Java 11

Christopher M. Judd
Almost overnight, Docker has become the de facto standard that developers and system administrators use for packaging, deploying, and running distributed applications. It provides tools for simplifying DevOps by enabling developers to create templates called images that can be used to create lightweight virtual machines called containers, which include their applications and all of their applications' dependencies. These lightweight virtual machines can be promoted through testing and production environments where sysadmins deploy and run them. Docker makes it easier for organizations to automate infrastructure, isolate applications, maintain consistency, and improve resource utilizations.

Similar to the popular version control software Git, Docker has a social aspect, in that developers and sysadmins are able to share their images via Docker Hub. Docker is an open-source solution that runs natively on Linux but also works on Windows and Mac using a lightweight Linux distribution and VirtualBox. Many tools have also grown up around Docker to make it easier to manage and orchestrate complex distributed applications.

**About Docker**

**Docker Architecture**

Docker utilizes a client-server architecture and a remote API to manage and create Docker containers built upon Linux containers. Docker containers are created from Docker images. The relationship between containers and images are analogous to the relationship between objects and classes in object-oriented programming.

- **Docker Images**: A recipe or template for creating Docker containers. It includes the steps for installing and running the necessary software.
- **Docker Container**: Like a tiny virtual machine that is created from the instructions found within the Docker image.
- **Docker Client**: Command-line utility or other tool that takes advantage of the Docker API (https://docs.docker.com/reference/api/docker_remote_api) to communicate with a Docker daemon.
- **Docker Host**: A physical or virtual machine that is running a Docker daemon and contains cached images as well as runnable containers created from images.

**Docker Monitoring**

Get Detailed Insight into Docker Containers
JDK 11 Releases

The JDK 11 Early Access Program has concluded. Please visit Java SE Downloads for production ready builds.

Older releases, which do not include the most up to date security vulnerability fixes and are no longer recommended for use in production, remain available in the OpenJDK Archive.
Archived OpenJDK General-Availability Releases

This page is an archive of previously released builds of the JDK licensed under the GNU General Public License, version 2, with Classpath Exception.

**WARNING:** These older versions of the JDK are provided to help developers debug issues in older systems. They are not updated with the latest security patches and are not recommended for use in production.

### Releases

<table>
<thead>
<tr>
<th>12 GA</th>
<th>Windows</th>
<th>64-bit</th>
<th>zip (sha256) 188M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mac</td>
<td>64-bit</td>
<td>tar.gz (sha256) 182M</td>
</tr>
<tr>
<td></td>
<td>Linux</td>
<td>64-bit</td>
<td>tar.gz (sha256) 189M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11.0.2</th>
<th>Windows</th>
<th>64-bit</th>
<th>zip (sha256) 179 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mac</td>
<td>64-bit</td>
<td>tar.gz (sha256) 174 MB</td>
</tr>
<tr>
<td></td>
<td>Linux</td>
<td>64-bit</td>
<td>tar.gz (sha256) 179 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11.0.1</th>
<th>Windows</th>
<th>64-bit</th>
<th>zip (sha256) 179 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mac</td>
<td>64-bit</td>
<td>tar.gz (sha256) 174 MB</td>
</tr>
<tr>
<td></td>
<td>Linux</td>
<td>64-bit</td>
<td>tar.gz (sha256) 179 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11 GA</th>
<th>Windows</th>
<th>64-bit</th>
<th>zip (sha256) 179 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mac</td>
<td>64-bit</td>
<td>tar.gz (sha256) 175 MB</td>
</tr>
<tr>
<td></td>
<td>Linux</td>
<td>64-bit</td>
<td>tar.gz (sha256) 179 MB</td>
</tr>
</tbody>
</table>

http://jdk.java.net/archive/
JDK Distributions
OpenJDK
Oracle JDK (Commercial Support)
Alternatives
Amazon Corretto
Azul Zulu
BellSoft Liberia JDK
SapMachine
Java is Still Free 2.0.3

This is a repeat of (version 2.0.3) of the Google Doc put together by the Java Champions community of independent Java leaders and experts.

- Older version 1.0.0—17th of Sept 2018—if you’re curious where we were ~6 months ago.
- Portuguese translation by @leomrlima—TBD
- Spanish translation by @aalmiray et al—TBD

... 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 detail.
Prebuilt OpenJDK Binaries for Free!

Java™ is the world’s leading programming language and platform. AdoptOpenJDK uses infrastructure, build and test scripts to produce prebuilt binaries from OpenJDK™ class libraries and a choice of either the OpenJDK HotSpot or Eclipse OpenJ9 VM.

All AdoptOpenJDK binaries and scripts are open source licensed and available for free.

Download for macOS x64

1. Choose a Version
   - OpenJDK 8 (LTS)
   - OpenJDK 11 (LTS)
   - OpenJDK 12 (Latest)

2. Choose a JVM
   - HotSpot
   - OpenJ9

[Latest release]
jdk8u214-b04

Other platforms
Release Archive & Nightly Builds

AdoptOpenJDK now also distributes OpenJDK upstream builds!

https://adoptopenjdk.net/
docker pull openjdk:11
docker run -it --rm openjdk:11 bash
docker run -it --rm openjdk:11 jshell

docker pull adoptopenjdk/openjdk:11
docker run -it --rm adoptopenjdk/openjdk:11 bash
docker run -it --rm adoptopenjdk/openjdk:11 jshell

docker pull amazoncorretto:11
docker run -it --rm amazoncorretto:11 bash
docker run -it --rm amazoncorretto:11 jshell

docker pull azul/zulu-openjdk:11
docker run -it --rm azul/zulu-openjdk:11 bash
docker run -it --rm azul/zulu-openjdk:11 jshell
JEP 330: Launch Single-File Source-Code Programs

Release: 11
Component: tools/javac

Enhance the java launcher to run a program supplied as a single file of Java source code, including usage from within a script by means of "shebang" files and related techniques.

https://openjdk.java.net/jeps/330
public class HelloWorld {

    public static void main(String[] args) throws Exception {
        System.out.println("Hello World");
    }

}
public class HelloWorld {

    public static void main(String[] args) throws Exception {
        System.out.println("Hello World");
    }

}
public class HelloWorld {

    public static void main(String[] args) throws Exception {
        System.out.println("Hello World");
    }

}
HelloWorld

```java
#!/usr/bin/java --source 11

public class HelloWorld {

    public static void main(String[] args) throws Exception {
        System.out.println("Hello World");
    }

}
```

no .java extension

shebang
Define a new HTTP client API that implements HTTP/2 and WebSocket, and can replace the legacy HttpURLConnection API.
import java.net.*;
import java.net.http.*;
import java.net.http.HttpClient.*;
import java.net.http.HttpResponse.*;
import java.time.*;

public class jcurl {

    public static void main(String[] args) throws Exception {

        HttpClient client = HttpClient.newBuilder()
            .version(Version.HTTP_1_1)
            .followRedirects(Redirect.NORMAL)
            .connectTimeout(Duration.ofSeconds(20))
            .build();

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(args[0]))
            .timeout(Duration.ofMinutes(1))
            .build();

        HttpResponse<String> response = client.send(request, BodyHandlers
            .ofString());
        System.out.println(response.statusCode());
        System.out.println(response.body());
    }
}
JEP 323: Local-Variable Syntax for Lambda Parameters

Release: 11
Component: tools

Allow var to be used when declaring the formal parameters of implicitly typed lambda expressions.

https://openjdk.java.net/jeps/323
Java 11 add `var` to Lambda parameters

```java
list.stream()
    .map((var s) -> s.toLowerCase())
    .collect(Collectors.toList());
```
Java 11 add `var` to Lambda parameters

```java
list.stream()
    .map((var s) -> s.toLowerCase())
    .collect(Collectors.toList());
```

Lambda's already act like `var` so why add vars to Lambda?

```java
list.stream()
    .map(s -> s.toLowerCase())
    .collect(Collectors.toList());
```
Lambda's already act like var so why add vars to Lambda?

```
list.stream()
    .map((var s) -> s.toLowerCase())
    .collect(Collectors.toList());
```

```
list.stream()
    .map(s -> s.toLowerCase())
    .collect(Collectors.toList());
```

```
list.stream()
    .map((@Notnull var s) -> s.toLowerCase())
    .collect(Collectors.toList());
```
JEP 327: Unicode 10

Release: 11
Component: core-libs/java.lang

Upgrade existing platform APIs to support version 10.0 of the Unicode Standard.

https://openjdk.java.net/jeps/327
main, WRITE: TLSv1.2 Handshake, length = 40
[Raw write]: length = 45
0000: 16 03 03 00 28 00 00 00 00 00 00 00 C5 62 F9 ....(............b.
0010: C6 4B 80 EE 26 C5 0D 1A 79 47 5F DA CD 3B DD 10 .K..&...yG_..;..
0020: D7 B5 27 64 DE 12 57 6D 35 35 00 1A EB ..'d..Wm55...
[Raw read]: length = 5
0000: 14 03 03 00 01 ......
[Raw read]: length = 1
0000: 01 .
main, READ: TLSv1.2 Change Cipher Spec, length = 1
Java 11

javax.net.ssl|DEBUG|0D|HttpClient-1-Worker-0|2019-07-08 21:12:46.516 EDT|ServerHello.java: 884|Consuming ServerHello handshake message ( 
"ServerHello": { 
  "server version": "TLSv1.2", 
  "random": "C8 62 CE 2D 4B E7 B9 A4 F5 C9 D5 17 89 62 DD 26 B9 D0 81 7D 20 43 AD 47 93 8D 71 27 27 6B C3 07", 
  "session id": "25 88 4F 27 6D F4 D5 59 8F 18 40 4C 61 D0 91 3E 4B F7 9B 52 1E 30 3E 42 19 1A C0 AF 65 AF 47 FA", 
  "cipher suite": "TLS_AES_128_GCM_SHA256(0x1301)", 
  "compression methods": "00", 
  "extensions": [ 
    "key_share (51)": { 
      "server_share": { 
        "named group": secp256r1 
      }, 
    }, 
    "supported_versions (43)": { 
      "selected version": [TLSv1.3] 
    } 
  ] 
}
javax.net.ssl|DEBUG|0D|HttpClient-1-Worker-0|2019-07-08 21:12:46.517 EDT|SSLExtensions.java: 188|Consumed extension: supported_versions 
javax.net.ssl|DEBUG|0D|HttpClient-1-Worker-0|2019-07-08 21:12:46.517 EDT|ServerHello.java: 980|Negotiated protocol version: TLSv1.3
Implement the ChaCha20 and ChaCha20-Poly1305 ciphers as specified in RFC 7539. ChaCha20 is a relatively new stream cipher that can replace the older, insecure RC4 stream cipher.
JEP 324: Key Agreement with Curve25519 and Curve448

Release: 11
Component: security-libs / javax.crypto

Implement key agreement using Curve25519 and Curve448 as described in RFC 7748.
Nests allow nested classes that are part of the same enclosing class but compiled to different class files, to access each other's private members without the need for compilers to insert synthetic generated accessibility-broadening bridge methods. This is a Java class bytecode level change.
public class Outer {

    private static int x = 5;

    public static class Nested {
        public static void doSomething () {
            System.out.println(x);
        }
    }
}

JEP 318: Epsilon: No-Op GC
JEP 333: ZGC: Scalable Low-Latency GC
Develop a GC that handles memory allocation but does not implement any actual memory reclamation mechanism. Once the available Java heap is exhausted, the JVM will shut down.
Why would you want a Garbage Collector that doesn't collect Garbage?
Why would you want a Garbage Collector that doesn't collect Garbage?

- performance testing
- memory pressure testing
- extremely short lived jobs
- last-drop latency improvements
- last-drop throughput improvements

-XX:+UnlockExperimentalVMOptions -XX:+UseEpsilonGC
ZGC is a scalable low-latency garbage collector.

- GC pause times should not exceed 10ms
- Handle heaps ranging from relatively small (a few hundreds of mega) to very large (many terabytes) in size
- No more than 15% application throughput reduction compared to using G1
- Initially supported by Linux/x64 platforms

```
-XX:+UnlockExperimentalVMOptions
-XX:+UseZGC
```

https://openjdk.java.net/jeps/333
Removed

- JEP 320: Java EE
- JAX-WS (Java API for XML-Based Web Services)
- JAXB (Java Architecture for XML Binding)
- JAF (JavaBean Activation Framework)
- CommonAnnotations
Removed

- JEP 320: Java EE
- JAX-WS (Java API for XML-Based Web Services)
- JAXB (Java Architecture for XML Binding)
- JAF (JavaBean Activation Framework)
- Common Annotations

Maven™

- com.sun.xml.ws : jaxws-ri
- com.sun.xml.bind : jaxb-ri
- wsgen and wsimport: com.sun.xml.ws : jaxws-tools
- schemagen and xjc: com.sun.xml.bind : jaxb-jxc
- com.sun.xml.bind : jaxb-xjc
- javax.xml.ws : jaxws-api
- javax.xml.bind : jaxb-api
- javax.activation : javax.activation-api
- javax.annotation : javax.annotation-api
Removed

- JEP 320: Java EE
- JAX-WS (Java API for XML-Based Web Services)
- JAXB (Java Architecture for XML Binding)
- JAF (JavaBean Activation Framework)
- Common Annotations

- JEP 320: CORBA Modules
- RMI-IIOP
- Java IDL
- JTA (Java Transaction API)*
- J2EE Activity Service for Extended Transactions
Removed

- JEP 320: Java EE
- JAX-WS (Java API for XML-Based Web Services)
- JAXB (Java Architecture for XML Binding)
- JAF (JavaBean Activation Framework)
- Common Annotations

Depreciated

- JEP 335: Nashorn JavaScript Engine
  - jdk.scripting.nashorn
  - jdk.scripting.nashorn.shell
  - jjs

- JEP 320: CORBA Modules
  - RMI-IIOP
  - Java IDL
  - JTA (Java Transaction API)*
  - J2EE Activity Service for Extended Transactions
Removed

- JEP 320: Java EE
  - JAX-WS (Java API for XML-Based Web Services)
  - JAXB (Java Architecture for XML Binding)
  - JAF (JavaBean Activation Framework)
  - Common Annotations

- JEP 320: CORBA Modules
  - RMI-IIOP
  - Java IDL
  - JTA (Java Transaction API)*
  - J2EE Activity Service for Extended Transactions

Deprecated

- JEP 335: Nashorn JavaScript Engine
  - jdk.scripting.nashorn
  - jdk.scripting.nashorn.shell
  - jjs

- JEP 336: Pack200 Tools and API
  - pack200
  - unpack200
ABOUT DOCKER

Almost overnight, Docker has become the de facto standard that developers and system administrators use for packaging, deploying, and running distributed and cloud native applications. It provides tools for simplifying DevOps by enabling developers to create templates called images that can be used to create lightweight virtual machines called containers, which include their applications and all of their applications’ dependencies. These lightweight virtual machines can be promoted through testing and production environments where sysadmins deploy and run them.

Docker makes it easier for organizations to automate infrastructure, isolate applications, maintain consistency, and improve resource utilizations. Like the popular version control software Git, Docker has a social aspect in that developers and sysadmins can share their images via Docker Hub. Docker is an open source solution that is available as the free Community Edition or the subscription based Enterprise Edition for multiple platforms. Docker runs natively on Linux since Docker was originally built on Linux containers but it also works on Mac and Windows. Many enterprise tools have also grown up around Docker to make it easier to manage and orchestrate complex distributed and clustered applications architectures.

DOCKER ARCHITECTURE

Docker utilizes a client-server architecture and a remote API to manage and create Docker containers and images. Docker containers are created from Docker images. The relationship between containers and images are analogous to the relationship between objects and classes in object-oriented programming, where the image describes the container and the container is a running instance of the image.

Docker
Images
A recipe or template for creating Docker containers. It includes the steps for installing and running the necessary software

Docker
Container
Like a tiny virtual machine that is created from the instructions found within the Docker image

Docker
Client
Command-line utility or other tool that takes advantage of the Docker API (docs.docker.com/reference/api/docker_remote_api) to communicate with a Docker daemon

Docker
Host
A physical or virtual machine that is running a Docker daemon and contains cached images as well as runnable containers created from images

Docker
Registry
A repository of Docker images that can be used to create Docker containers. Docker Hub (hub.docker.com) is the most popular social example of a Docker repository.

Docker
Machine
A utility for managing multiple Docker hosts, which can run locally in VirtualBox or remotely in a cloud hosting service such as Amazon Web Services, Microsoft Azure, Google Cloud Platform, or Digital Ocean.