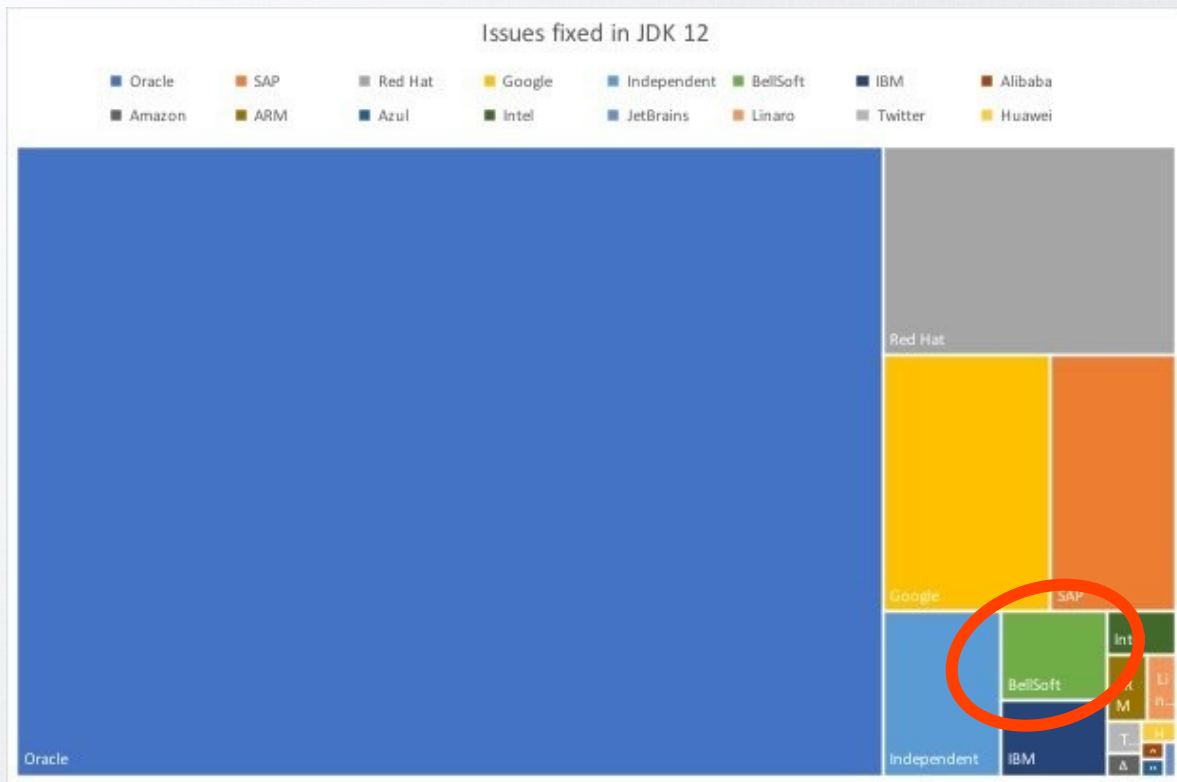
<http://bell-sw.com>

Ex-employers

ORACLE[®]



Recent statistics



blogs.oracle.com/java-platform-group/the-arrival-of-java-12

Java is 24. Still WORA

6666



9 Million
Java Developers



7 Billion
Devices



#Java20

Mature evolving open source technology

medium.com/@javachampions/java-is-still-free-2-0-0-6b9aa8d6d244

Java Is Still Free

2.0.3

With the changes to Oracle JDK distribution and support, there has been considerable uncertainty over the rights to use Oracle JDK vs Oracle OpenJDK builds vs OpenJDK builds from other providers. There are various ways to get free updates (**including security**), and (new and existing) paid support models available from various vendors to consider. This document has a [Shorter Version](#) and a much [Longer Version](#) section with all of the details.

Want to Comment?

Try our [Java Is Still Free Mirror post on Medium](#).

Shorter Version

You can still get the Oracle OpenJDK builds and OpenJDK by other providers for free under an open source license, and the [Oracle JDK remains free](#) in some circumstances. See the callout and the rest of the section for the nuances on this.

Java SE / OpenJDK / Oracle OpenJDK Builds / Oracle JDK

The [OpenJDK](#) community creates and maintains the (GPLv2+CE) open-source Reference Implementation (RI) of the **Java SE** Specification as governed by the [Java Community Process](#) (JCP) and defined through an umbrella Java Specification Request (JSR) for each feature release.

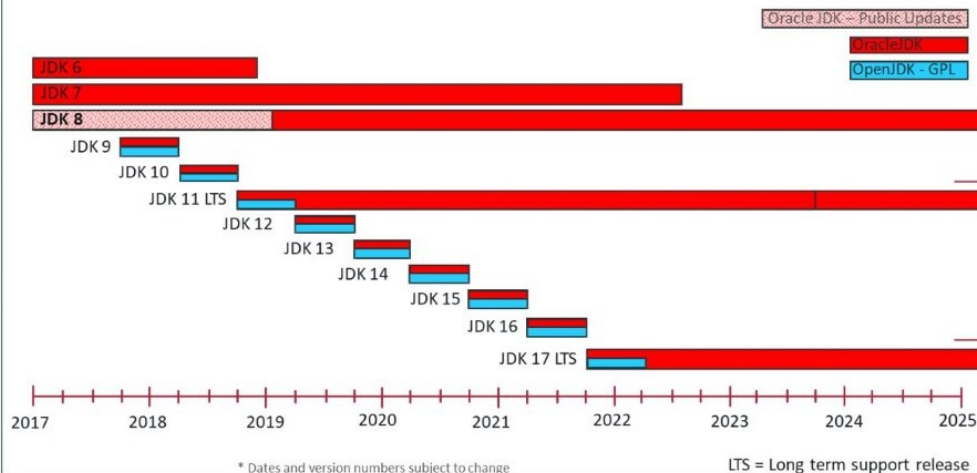
There are implementations (mostly OpenJDK based) of Java SE from various providers such as AdoptOpenJDK, Amazon, Azul, [BellSoft](#), Eclipse, IBM, jClarity, Red Hat, Oracle, SAP, and others.

Oracle JDK 8 is going through the "End of Public Updates" [process](#), which means the April 2019 update will restrict commercial use. However, since Java SE 9, Oracle is also providing [Oracle OpenJDK builds](#) which are free for commercial use (but only updated for 6 months). There are also free **OpenJDK** builds which will be updated (**including security patches**) from other providers like AdoptOpenJDK, Amazon, Azul, [BellSoft](#), IBM, jClarity, Red Hat, Linux distros et al.

Going forward there are several options to get a JDK. We focus on Java SE 8 and Java SE 11 which is the first Long Term Support (LTS) release under the new release cadence.

Starting with Java SE 9

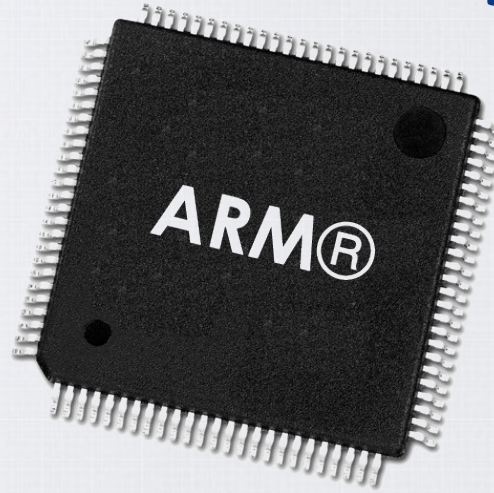
New Java Release Timeline*



- Specification
- Variety of languages
- Ecosystem
- Security
 - ♦ Managed Platform
 - ♦ Data typing
 - ♦ Verification
 - ♦ Memory management
 - ♦ APIs
 - ♦ Cryptography
 - ♦ Authentication
 - ♦ Public Key Infrastructure



- Variety of implementations
- Variety of platforms
- Binary distributions
- Security
 - ♦ Critical patch updates



OpenJDK

- Tarballs, installers
- Apt, yum
- Docker Hub
- Other channels

Binaries

bell-sw.com/java.html

Download: Liberica JDK

Liberica JDK version 12

[Release Notes](#)

Liberica is a 100% open-source Java 12 implementation. It is built from OpenJDK which BellSoft contributes to, is thoroughly tested and passed the JCK provided under the license from OpenJDK. The versions of Liberica for Windows x86_64, Windows x86, Mac x86_64, Linux x86_64 and ARMv7 also contain JavaFX 12. The version for Linux ARMv7 contains Device IO API as additional module and JavaFX with hardware-accelerated EGL support. We currently provide binaries suitable for different hardware and OS combinations:

- **Windows x86_64** (64-bit version for Microsoft Windows)
- **Windows x86** (32-bit version for Microsoft Windows)
- **macOS x86_64** (64-bit version for Apple macOS)
- **Linux x86_64** (64-bit version for Linux servers and desktops)
- **Linux x86** (32-bit version for Linux desktops and embedded)
- **Alpine Linux x86_64** (64-bit version for Alpine Linux with musl)
- **Linux ARMv8** (64-bit version for ARMv8 servers and embedded)
- **Linux ARMv7 HardFloat** (32-bit version for Raspbian on Raspberry Pi 2 and Raspberry Pi 3)
- **Solaris SPARC** (SPARCv9 version for Solaris)
- **Solaris x86_64** (x86_64 version for Solaris)

Docker images for Liberica JDK can be found in BellSoft [Docker hub](#), [YUM](#) and [APT](#) repositories are available.

Commercial support is available for Liberica JDK. For detailed information, quote requests and additional platforms please contact sales@bell-sw.com.



Release Notes

- For the edge
 - ◆ Device IO (I2C, DAC/ADC, GPIO, UART, SPI)
 - ◆ OpenJFX (EGL, FB)
- For the cloud
 - ◆ Fast Server VM
 - ◆ GCs (G1, Shenandoah...)
 - ◆ Compilers (C2, Graal)
 - ◆ JFR, AOT, AppCDS...

Liberica JDK 12

The full version string for this binary release of OpenJDK is 12+33, and the version number is 12.

Liberica JDK 12

Liberica is a certified, Java SE 12-compliant distribution of OpenJDK 12 which works on server (Linux x86_64, Linux ARM64, Solaris SPARC, Solaris x64, Windows 64), desktop (Windows 64, Windows 32, Mac, Linux x86_64, Linux x86), and embedded devices (Linux x86, Linux ARM64, Linux ARMv7, including Raspberry Pi 2 & 3 (ARMv6 hardfloat)). It has the following notable additions:

- Linux & Windows x86_64 binaries contain experimental support for Shenandoah GC, Linux x86_64 version contains experimental support for ZGC.
- Linux x86_64, ARMv8 and ARMv7 distributions include a choice of Client VM, Server VM and Minimal VM.
- Alpine Linux x86_64 version is built with musl support.
- Windows x86_64, Mac, Linux x86_64 and Linux ARMv7 distributions contain OpenJFX 12.
- Linux ARMv7 distribution contains Device IO API compiled for Raspberry Pi.

Please refer to the Oracle JDK 12 [release notes](#) for further information on JDK 12 features. This document further outlines the peculiarities of Liberica distribution as compared to Oracle JDK 12 distribution.

Supported Server and Desktop configurations

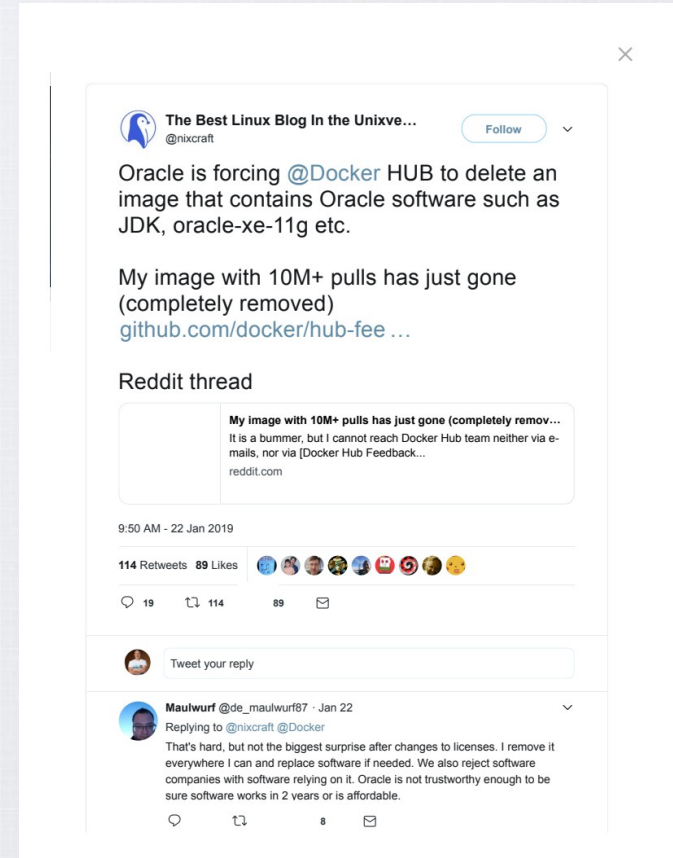
Liberica is supported on the following OSes:

- Ubuntu Linux 16.04, 18.04 (x86, x86_64, ARMv8)
- Red Hat, Oracle Linux and CentOS 6.x, 7.x (x86, x86_64, ARMv8)
- Alpine Linux 3.7+
- SUSE Linux 15+ and tumbleweed (x86_64)
- Apple macOS 11.10+
- Solaris 11.1+ (SPARC and x64)
- Microsoft Windows 2019, Microsoft Windows 2016, Microsoft Windows 2012 R2, Windows 10, Windows 8, Windows 7 (64 and 32 bit)

Derive app runtimes: JDK 11 ARMv7 with Minimal VM and java.base is **17 MB**, including Security manager

- Standard deployment nowadays
- Java 8, 11, 12, 13+ gets special support
 - ♦ cgroups
 - ♦ Namespaces
- ♦ Everyone needs base images

Linux Containers



Base images

<https://hub.docker.com/u/bellsoft>

	JRE 8	JDK 11
Debian	227 MB	622 MB
Centos	307 MB	702 MB
Alpine	114 MB	509 MB
Alpine musl base		37 MB

The screenshot shows the Docker Hub profile for the user 'bellsoft'. The profile lists several repositories, all of which are 'bellsoft/liberica-openjdk-*' images. The repositories are listed with their names, the user 'bellsoft', the update time, and the download count. The repositories are:

- bellsoft/liberica-openjdk-alpine (407 Downloads)
- bellsoft/liberica-openjdk-centos (235 Downloads)
- bellsoft/liberica-openjdk-debian (257 Downloads)
- bellsoft/liberica-openjdk-alpine (1.3K Downloads)
- bellsoft/liberica-openjdk-centos (832 Downloads)
- bellsoft/liberica-openjdk-debian (1.3K Downloads)
- bellsoft/liberica-openjdk-alpine-musl (115 Downloads)

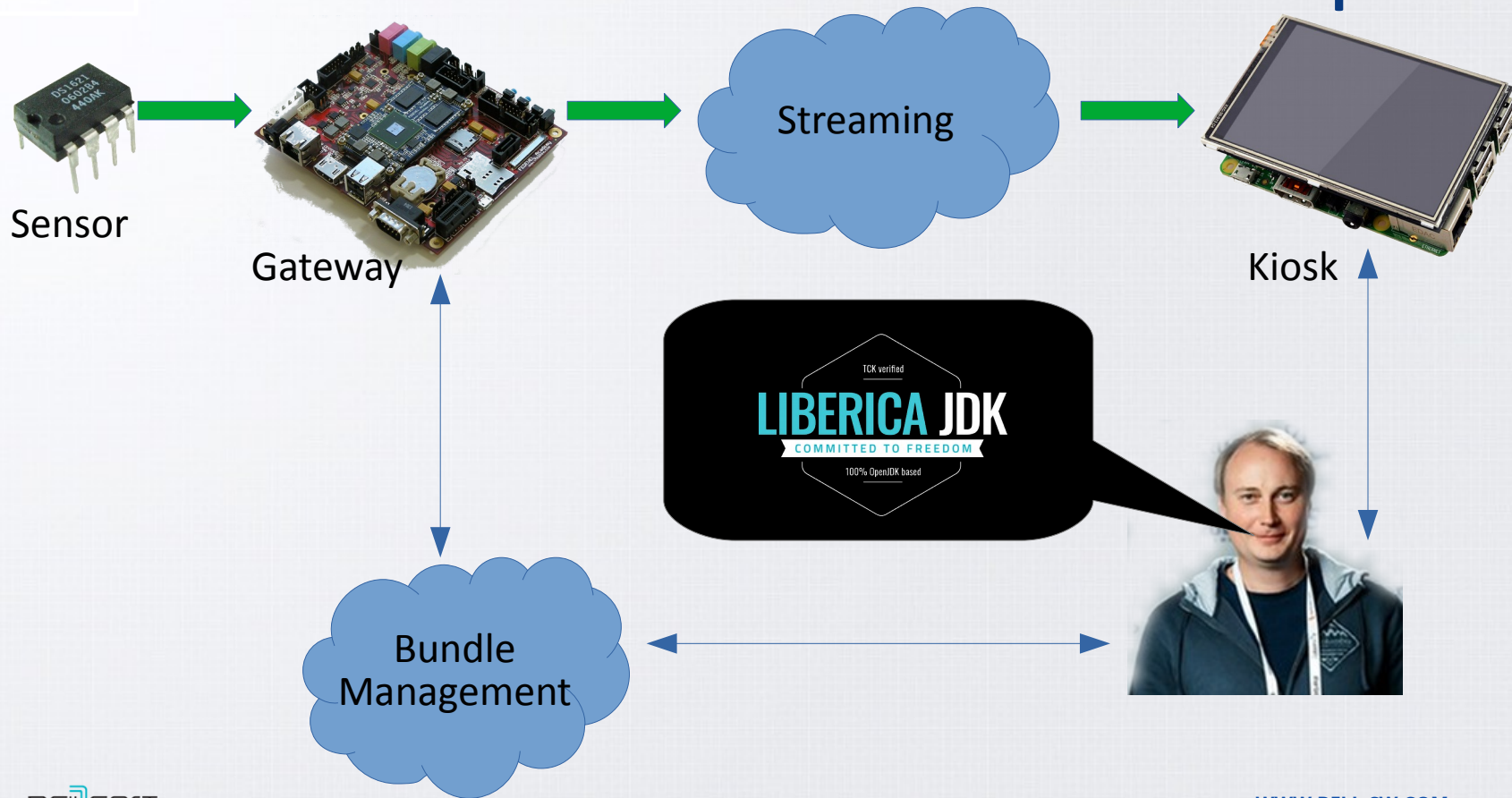
The footer of the page shows the Docker Hub logo, the user 'bellsoft', and the copyright notice 'Copyright © 2013 Docker Inc. All rights reserved.'.

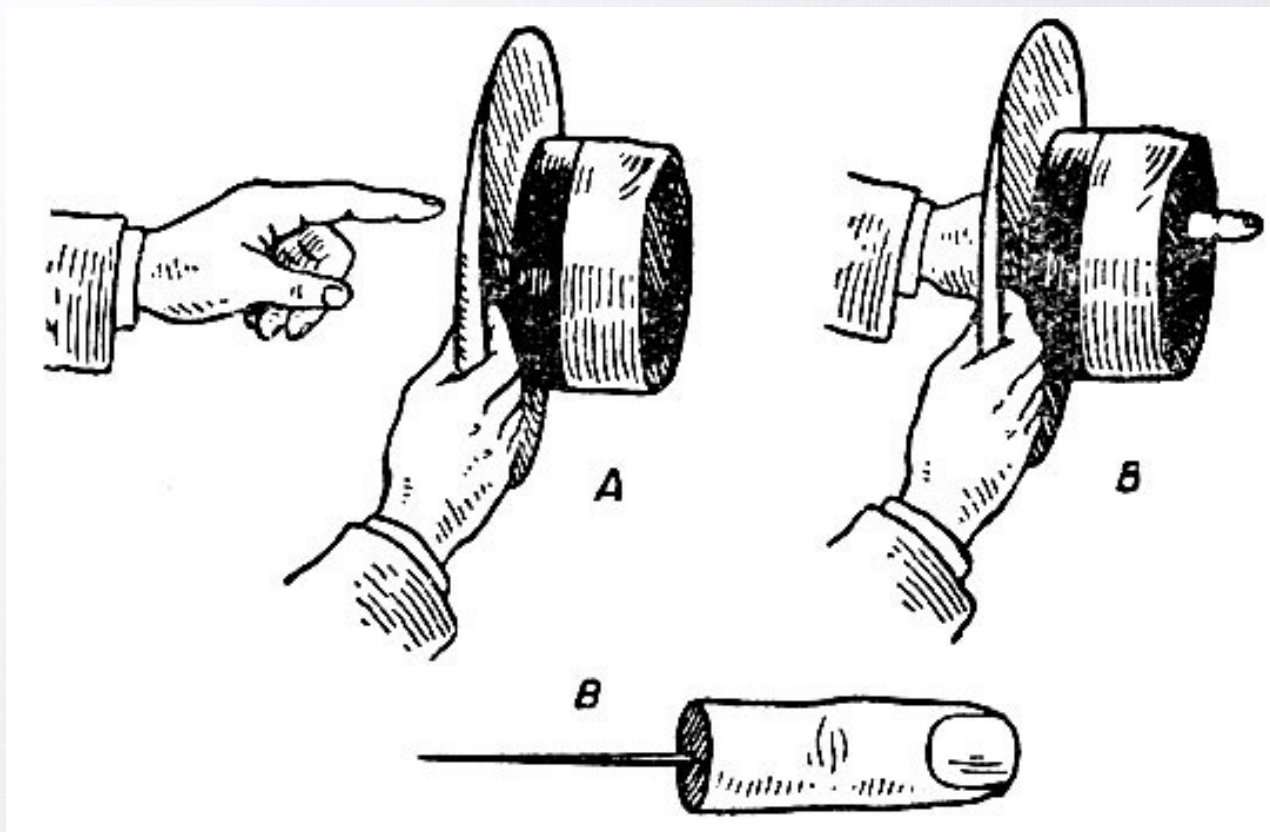
Base images

- Java versions
 - ◆ 12
 - ◆ 11 (LTS)
 - ◆ 8 (LTS)
- Distro
 - ◆ Debian
 - ◆ CentOS
 - ◆ Alpine
 - ◆ Alpine musl
- Arch
 - ◆ x86_64
 - ◆ ARM64
 - ◆ ARM32

The screenshot shows the Docker Hub interface for the `bellsoft/liberica-openjdk-debian` repository. At the top, there's a banner for DockerCon. Below it, the repository name is displayed with a star icon and a pull count of 1.5K. A description states that Liberica is a 100% open-source Java implementation built from OpenJDK. The 'Overview' tab is selected, showing a 'What is Liberica?' section with details about its open-source nature and supported architectures: x86_64 (aka amd64), aarch64 (i.e. ARM64), and armhf (for devices like Raspberry Pi 3/2). A 'Docker Pull Command' box shows the command `docker pull bellsoft/lib`. The 'Tags' section lists various versions including 11.0.2, 11, latest, 10.0.2, 10, 9.0.4, 9, armhf only (Raspberry Pi 3/2), and build - build - 8 - amd64 and arm64 only. The 'Usage' section provides example commands for running a container and installing the application.

Example





OSGi Device Management

- Manage devices
- Deploy bundles
- Control bundles
- Set permissions

The screenshot displays the Bosch OSGi Device Management web interface. The top navigation bar includes links for Log Out, Settings, and About, along with the Bosch logo and the tagline 'Invented for life'. The main content area is divided into two sections: 'OSGi Bundles' and 'Bundles Info'.

OSGi Bundles - 2 results

OSGi Bundle	Location	Bundle ID	State	Persist	Start
kotlin-osgi-bundle 1.3.21	mprm://org.jetbrains.kotlin.osgi-bunc95	95	Active	Yes	9
Tdemo-sensor-kotlin 1.0.0.20190	mprm://tdemo-sensor-kotlin	96	Resolve No		9

Bundles Info

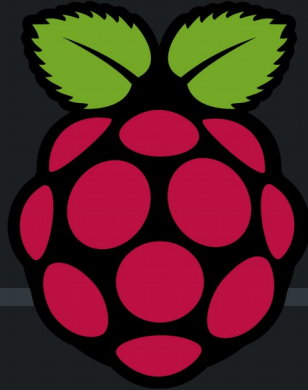
OSGi Bundle Properties

Location:

Bundle ID:

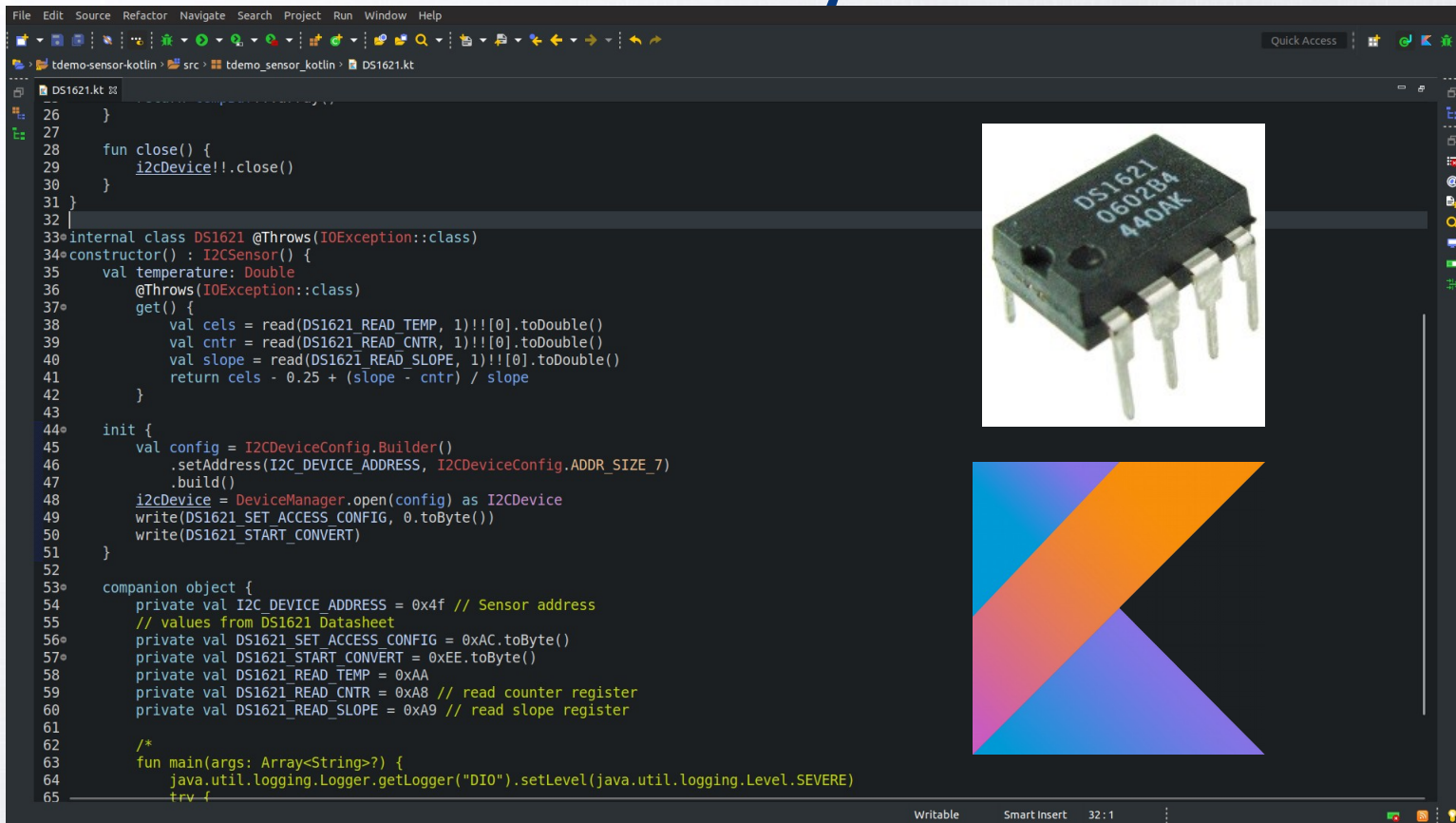
State:

Manifest

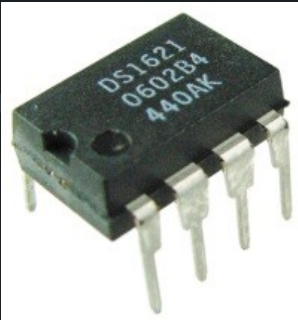


```
File Edit Source Refactor Navigate Search Project Run Window Help
tdemo-sensor-kotlin > src > tdemo_sensor_kotlin > Activator.java
DS1621.kt J Activator.java
14 DS1621 ds;
15 Timer timer;
16 Producer<Long, Double> producer;
17
18 @Override
19 public void start(BundleContext context) throws Exception {
20     java.util.logging.Logger.getLogger("DIO").setLevel(java.util.logging.Level.SEVERE);
21     try {
22         ds = new DS1621();
23         producer = KafkaAccess.INSTANCE.createProducer();
24         timer = new Timer();
25         timer.scheduleAtFixedRate(new TimerTask() {
26             @Override
27             public void run() {
28                 try {
29                     long ms = System.currentTimeMillis();
30                     double extT = ds.getTemperature();
31                     double intT = Zones.getTemperature(0);
32                     System.out.format("Ms: %d t (ext): %f t (int): %f\n", ms, extT, intT);
33                     producer.send(new ProducerRecord<("external", ms, extT));
34                     producer.send(new ProducerRecord<("internal", ms, intT));
35                 } catch (Exception e) {
36                     e.printStackTrace();
37                 }
38             }
39             }, 0, 1000);
40     } catch (Exception ex) {ex.printStackTrace();}
41 }
42
43 @Override
44 public void stop(BundleContext context) throws Exception {
45     if (timer != null) {
46         timer.cancel();
47         producer.close();
48         ds.close();
49     }
50 }
51 }
52 }
53 }
```

Gateway. DIO Sensor & Kotlin



```
File Edit Source Refactor Navigate Search Project Run Window Help
tdemo-sensor-kotlin > src > tdemo_sensor_kotlin > DS1621.kt
DS1621.kt
26 }
27
28 fun close() {
29     i2cDevice!!.close()
30 }
31 }
32
33 internal class DS1621 @Throws(IOException::class)
34 constructor() : I2CSensor() {
35     val temperature: Double
36     @Throws(IOException::class)
37     get() {
38         val cels = read(DS1621_READ_TEMP, 1)!![0].toDouble()
39         val cntr = read(DS1621_READ_CNTR, 1)!![0].toDouble()
40         val slope = read(DS1621_READ_SLOPE, 1)!![0].toDouble()
41         return cels - 0.25 + (slope - cntr) / slope
42     }
43 }
44 init {
45     val config = I2CDeviceConfig.Builder()
46         .setAddress(I2C_DEVICE_ADDRESS, I2CDeviceConfig.ADDR_SIZE_7)
47         .build()
48     i2cDevice = DeviceManager.open(config) as I2CDevice
49     write(DS1621_SET_ACCESS_CONFIG, 0.toByte())
50     write(DS1621_START_CONVERT)
51 }
52
53 companion object {
54     private val I2C_DEVICE_ADDRESS = 0x4f // Sensor address
55     // values from DS1621 Datasheet
56     private val DS1621_SET_ACCESS_CONFIG = 0xAC.toByte()
57     private val DS1621_START_CONVERT = 0xEE.toByte()
58     private val DS1621_READ_TEMP = 0xAA
59     private val DS1621_READ_CNTR = 0xA8 // read counter register
60     private val DS1621_READ_SLOPE = 0xA9 // read slope register
61 }
62
63 /*
64 fun main(args: Array<String>?) {
65     java.util.logging.Logger.getLogger("DIO").setLevel(java.util.logging.Level.SEVERE)
66     try {
67         // ...
68     }
69 }
```



```
File Edit Source Refactor Navigate Search Project Run Window Help
tdemo-sensor-kotlin > src > tdemo_sensor_kotlin > KafkaAccess.kt
DS1621.kt Activator.java KafkaAccess.kt
1 package tdemo_sensor_kotlin
2
3 import org.apache.kafka.clients.producer.*
4
5
6
7
8 object KafkaAccess {
9     private val BOOTSTRAP_SERVERS = "xxx.xxx.xxx.xxx:9092"
10    private val CLIENT_ID = "OSGiThermo"
11
12    fun createProducer(): Producer<Long, Double>? {
13        val props = HashMap<String, Any>() // producer.properties
14        /* Kafka */
15        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, BOOTSTRAP_SERVERS)
16        /* TLS connection */
17        // props.put("security.protocol", "SSL")
18        // props.put("ssl.enabled.protocols", "TLSv1.1")
19        // props.put("ssl.protocol", "TLSv1.1")
20        // props.put("ssl.truststore.type", "JKS")
21        // props.put("ssl.truststore.location", "/var/private/ssl/kafka.client.truststore.jks")
22        // props.put("ssl.truststore.password", "<password1>")
23        /* Client certificate authentication on Broker */
24        // props.put("ssl.keystore.location", "/var/private/ssl/kafka.client.keystore.jks")
25        // props.put("ssl.keystore.password", "<password2>")
26        // props.put("ssl.key.password", "<password3>")
27        /* Topic */
28        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, LongSerializer::class.java.getName())
29        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, DoubleSerializer::class.java.getName())
30        /* Client behavior */
31        props.put(ProducerConfig.CLIENT_ID_CONFIG, CLIENT_ID)
32        props.put(ProducerConfig.ACKS_CONFIG, "1")
33        props.put(ProducerConfig.RETRIES_CONFIG, "0")
34        props.put(ProducerConfig.BATCH_SIZE_CONFIG, "2")
35        props.put(ProducerConfig.LINGER_MS_CONFIG, "0")
36        return KafkaProducer(props)
37    }
38 }
```



Cloud. Kafka & Docker

19

packet

```
File Edit View Search Terminal Help

processor      : 93
BogoMIPS      : 200.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x43
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0x0a1
CPU revision  : 1

processor      : 94
BogoMIPS      : 200.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x43
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0x0a1
CPU revision  : 1

processor      : 95
BogoMIPS      : 200.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x43
CPU architecture: 8
CPU variant   : 0x1
CPU part      : 0x0a1
CPU revision  : 1

dchuyko@dpochepek:~$ uname -a
Linux dpochepek 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:20 UTC 2018 aarch64 aarch64 aarch64 GNU/Linux
dchuyko@dpochepek:~$ free
              total        used        free      shared  buff/cache   available
Mem:      131963460     4494232     64420424      1047892      63048804     125442672
Swap:      3321056       217680       3103376

dchuyko@dpochepek:~$ docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED            STATUS            PORTS                               NAMES
26517e62eccc       bellsoft/liberica-openjdk-debian       "/kafka/bin/kafka-se..." 3 hours ago       Up 3 hours       0.0.0.0:9092->9092/tcp             kafka
08d82b4036fd       bellsoft/liberica-openjdk-debian       "/kafka/bin/zookeepe..." 3 hours ago       Up 3 hours       0.0.0.0:2181->2181/tcp             zookeeper
```

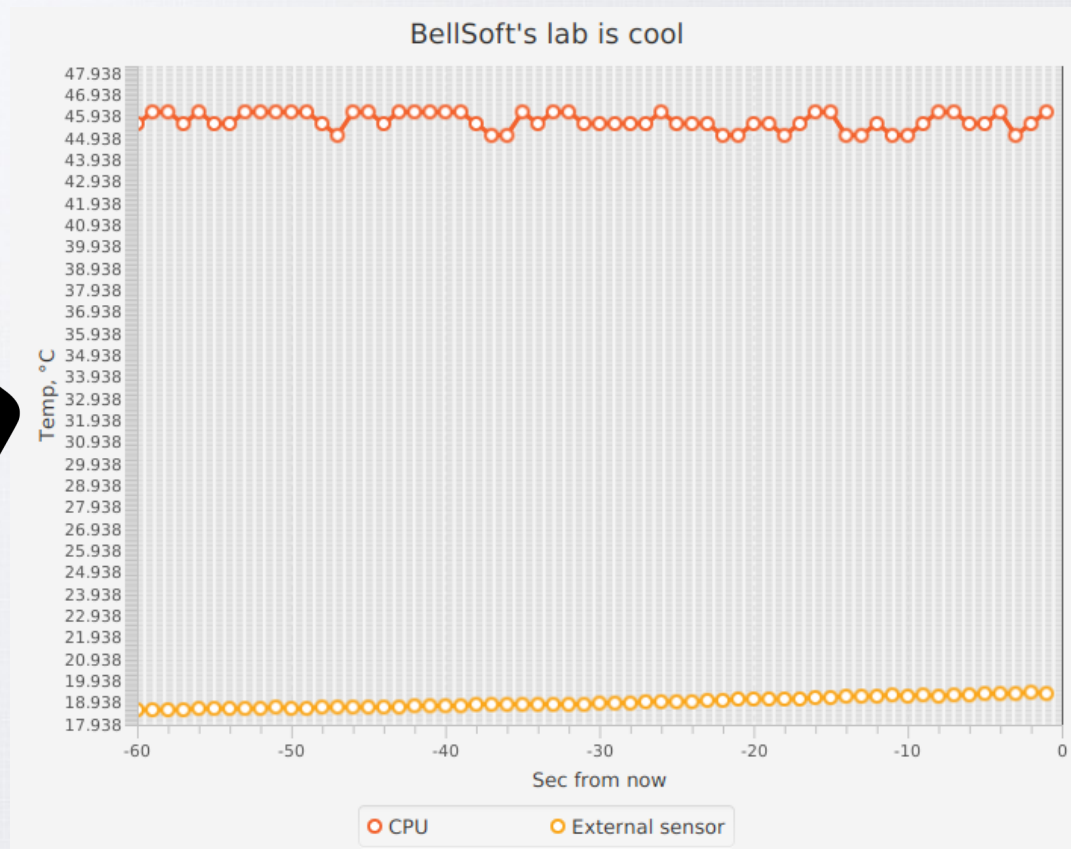
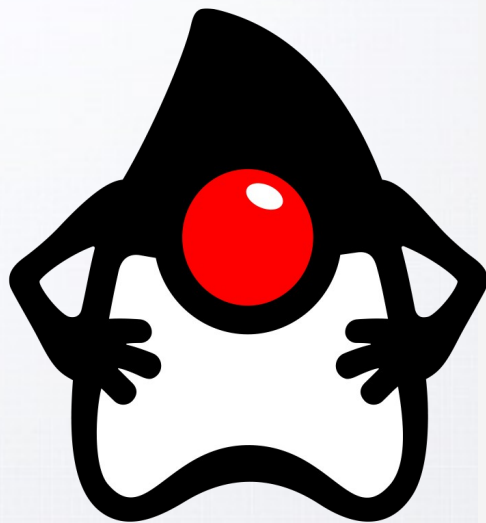


Kiosk. OpenJFX & Kafka



```
File Edit Source Refactor Navigate Search Project Run Window Help
demo-kiosk-jfx > src > demo_kiosk_jfx > TemperatureApp.java
TemperatureApp.java
31- public TemperatureApp() {
32-     animation = new Timeline();
33-     animation.getKeyFrames().add(new KeyFrame(Duration.seconds(1.0), (e) -> updateData()));
34-     animation.setCycleCount(Animation.INDEFINITE);
35-     data1 = FXCollections.observableArrayList();
36-     data2 = FXCollections.observableArrayList();
37- }
38-
39- void updateData() {
40-     ConsumerRecords<Long, Double> records = consumer.poll(50);
41-     records.forEach(record -> {
42-         long delta = (record.key() - System.currentTimeMillis()) / 1000;
43-         Data<Number, Number> point = new Data<>(delta + 1, record.value());
44-         if (record.topic().equals(TOPIC_INT))
45-             data1.add(point);
46-         if (record.topic().equals(TOPIC_EXT))
47-             data2.add(point);
48-     });
49-     consumer.commitSync();
50-     data1.forEach(d -> d.setXValue(d.getXValue().longValue() - 1));
51-     data2.forEach(d -> d.setXValue(d.getXValue().longValue() - 1));
52-     yAxis.setUpperBound(data1.stream().mapToDouble(d -> d.getYValue().doubleValue()).max().orElse(60) + 0.5);
53-     yAxis.setLowerBound(data2.stream().mapToDouble(d -> d.getYValue().doubleValue()).min().orElse(0) - 0.5);
54- }
55-
56- public Parent createContent() {
57-     xAxis = new NumberAxis("Sec from now", -60, 0, 10);
58-     yAxis = new NumberAxis("Temp, °C", 20, 53, 0.1);
59-     ObservableList<Series<Number, Number>> chartData = FXCollections.observableArrayList();
60-     chartData.add(new Series<Number, Number>("CPU", data1));
61-     chartData.add(new Series<Number, Number>("External sensor", data2));
62-     chart = new LineChart<Number, Number>(xAxis, yAxis, chartData);
63-     chart.setTitle("BellSoft's lab is cool");
64-     return chart;
65- }
66-
67- @Override
68- public void start(Stage primaryStage) throws Exception {
69-     primaryStage.setTitle("Monitoring kiosk");
70-     primaryStage.setFullScreen(true);
71-     primaryStage.setScene(new Scene(createContent()));
72- }
```

0000



Summary

- Liberica JDK on Arm brings value to the ecosystem from edge or devices to cloud
 - Min runtime: 17 Mb footprint and energy savings
 - Max runtime: Optimized for throughput, low latency on high-end ARM servers
- JVM runtime unleashes polyglot programming approach
 - Java, Scala, Kotlin, you name it
- Running on JVM runtime allows to re-use applications and libraries on ARM hardware with zero changes
- Liberica JDK is available for edge and cloud
 - Free for use and distribution
 - With commercial support option (24x7 worldwide coverage extending from servers and clouds to edge and any screen)