Migrate early, migrate often!
JDK release cadence strategies

Dan Heidinga
Eclipse OpenJ9 Project Lead
Interpreter Lead, IBM Runtimes
@danheidinga
DanHeidinga

Theresa Mammarella
Eclipse OpenJ9 Software Developer
@t_mammarella
theresa-m
Who am I?

- Theresa Mammarella
- Software developer on Runtimes at IBM, OpenJ9
- Computer Engineering at University of British Columbia
Important disclaimers

- THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
- WHILST EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.
- ALL PERFORMANCE DATA INCLUDED IN THIS PRESENTATION HAVE BEEN GATHERED IN A CONTROLLED ENVIRONMENT. YOUR OWN TEST RESULTS MAY VARY BASED ON HARDWARE, SOFTWARE OR INFRASTRUCTURE DIFFERENCES.
- ALL DATA INCLUDED IN THIS PRESENTATION ARE MEANT TO BE USED ONLY AS A GUIDE.
- IN ADDITION, THE INFORMATION CONTAINED IN THIS PRESENTATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM, WITHOUT NOTICE.
- IBM AND ITS AFFILIATED COMPANIES SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.
- NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:
  - CREATING ANY WARRANT OR REPRESENTATION FROM IBM, ITS AFFILIATED COMPANIES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS
Outline

- The brave new world of Java
- Always be migrating
- Continuous migration strategies
Historical Java release model

- Java 8: 8 years, 6 months
- Java 7: 3 years, 9 months
- Java 6: 6 years, 4 months
Brave new world

Java 9  
(09/2017)

Java 10  
(03/2018)

Java 11  
(09/2018)

Java 12  
(03/2019)

Java 17  
(09/2021)
6 month release cadence

Java 9 (09/2017)
Java 10 (03/2018)
Java 11 (09/2018)
Java 12 (03/2019)
Java 17 (09/2021)
LTS = Long Term Support?
LTS: up to the community

Java 9
(09/2017)

Java 10
(03/2018)

Java 11
LTS
(09/2018)

Java 12
(03/2019)

Java 17
LTS
(09/2021)
Prebuilt OpenJDK binaries

adoptopenjdk.net

Docker Hub

hub.docker.com/u/adoptopenjdk
Support Levels

... bugs will be fixed on a “best effort” basis ...

... for higher levels of assurance you should contact commercial companies offering support ...
IBM Support for OpenJDK with Eclipse OpenJ9

IBM **Advanced** Support for **Runtime Frameworks**
- Web Frameworks
- Key module dependencies*

IBM **Foundation** Support for **Runtimes**
- Language Runtime
- Developer and Monitoring tools
- Docker and Kubernetes support

* Supported module list is expected to grow over time.
Always be migrating

“eat the elephant a bite at a time”

- The gap between LTS releases is HUGE
- How can an application keep up?
  - Move to a stable base
  - Track changes in 6 month pieces
Migrate to a stable base: Java 8

“the best time to plant a tree is either 20 years ago or today”

- Migration 6-7-8 is mostly smooth
- Java 8 is the oldest stable base you should use, support through 2025
- By not upgrading you are leaving performance improvements on the table!
Track future changes: Java 9

Major ecosystem changes!

- Java Platform Module System makes internal APIs inaccessible
- API elements are deprecated for removal (Java EE, Corba and more)
- Version system has changed, URLClassLoader is replaced, Etc.
JEP 277: Enhanced Deprecation

@Deprecated
- Element is dangerous or a better alternative exists, use is discouraged

@Deprecated(since="9")
- Since(): Release in which API element was deprecated

@Deprecated(since="1.4", forRemoval=true)
- forRemoval(): Marked true if API element will be removed in a future release
Continuous integration pipeline

Start
- Setup (Java 8)
- Build (Java 8)
- Test (Java 8)
End
javac warnings

- Pay attention to warnings!
- Fixing them gets you clean on the current release
- Treat warnings as errors so build will fail if warnings are present

javac -Werror
Continuous migration strategies

Start (Java 8)  Setup (Java 8)  Build (Java 8)  Test (Java 8)  End

Start (Java 11)  Setup (Java 11)  Build (Java 11)  Test (Java 11)

20
Continuous migration strategies
jdeps Overview

JDEPS = Java Class Dependency Analyzer
- Java 8+ JDKs
- Analyzes the dependencies by class or package (default) level
- Not just for migrating to modularity!
Demo

github.com/theresa-m/migrateearly-demo

hub.docker.com/u/adoptopenjdk
jdeps docker stage

Start

Setup

Build

Migration checks

Test

End
jdeprscan Overview

- Java 9+ JDKs
- Scans jar/class files to identify deprecated API elements
- Deprecated elements may cause behavioral issues or be removed
Demo

github.com/theresa-m/migrateearly-demo

hub.docker.com/u/adoptopenjdk
Continuous migration strategies
Multi-release JAR files

- Java 9+ SDKs
- Enables JAR files to support multiple Java versions
- Combination of shared and version specific classes

sun.misc.Unsafe to java.lang.invoke.VarHandle
Multi-release JAR files

DemoApp.jar
  • migrate
    • early
      • Demo.class
      • DemoRunnable.class
  • META-INF
    • MANIFEST.MF

import sun.misc.Unsafe;
Multi-release JAR files

DemoApp.jar
- migrate
  - early
    - Demo.class
    - DemoRunnable.class
- META-INF
  - MANIFEST.MF

MultiReleaseDemoApp.jar
- migrate
  - early
    - Demo.class
    - VersionedCode.class
    - DemoRunnable.class
- META-INF
- MANIFEST.MF
  - versions
    - 9
    - VersionedCode.class
Summarize the strategies

- Expect to run parallel CI pipeline at least once every 6 months
- Run jdeps & jdeprscan as part of every build to prevent new regressions
- Use multi-release jars if no workaround for current and latest versions
Key points to take away

- The Java ecosystem is delivering innovation faster than ever before!
- The free lunch is over. Migrate early, migrate often
- Even on an LTS, use the JDK tools to prepare for the next migration
Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1  0  +1