OpenJDK and Eclipse OpenJ9

Give your Java applications a thrill !

Mark Stoodley

Project lead for Eclipse OpenJ9 and Eclipse OMR

Senior Software Developer, IBM Runtime Technologies

A Tale of Three Open Source Projects

OpenJDK

Java Class Libraries
(without Hotspot)

OpenJ9

Cross platform
Java Virtual Machine
(JVM)

Contributed in 2017
to Eclipse from
IBM SDK for Java
(and still built into it nightly!)

AdoptOpenJDK

Cross platform
Build, Test, and Certify
OpenJDK binaries
with openly maintained
build farm

A Tale of Three Open Source Projects

OpenJDK + OpenJ9 + AdoptOpenJDK

OpenJDK class libraries + Eclipse OpenJ9 JVM
developed in the open
built and tested **in the open** by AdoptOpenJDK
certified for production use

NOT about these commercial distributions…

OracleJDK

IBM SDK for Java

Well, mostly not…

Bold claim #1

OpenJDK with OpenJ9

is the best OpenJDK solution

for your Java workloads

Bold claim #2

AdoptOpenJDK

is the best place to get

your OpenJDK (with OpenJ9!) binaries

Bold claim #3

Eclipse OpenJ9 is

the best open source JVM project

in the open Java ecosystem

Outline

1. OpenJDK with OpenJ9
   – What's so great about it?

2. How to get OpenJDK with OpenJ9
   – How can you try it out?

3. OpenJ9 and the Java ecosystem
   – How does OpenJ9 relate?

Why use OpenJDK with OpenJ9?

- Easy to swap!
  - Install OpenJDK with OpenJ9 then point your apps at the new ‘java’
  - Has the same Java class libraries your application is used to
  - Best results: add a few simple command-line options
- ~30% faster server start-up (class sharing + cached JIT code (AOT))
- ~50% less physical memory use (heap&native memory management)
- “Designed for Cloud” configuration options:
  - Idle mode tuning for your less active JVMs (JIT, GC heuristics)
  - Faster ramp-up in CPU constrained environments (JIT heuristics)
- New JVM features (e.g. container support) with JDK8 and up

The importance of start-up (and ramp-up)

- Start-up: phase before first user transaction can be processed
  - Your customers get nothing during start-up
  - Faster start-up means more nimble elasticity and better developer productivity
  - When disaster strikes, faster start-up gets you back on your feet faster
- Ramp-up: phase where throughput not yet at steady-state
  - Your customers get reduced service, but transactions are completing
- As a Java user, you want fast start-up and fast ramp-up (of course!)
- As JVM implementer: different strategies needed for these two different phases

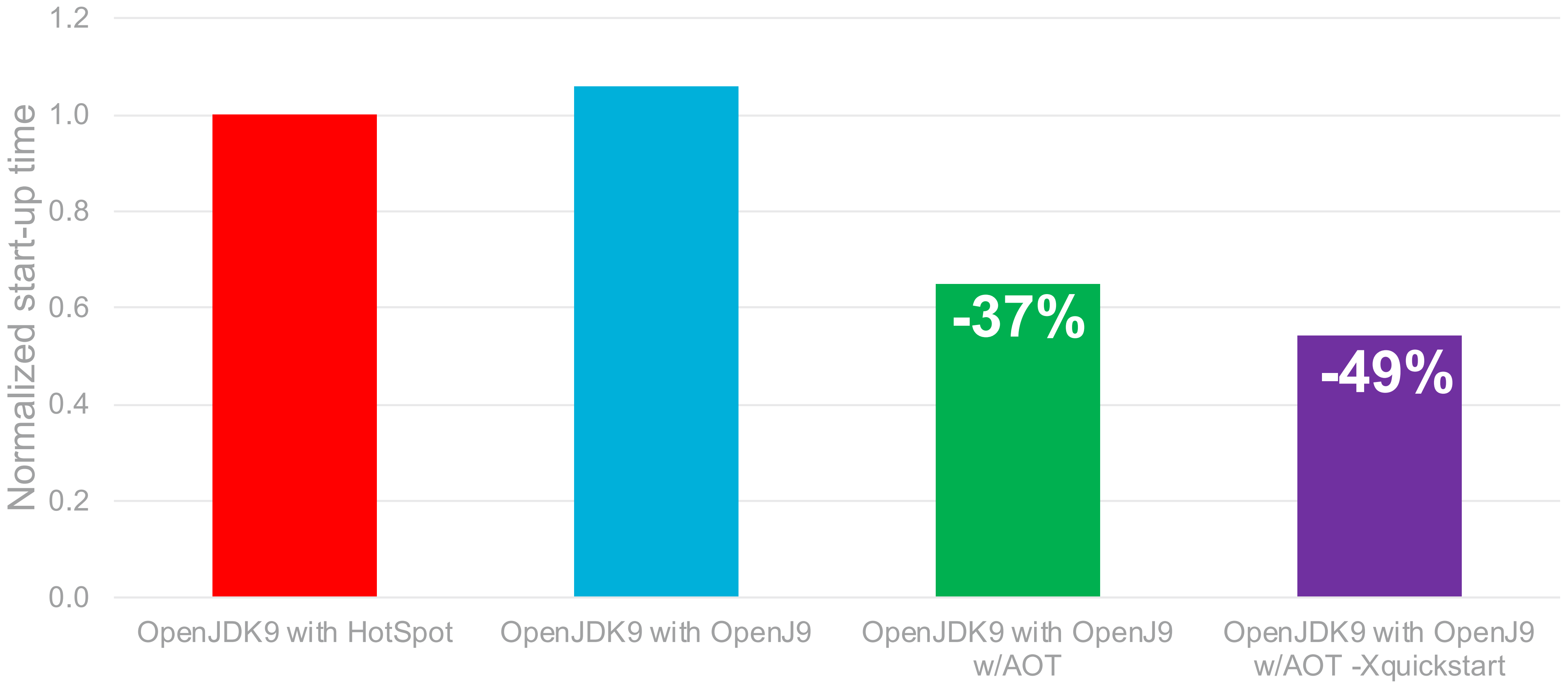
OpenJ9

~30% Faster Startup

OpenJ9

~50% Lower Physical Memory Use

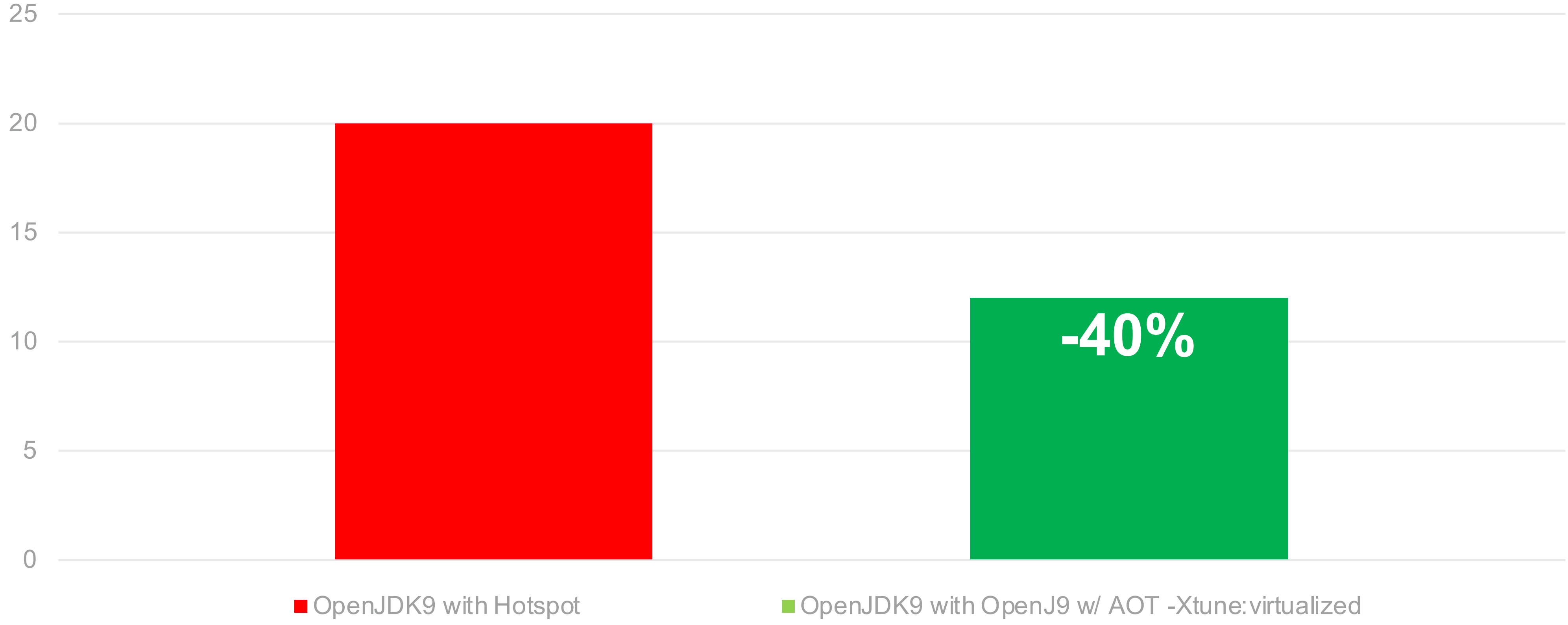
Faster start-up performance using OpenJ9 Enterprise workload: OpenLiberty with DayTrader3



Smaller Is Better

Benchmark: https://github.com/WASdev/sample.daytrader3
More details: https://github.com/eclipse/openj9-website/blob/master/benchmark/daytrader3.md

Faster start-up performance using OpenJ9: Developer experience: Eclipse Oxygen.2 IDE



Eclipse Oxygen.2 IDE Start-up Time

(seconds)

Smaller Is Better

Source: https://dzone.com/articles/hello-openj9-on-windows-i-didnt-expect-you-so-soon
Holger Voormann wrote “Hello OpenJ9 on Windows, I didn’t expect to see you so soon”. March 18, 2018

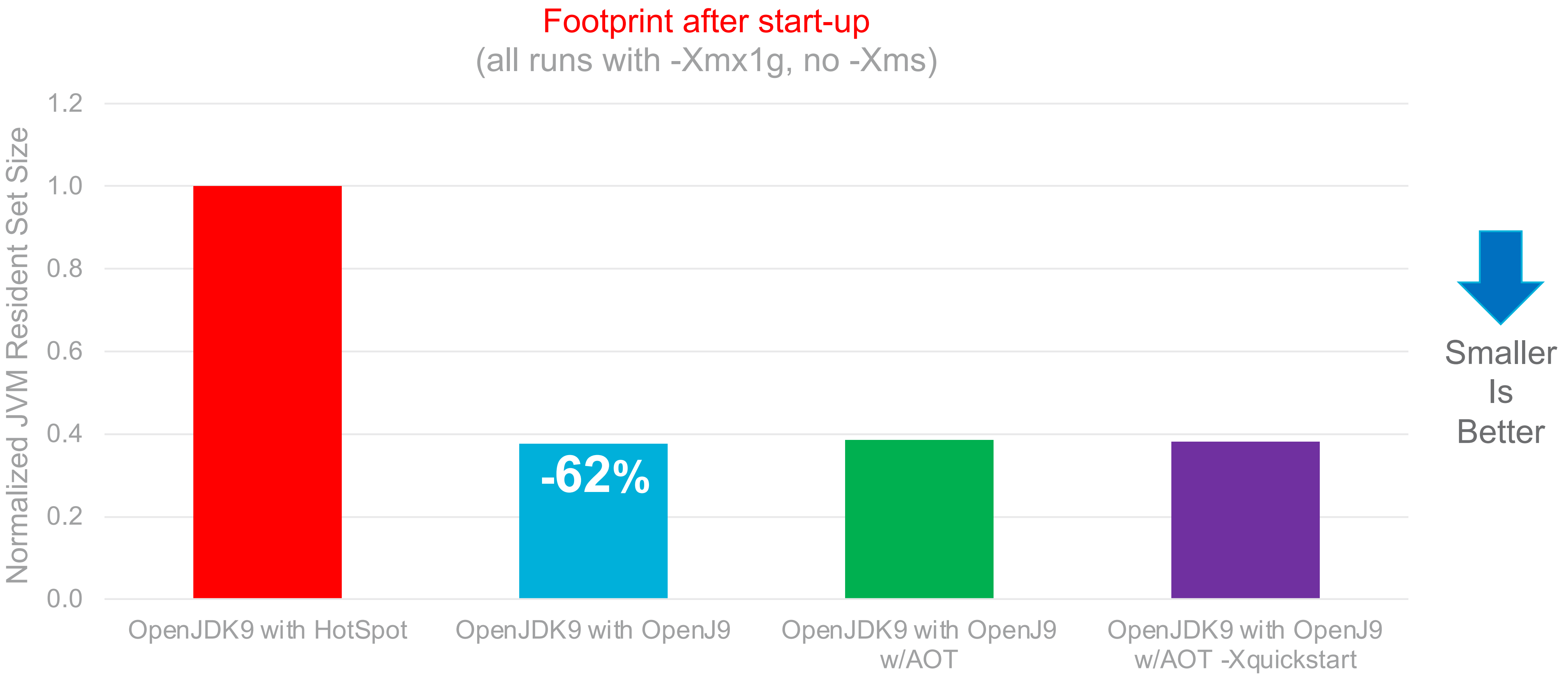
The importance of frugal physical memory use

- Physical memory use directly impacts density
  - Consume more: fewer things fit in a given memory envelope
  - Once you overflow, you need to buy more machines or VMs

- Unless your resources are free: higher density saves $$

- Capacity planning challenge
  - Imagine a JVM that always consumes available physical memory
  - That JVM *always* operates at worst case
  - What signals this JVM is actually getting close to limit?

Lower memory consumption using OpenJ9 Enterprise workload: OpenLiberty with DayTrader3

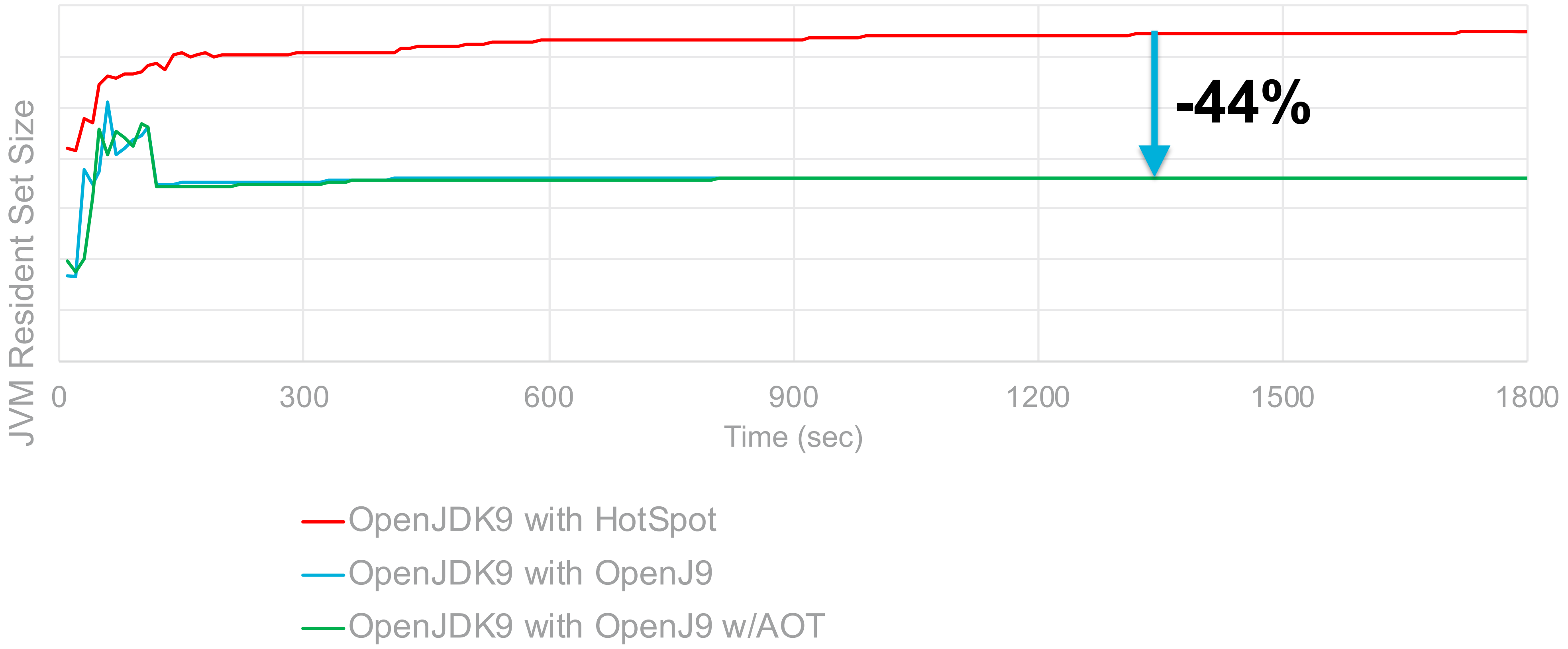


Benchmark: https://github.com/WASdev/sample.daytrader3
More details: https://github.com/eclipse/openj9-website/blob/master/benchmark/daytrader3.md

Lower memory consumption using OpenJ9
Enterprise workload: OpenLiberty with DayTrader3

Footprint under load

(all runs with -Xmx1g, no -Xms)



Smaller Is Better

Benchmark: https://github.com/WASdev/sample.daytrader3

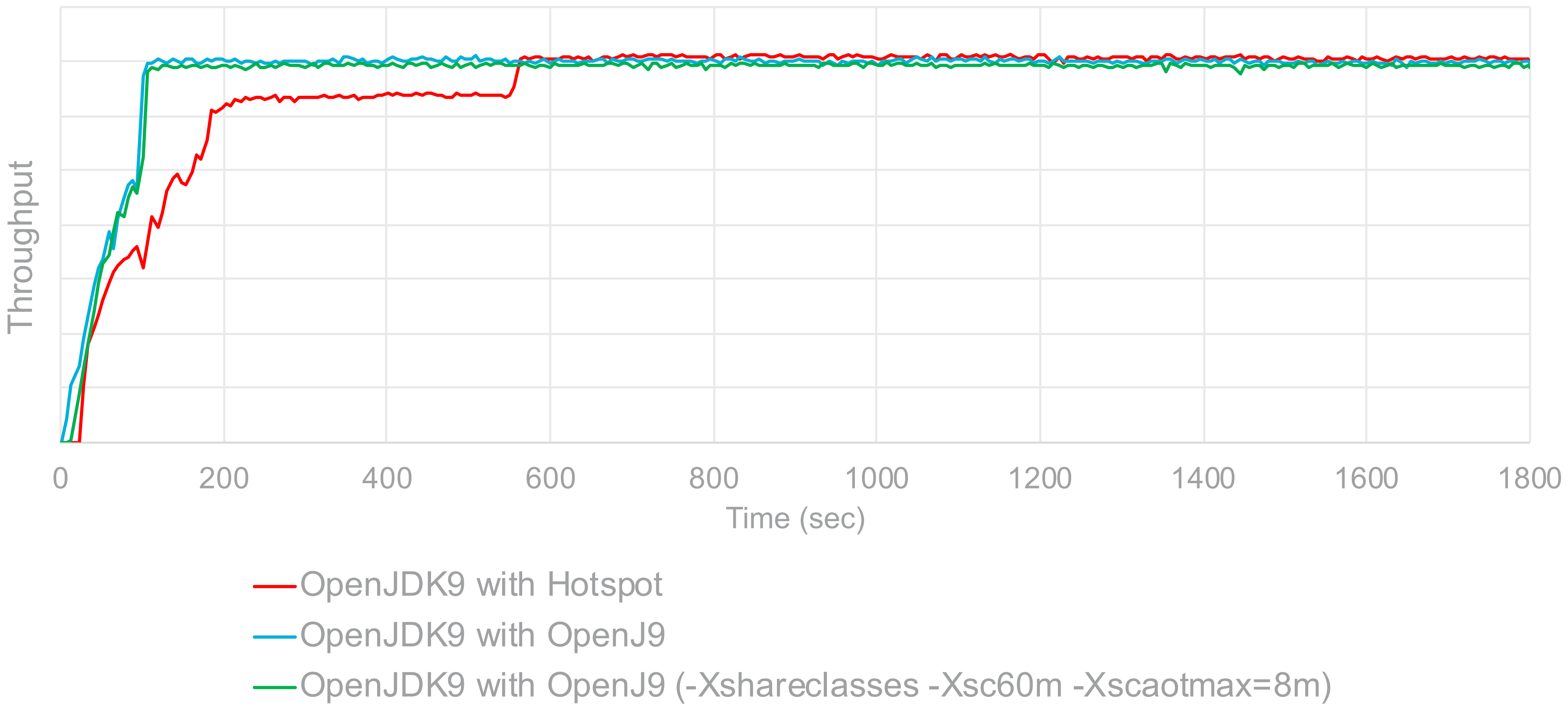
More details: https://github.com/eclipse/openj9-website/blob/master/benchmark/daytrader3.md

Sure, OpenJ9 starts fast and allocates frugally

But what about

throughput performance?

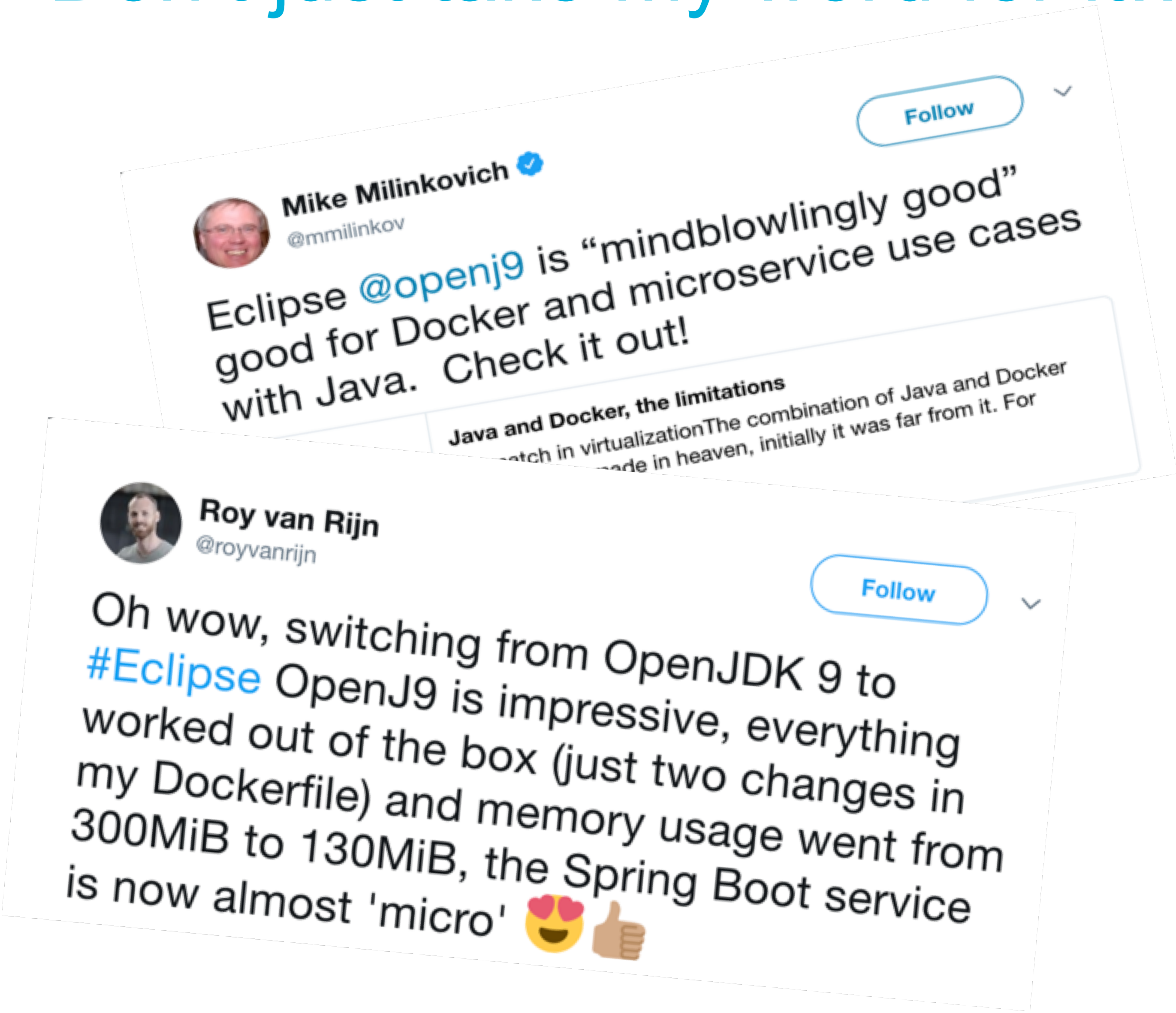
Comparable throughput using OpenJ9
Enterprise workload: OpenLiberty with DayTrader3



Bigger
Is
Better

Benchmark: https://github.com/WASdev/sample.daytrader3
More details: https://github.com/eclipse/openj9-website/blob/master/benchmark/daytrader3.md

Don’t just take my word for it…



Mike Milinkovich @mmilinkov

Follow

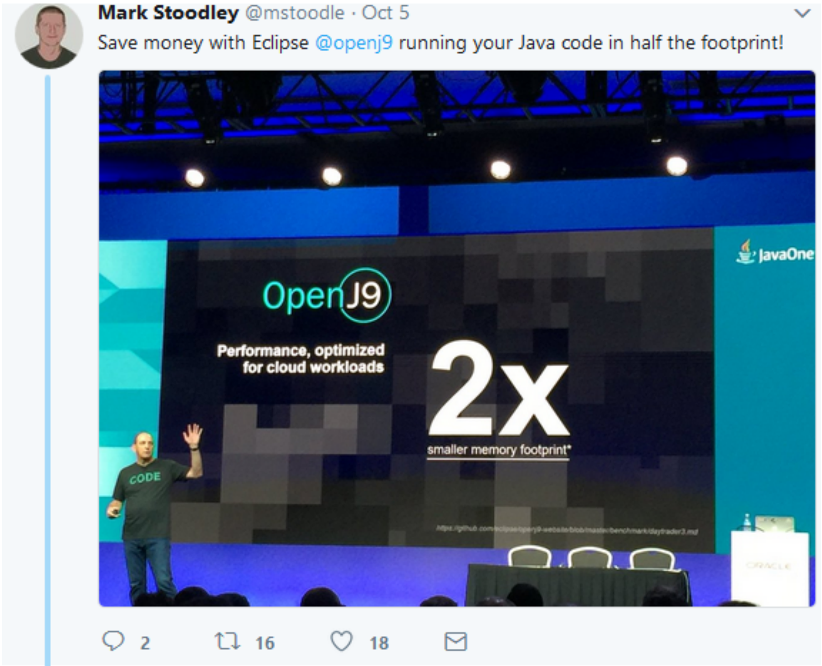
Eclipse @openj9 is “mindblowlingly good” good for Docker and microservice use cases with Java. Check it out!

Java and Docker, the limitations
atch in virtualizationThe combination of Java and Docker
ade in heaven, initially it was far from it. For

Roy van Rijn @royvanrijn

Follow

Oh wow, switching from OpenJDK 9 to #Eclipse OpenJ9 is impressive, everything worked out of the box (just two changes in my Dockerfile) and memory usage went from 300MiB to 130MiB, the Spring Boot service is now almost 'micro' 😍👍

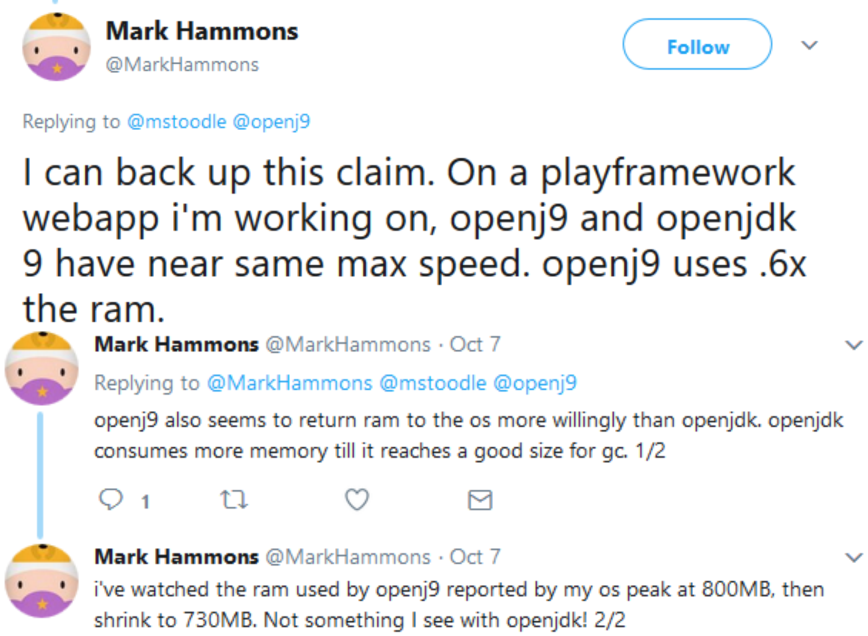


Mark Stoodley @mstoodle · Oct 5

Save money with Eclipse @openj9 running your Java code in half the footprint!

JavaOne
OpenJ9
Performance, optimized for cloud workloads
2x
smaller memory footprint*
CODE

2 16 18



Mark Hammons @MarkHammons

Follow

Replying to @mstoodle @openj9

I can back up this claim. On a playframework webapp i'm working on, openj9 and openjdk 9 have near same max speed. openj9 uses .6x the ram.

Mark Hammons @MarkHammons · Oct 7

Replying to @MarkHammons @mstoodle @openj9

openj9 also seems to return ram to the os more willingly than openjdk. openjdk consumes more memory till it reaches a good size for gc. 1/2

1

Mark Hammons @MarkHammons · Oct 7

i've watched the ram used by openj9 reported by my os peak at 800MB, then shrink to 730MB. Not something I see with openjdk! 2/2

A few examples of OpenJDK+OpenJ9 users

- Apache OpenWhisk: open source serverless platform
  - **"OpenWhisk actions over Eclipse OpenJ9 is ~25% faster than actions over Hotspot runtime."**
  - **"Eclipse OpenJ9 runtime based actions use 3x smaller memory footprint compared to Hotpot runtime."**
  - Source: https://medium.com/@ParamSelvam/apache-openwhisk-java-actions-on-eclipse-openj9-runtime-b21f1239d404
- Linkerd: open source network proxy designed to be deployed as a service mesh
  - **Release 1.4.5: "we've added experimental support for the OpenJ9 JVM."**
  - **"Preliminary tests with OpenJ9 exhibit a 3x reduction in startup time, a 40% reduction in memory footprint, and a 3x reduction in p99 latency**."
  - "…find a Linkerd+OpenJ9 Docker image at buoyantio/linkerd:1.4.5-openj9-experimental on Docker Hub."
  - Source: https://groups.google.com/forum/#!topic/linkerd-users/FE15LPAPEaA
- OpenJ9 and SpringBoot2 Microservices in Docker
  - **"OpenJ9 has 3x smaller footprint compared to OpenJDK with HotSpot or Oracle JRE…"**
  - **"OpenJ9 starts 30% faster than OpenJDK with HotSpot when Shared Classes are enabled."**
  - Source: https://medium.com/criciumadev/using-openj9-for-running-microservices-in-docker-ebb0b5da1e00

OpenJ9

Designed for the Cloud

Idle mode optimizations

- In data centers:
  - About 30% of VMs are comatose
  - About 50% of VMs are idle (active < 5% of the time)
  - Source: https://blog.anthesisgroup.com/zombie-servers-redux

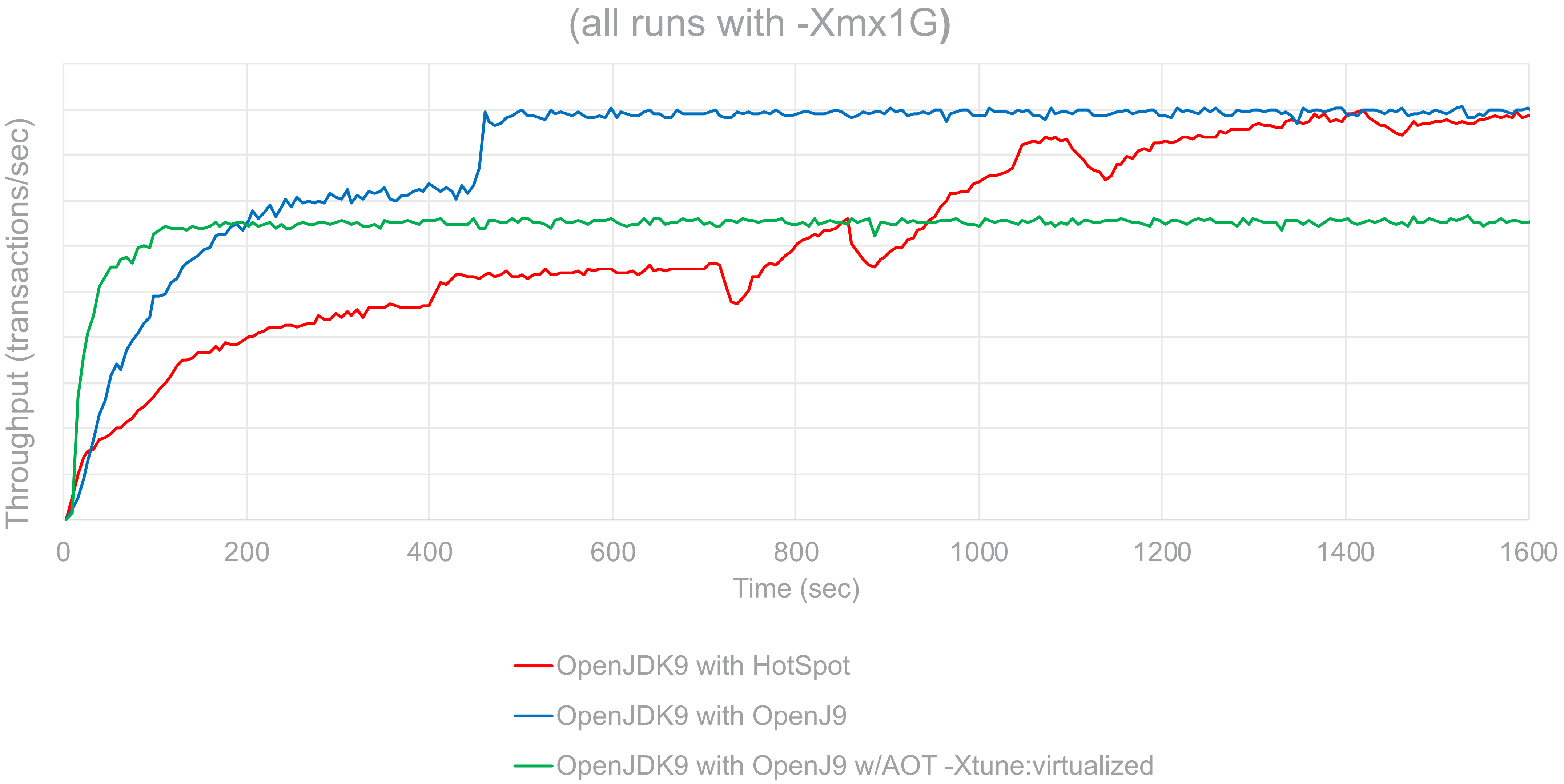
- OpenJ9 configuration option: `-XX:+IdleTuningGcOnIdle`
  - Compacts the heap and disclaims empty memory pages
  - Reduces sampling thread frequency: 55% fewer wake-ups than Hotspot
  - JIT compiler reduces optimization level (can be recompiled later!)

Ramping up in a CPU constrained environment

- Cloud and data center
  - Virtual machines with <= 1VCPU are not uncommon
  - “Hostile” environment for JVM: JIT compilation thread(s) must compete
  - Mismanaged? Slow ramp-up and potential for response time jitter

- OpenJ9 configuration option: `-Xtune:virtualized`
  - More conservative JIT optimization to reduce CPU pressure from JIT
  - With –Xshareclasses, use cached JIT code (AOT) more aggressively
  - Saves 20-30% compilation effort for ideally small throughput expense
  - Also some footprint reduction

Ramping-up with only one physical CPU core
Enterprise workload: OpenLiberty with DayTrader3



Sounds amazing!

How to try it out?

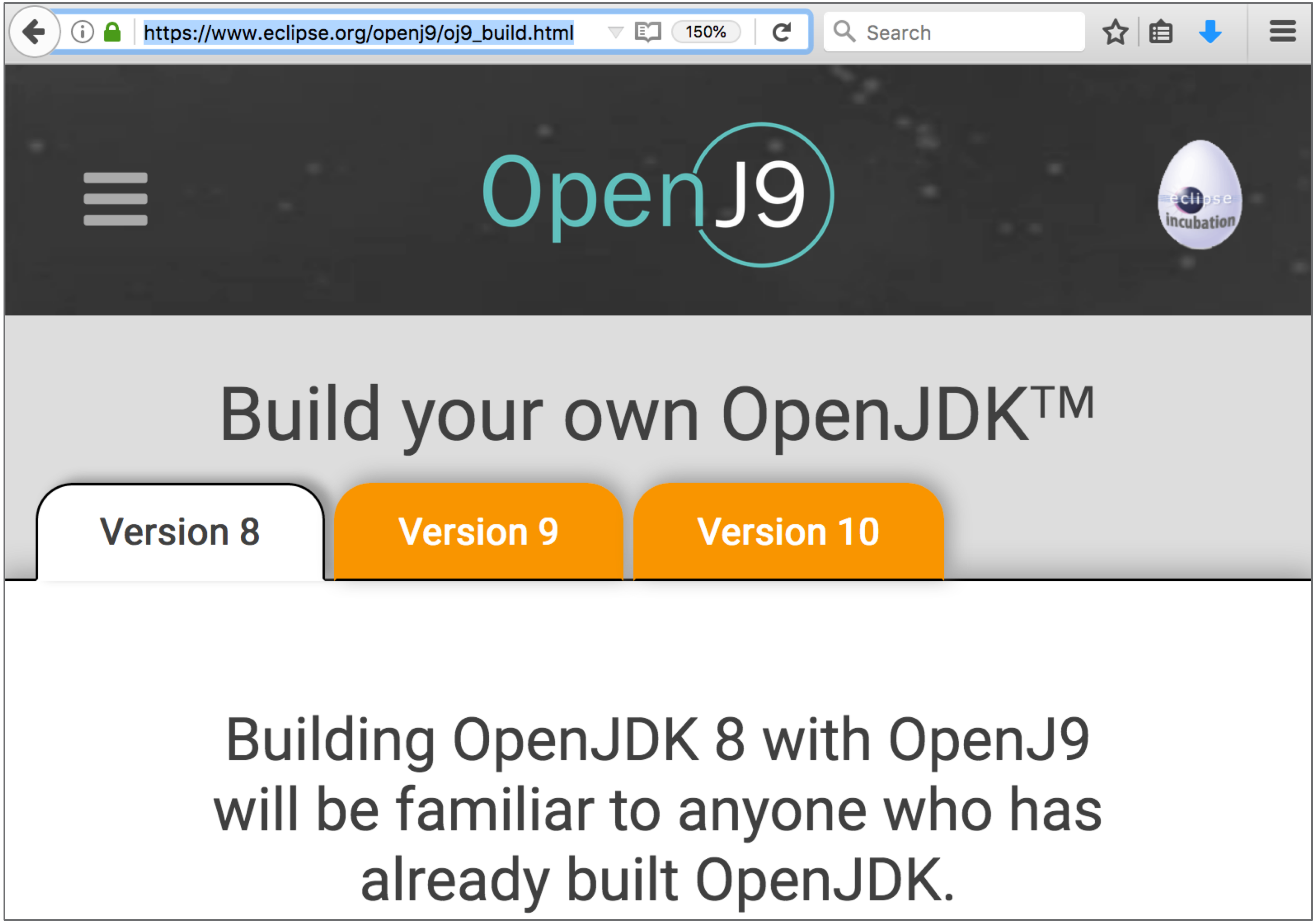
How to get OpenJDK with OpenJ9

A1.

Build it yourself from source ☺

See: https://www.eclipse.org/openj9/oj9_build.html for details

(it’s actually pretty easy: 4 major steps)



Build your own OpenJDK™

Building OpenJDK 8 with OpenJ9 will be familiar to anyone who has already built OpenJDK.

How to get OpenJDK with OpenJ9

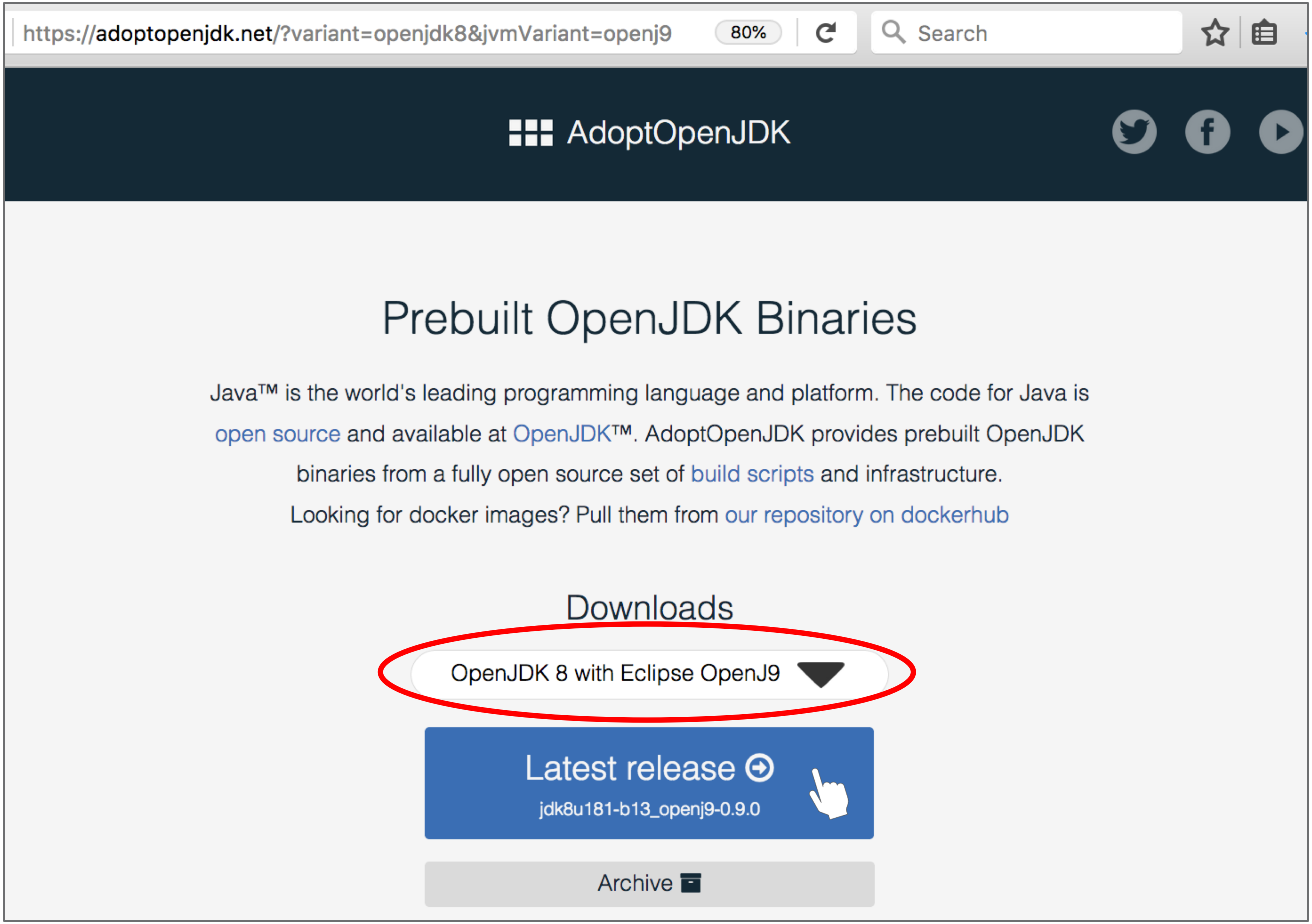
A2.

Download your platform binary from AdoptOpenJDK

Start at https://adoptopenjdk.net

Select an “OpenJDKx with OpenJ9” then press “Latest release”

(Currently x can be 8,9,10)



https://adoptopenjdk.net/?variant=openjdk8&jvmVariant=openj9
80%
Search
AdoptOpenJDK

Prebuilt OpenJDK Binaries

Java™ is the world's leading programming language and platform. The code for Java is open source and available at OpenJDK™. AdoptOpenJDK provides prebuilt OpenJDK binaries from a fully open source set of build scripts and infrastructure.

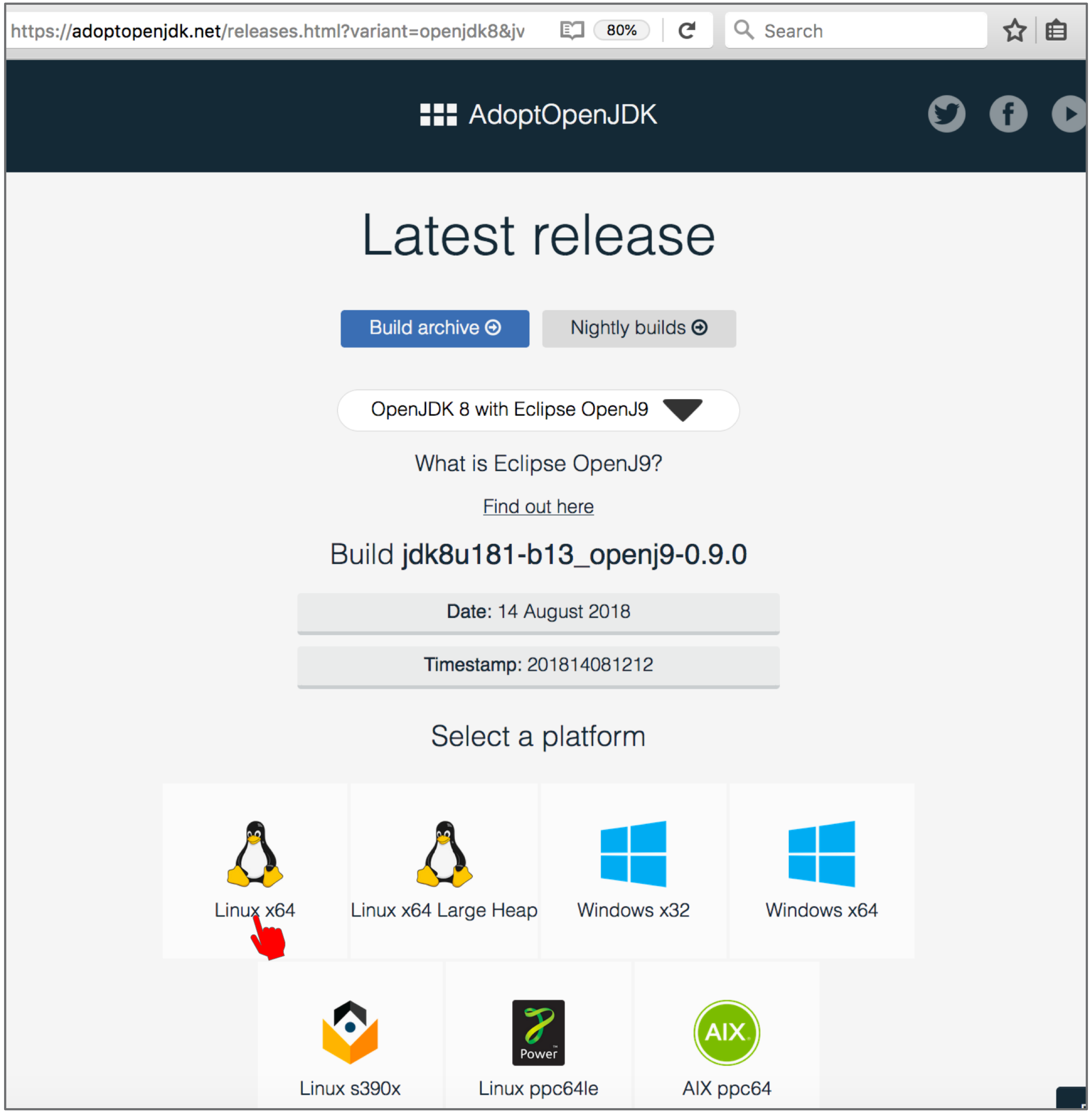
Looking for docker images? Pull them from our repository on dockerhub

Downloads

OpenJDK 8 with Eclipse OpenJ9

Latest release
jdk8u181-b13_openj9-0.9.0

Archive



Latest release

What is Eclipse OpenJ9?

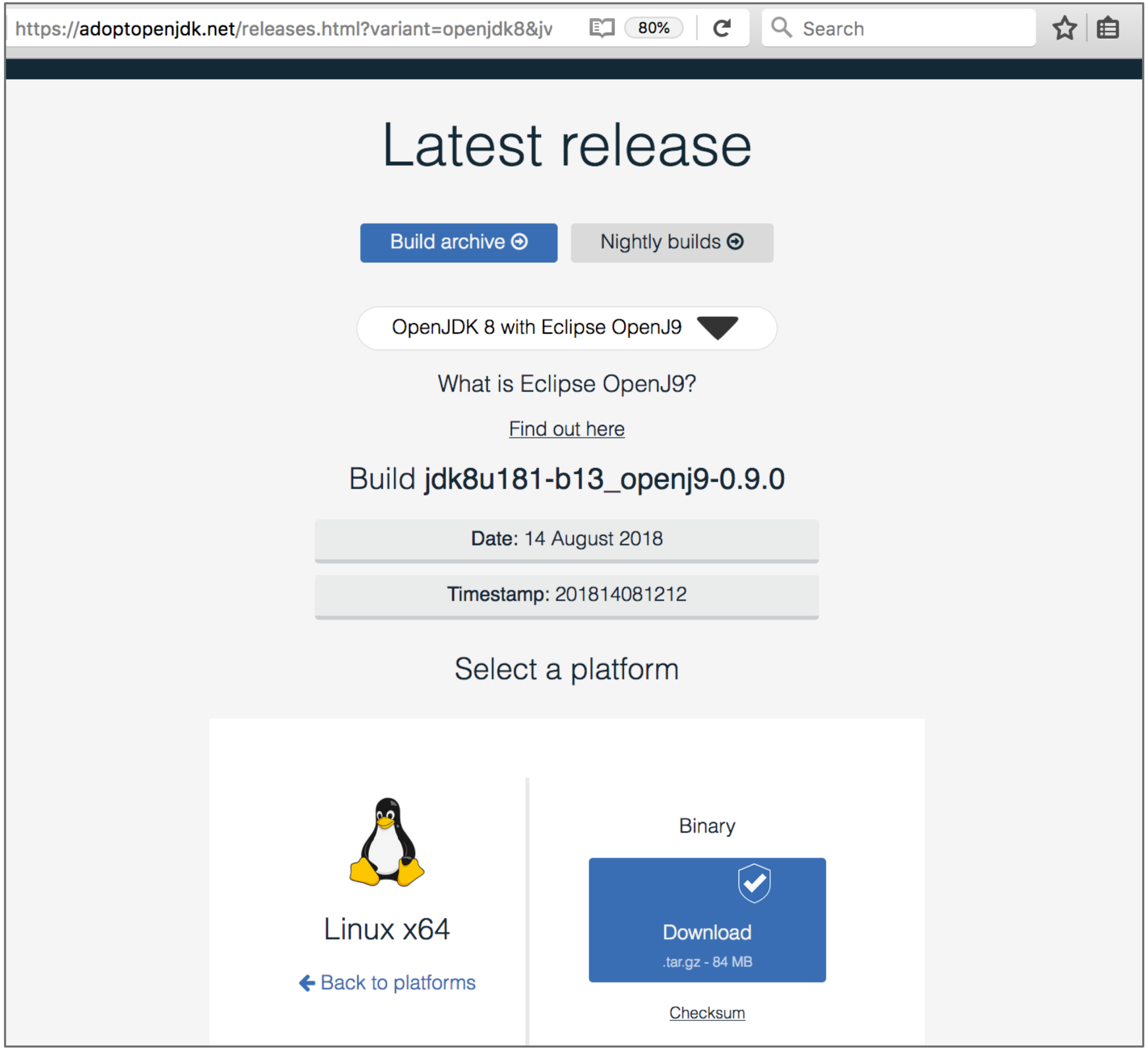
Find out here

Build **jdk8u181-b13_openj9-0.9.0**

Date: 14 August 2018

Timestamp: 201814081212

Select a platform



https://adoptopenjdk.net/releases.html?variant=openjdk8&jv

Latest release

Build archive ⊕   Nightly builds ⊕

OpenJDK 8 with Eclipse OpenJ9 ▼

What is Eclipse OpenJ9?

Find out here

Build **jdk8u181-b13_openj9-0.9.0**

Date: 14 August 2018

Timestamp: 201814081212

Select a platform

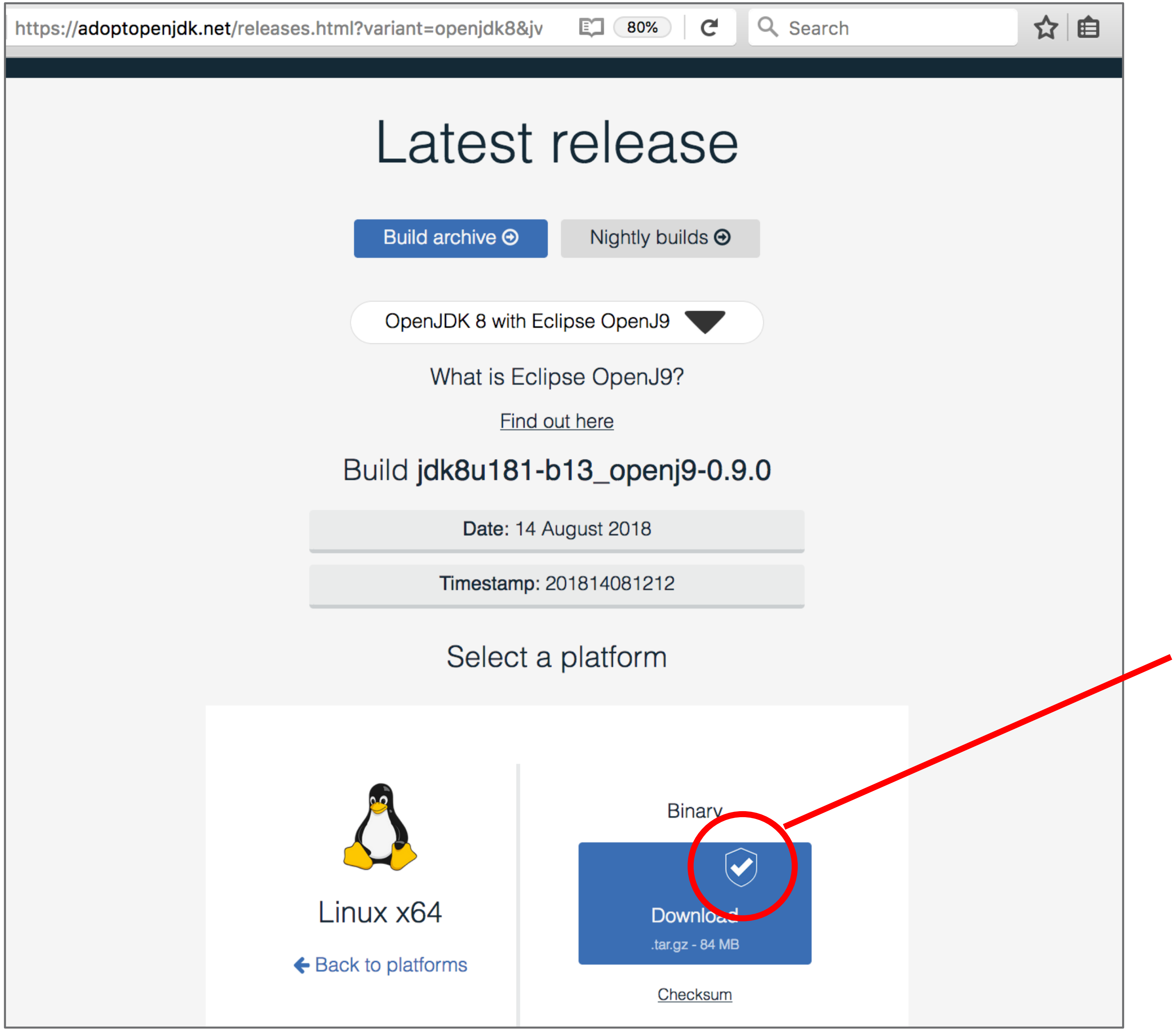
Linux x64

← Back to platforms

Binary

Download
.tar.gz - 84 MB

Checksum



Latest release

What is Eclipse OpenJ9?

Find out here

Build **jdk8u181-b13_openj9-0.9.0**

Date: 14 August 2018

Timestamp: 201814081212

Select a platform

Means JCK compliant, as tested by AdoptOpenJDK community!

How to get OpenJDK with OpenJ9

A3.

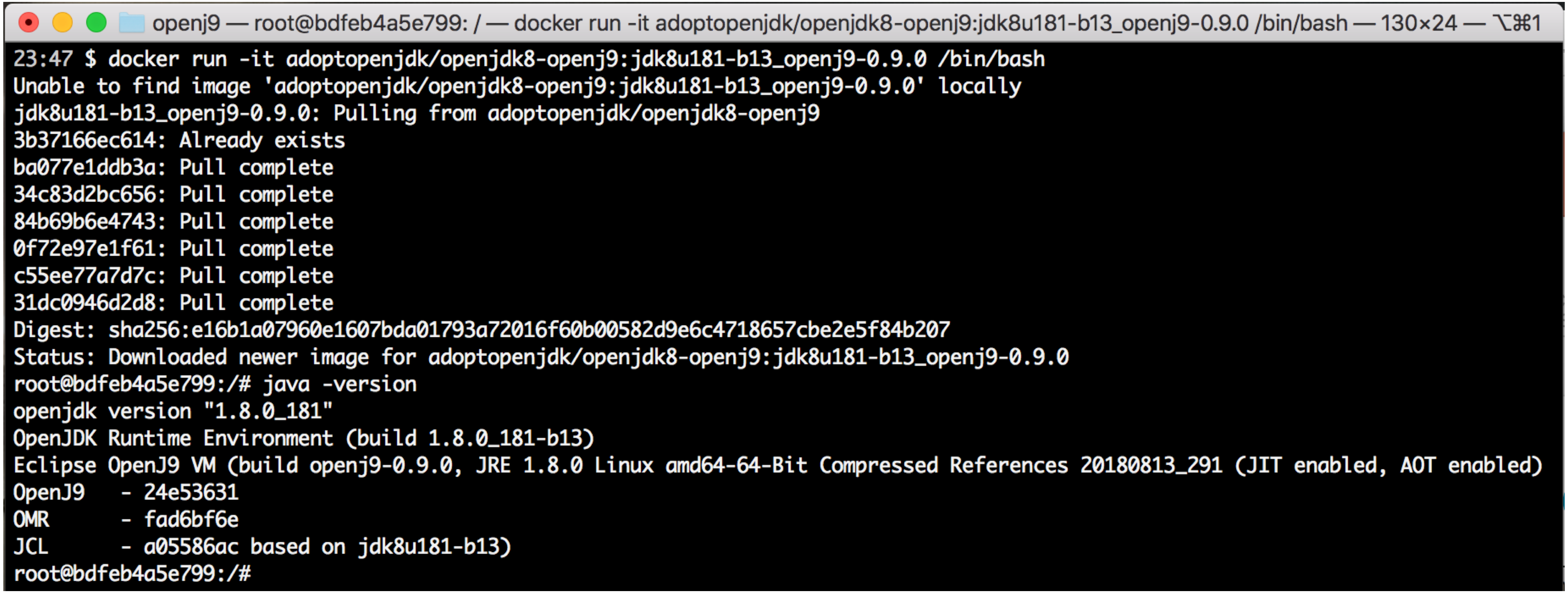
Pull a docker image from DockerHub

(built by AdoptOpenJDK)

See https://hub.docker.com/search/?q=openj9

adoptopenjdk/openjdkx-openj9 (Currently x=8,9.10)

Multi-arch docker images make it super easy



openj9 — root@bdfeb4a5e799: / — docker run -it adoptopenjdk/openjdk8-openj9:jdk8u181-b13_openj9-0.9.0 /bin/bash — 130×24 — ⌥⌘1

```
23:47 $ docker run -it adoptopenjdk/openjdk8-openj9:jdk8u181-b13_openj9-0.9.0 /bin/bash
Unable to find image 'adoptopenjdk/openjdk8-openj9:jdk8u181-b13_openj9-0.9.0' locally
jdk8u181-b13_openj9-0.9.0: Pulling from adoptopenjdk/openjdk8-openj9
3b37166ec614: Already exists
ba077e1ddb3a: Pull complete
34c83d2bc656: Pull complete
84b69b6e4743: Pull complete
0f72e97e1f61: Pull complete
c55ee77a7d7c: Pull complete
31dc0946d2d8: Pull complete
Digest: sha256:e16b1a07960e1607bda01793a72016f60b00582d9e6c4718657cbe2e5f84b207
Status: Downloaded newer image for adoptopenjdk/openjdk8-openj9:jdk8u181-b13_openj9-0.9.0
root@bdfeb4a5e799:/# java -version
openjdk version "1.8.0_181"
OpenJDK Runtime Environment (build 1.8.0_181-b13)
Eclipse OpenJ9 VM (build openj9-0.9.0, JRE 1.8.0 Linux amd64-64-Bit Compressed References 20180813_291 (JIT enabled, AOT enabled)
OpenJ9   - 24e53631
OMR      - fad6bf6e
JCL      - a05586ac based on jdk8u181-b13)
root@bdfeb4a5e799:/#
```

You can also find tags for: Latest, release, and nightly (“-nightly”) builds, default and slim (“-slim”) variants
Alpine (“-alpine”) for x86_64, Ubuntu 16.04 for x86_64, ppc64le, and s390x

How to get OpenJDK with OpenJ9

A4.

Use docker image as base for your own images

(built by AdoptOpenJDK)

See https://hub.docker.com/search/?q=openj9

adoptopenjdk/openjdkx-openj9 (Currently x=8,9.10)

Multi-arch docker images make it super easy

```
FROM adoptopenjdk/openjdk8-openj9:jdk8u181-b13_openj9-0.9.0
```

You can also find tags for: Latest, release, and nightly (“-nightly”) builds, default and slim (“-slim”) variants
Alpine (“-alpine”) for x86_64, Ubuntu 16.04 for x86_64, ppc64le, and s390x

You owe it to yourself to ask:

How are those binaries built?

How are they tested?

The AdoptOpenJDK Build Farm is three things:

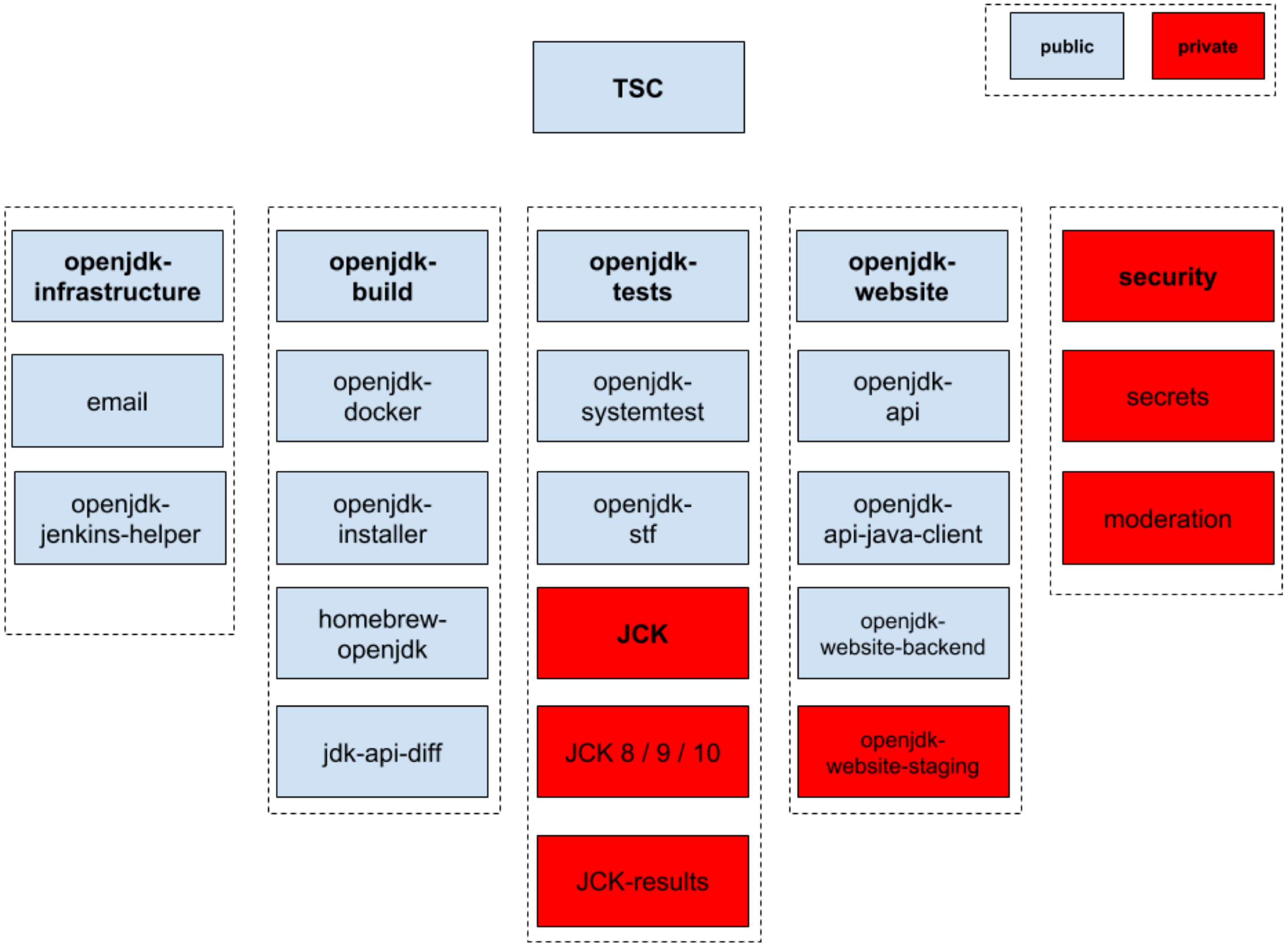
1. **Infrastructure as Code** - To host, build, test and deploy variants of OpenJDK (aka Java)! This infrastructure as code is designed to be usable by any person or organisation wishing to build a derivative build farm or parts of one.

2. **Professionally built OpenJDK binaries** - A place for end users to download professionally built and tested OpenJDK binaries.

3. **A Community of builders** - A place where those who build and test OpenJDK come together to share common code and practices.

Source: https://github.com/AdoptOpenJDK/TSC

AdoptOpenJDK is an Open Build Farm



public

private

TSC

openjdk-infrastructure

email

openjdk-jenkins-helper

openjdk-build

openjdk-docker

openjdk-installer

homebrew-openjdk

jdk-api-diff

openjdk-tests

openjdk-systemtest

openjdk-stf

JCK

JCK 8 / 9 / 10

JCK-results

openjdk-website

openjdk-api

openjdk-api-java-client

openjdk-website-backend

openjdk-website-staging

security

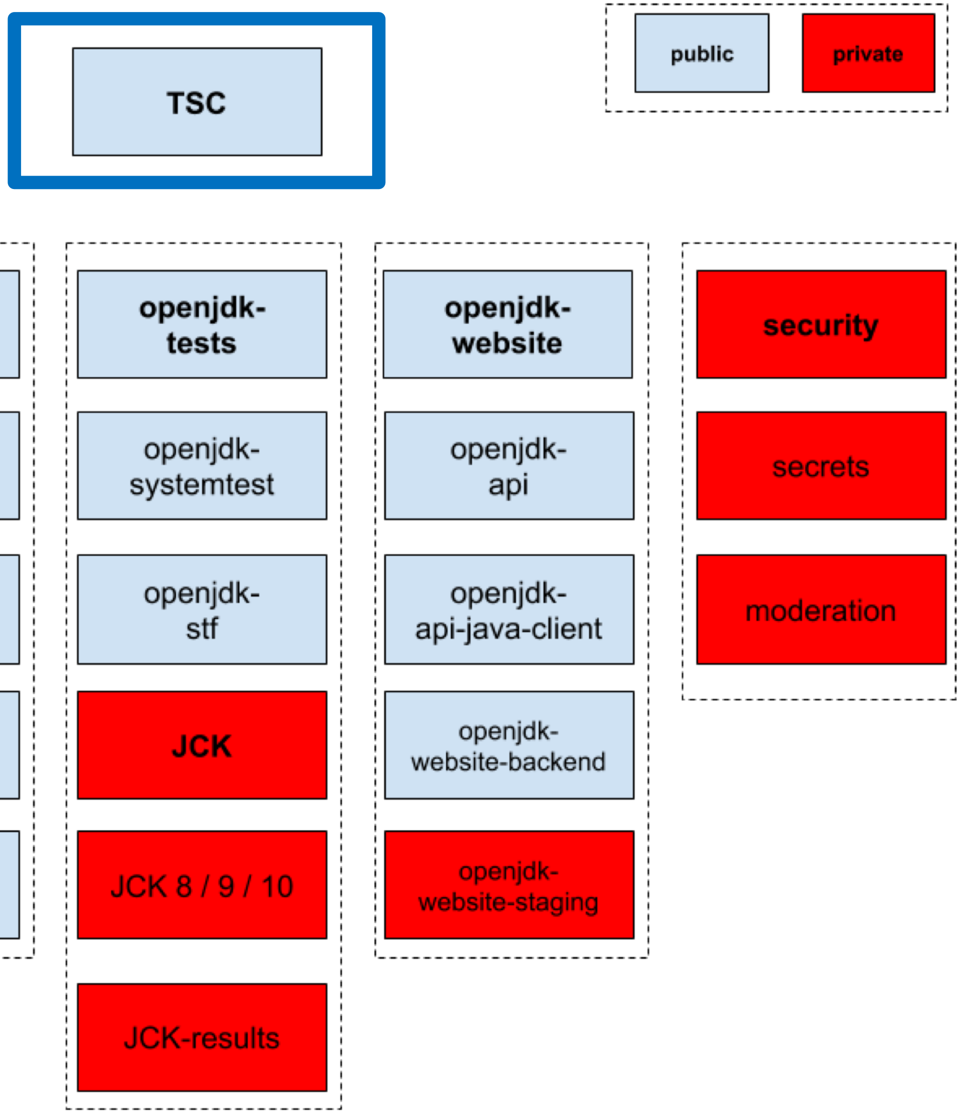
secrets

moderation

Source: https://github.com/AdoptOpenJDK/TSC/blob/master/images/Adopt_Build_Farm_Repo_Relationships.png

AdoptOpenJDK is an Open Build Farm

Vendor neutral technical steering committee



🟦 public 🟥 private

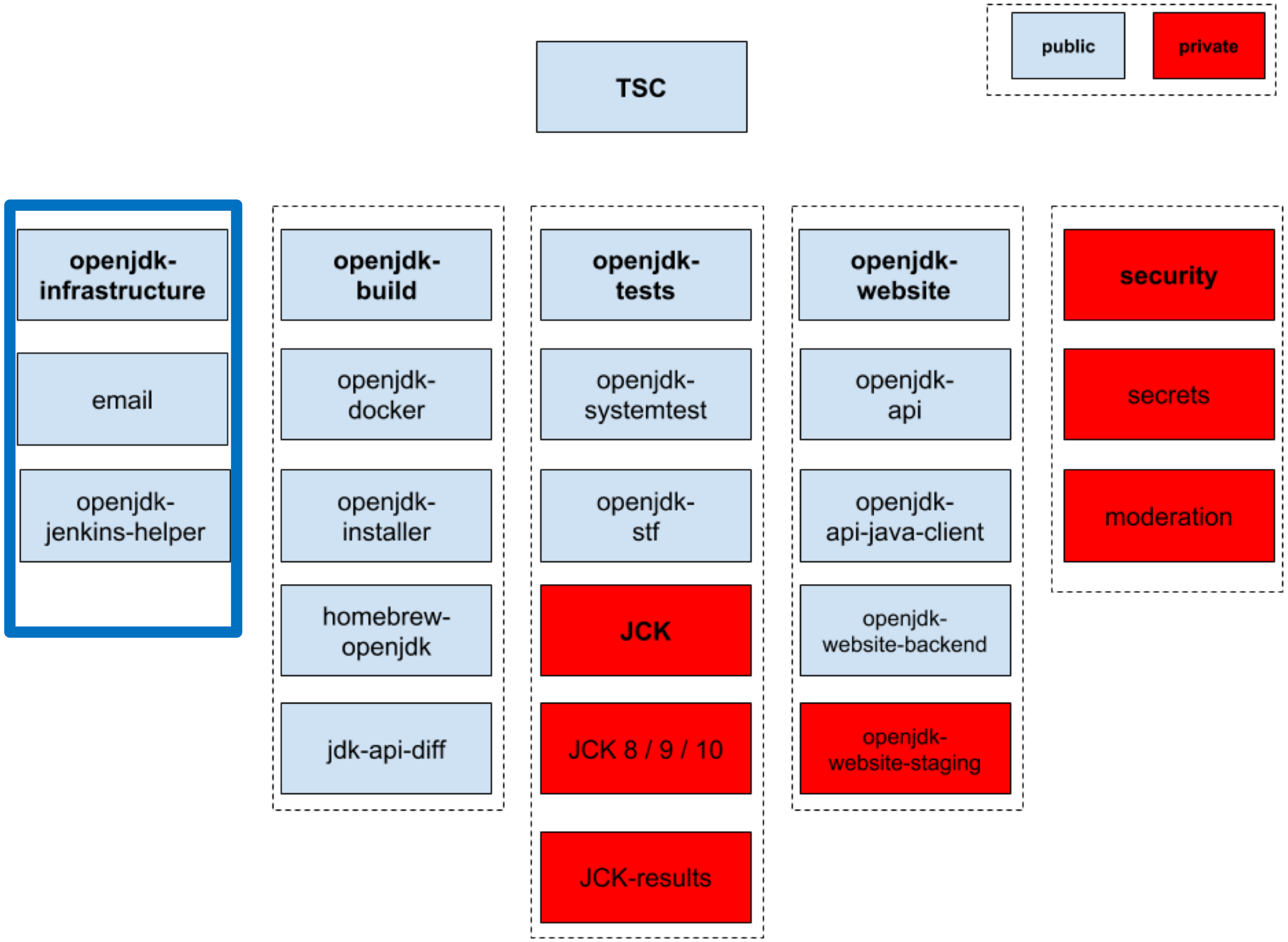
TSC

- **openjdk-infrastructure**
  - email
  - openjdk-jenkins-helper
- **openjdk-build**
  - openjdk-docker
  - openjdk-installer
  - homebrew-openjdk
  - jdk-api-diff
- **openjdk-tests**
  - openjdk-systemtest
  - openjdk-stf
  - 🟥 **JCK**
  - 🟥 JCK 8 / 9 / 10
  - 🟥 JCK-results
- **openjdk-website**
  - openjdk-api
  - openjdk-api-java-client
  - openjdk-website-backend
  - 🟥 openjdk-website-staging
- 🟥 **security**
  - 🟥 secrets
  - 🟥 moderation

Source: https://github.com/AdoptOpenJDK/TSC/blob/master/images/Adopt_Build_Farm_Repo_Relationships.png

AdoptOpenJDK is an Open Build Farm

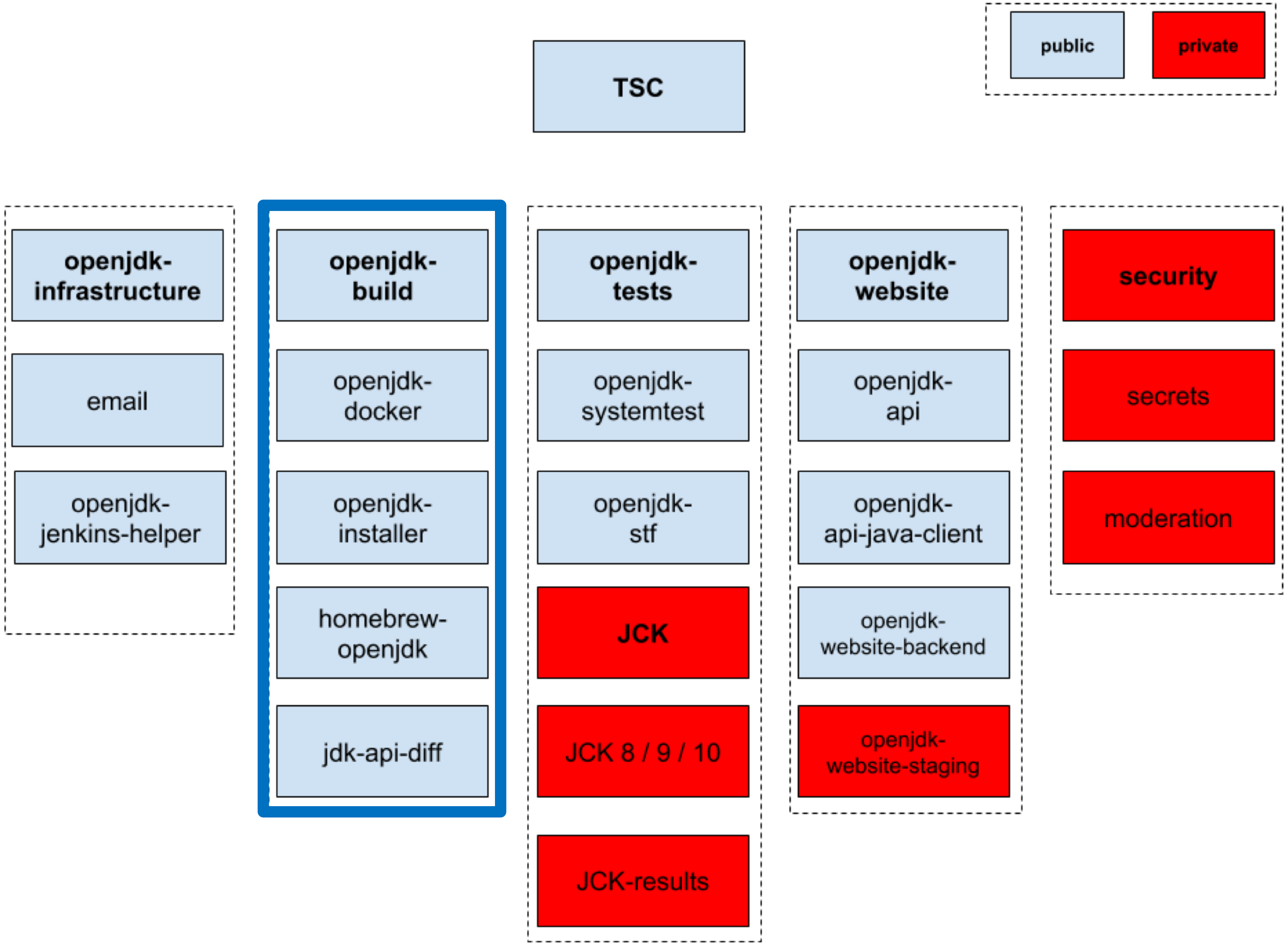
How infrastructure is managed



Source: https://github.com/AdoptOpenJDK/TSC/blob/master/images/Adopt_Build_Farm_Repo_Relationships.png

AdoptOpenJDK is an Open Build Farm

How to build OpenJDK releases for all platforms



public

private

TSC

openjdk-infrastructure

email

openjdk-jenkins-helper

openjdk-build

openjdk-docker

openjdk-installer

homebrew-openjdk

jdk-api-diff

openjdk-tests

openjdk-systemtest

openjdk-stf

JCK

JCK 8 / 9 / 10

JCK-results

openjdk-website

openjdk-api

openjdk-api-java-client

openjdk-website-backend

openjdk-website-staging

security

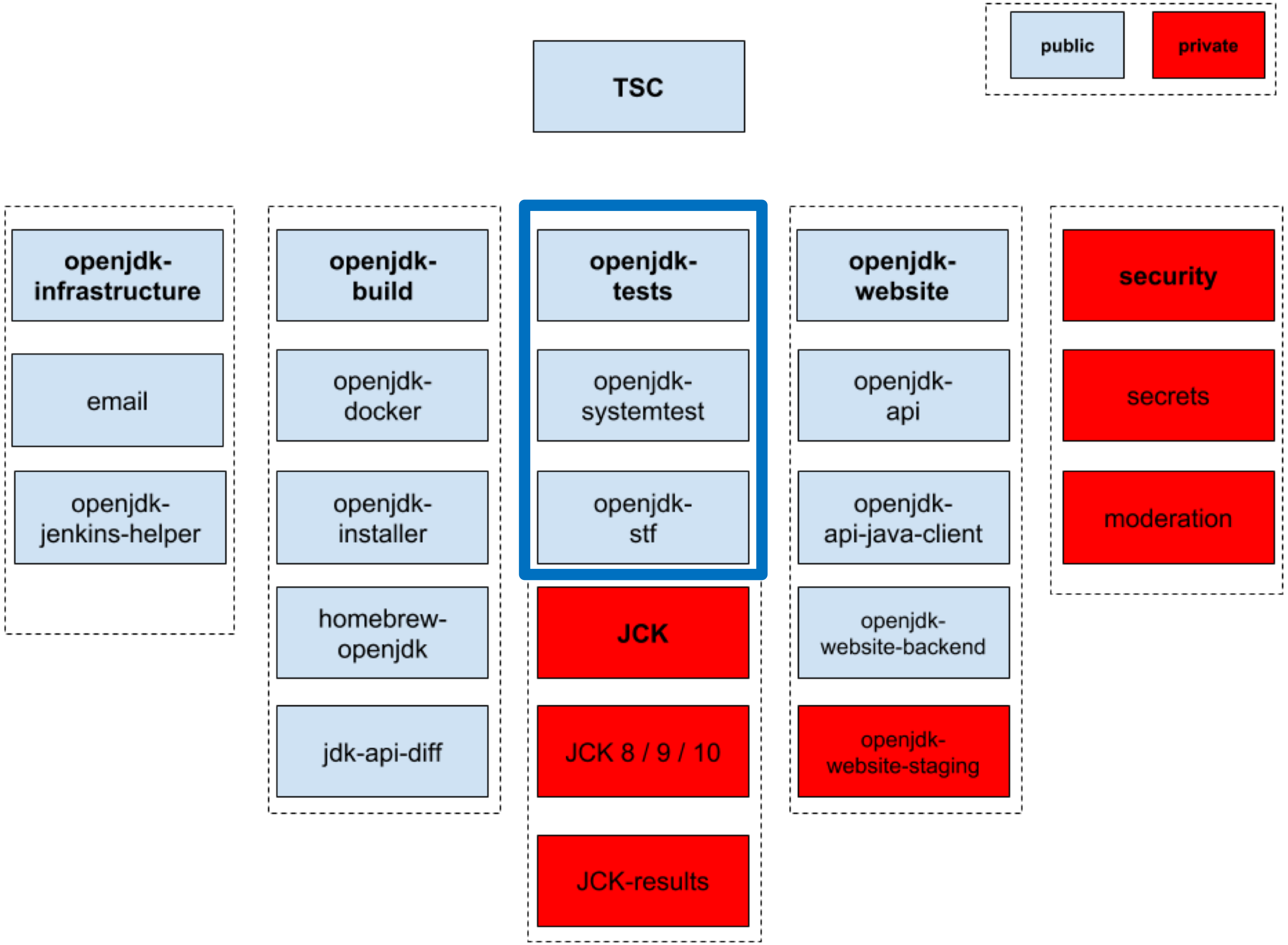
secrets

moderation

Source: https://github.com/AdoptOpenJDK/TSC/blob/master/images/Adopt_Build_Farm_Repo_Relationships.png

AdoptOpenJDK is an Open Build Farm

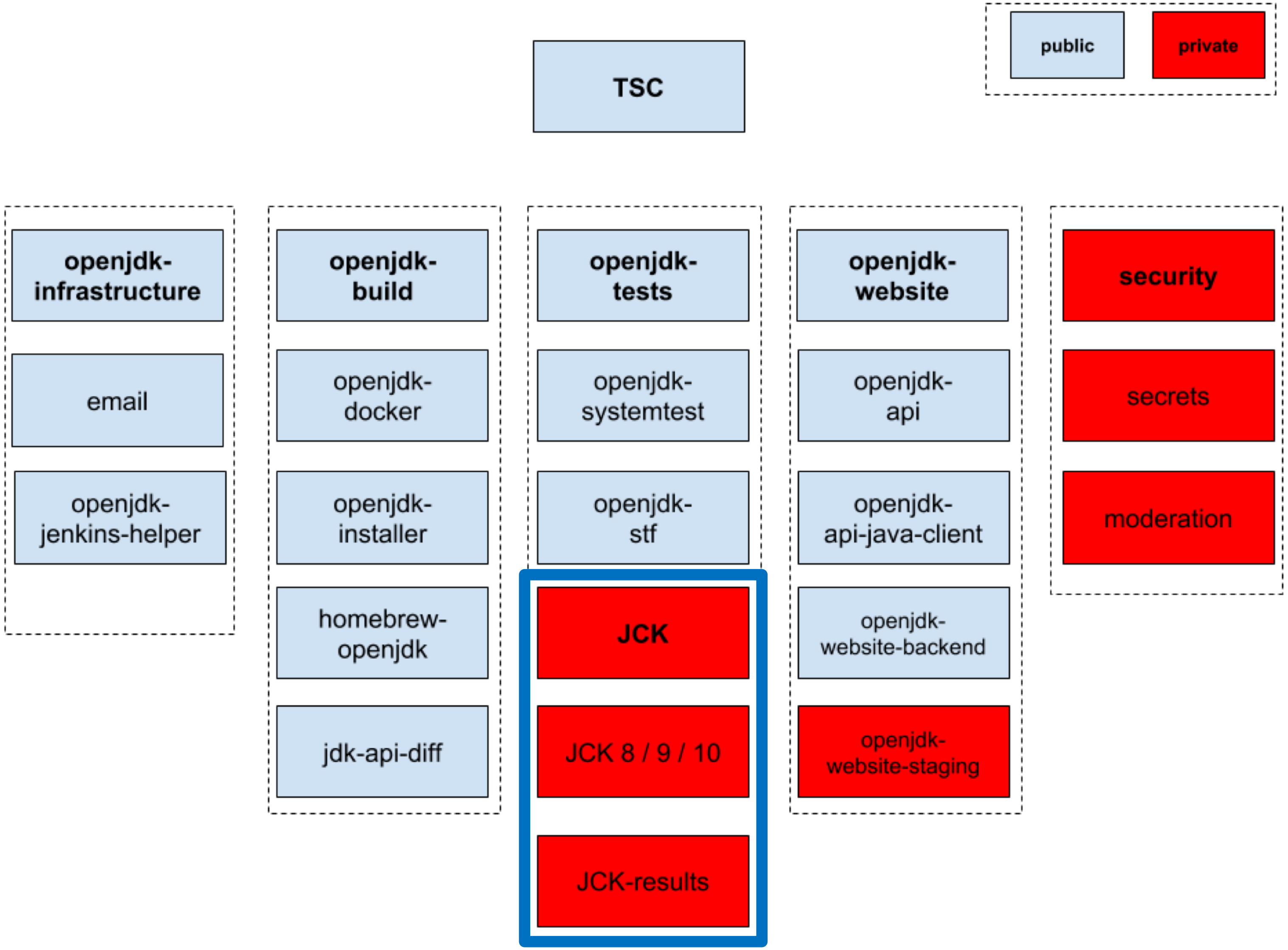
How to test OpenJDK releases to ensure high quality



Source: https://github.com/AdoptOpenJDK/TSC/blob/master/images/Adopt_Build_Farm_Repo_Relationships.png

AdoptOpenJDK is an Open Build Farm

Certify OpenJDK builds for all releases storing archived results



public | private

TSC

openjdk-infrastructure
email
openjdk-jenkins-helper

openjdk-build
openjdk-docker
openjdk-installer
homebrew-openjdk
jdk-api-diff

openjdk-tests
openjdk-systemtest
openjdk-stf
JCK
JCK 8 / 9 / 10
JCK-results

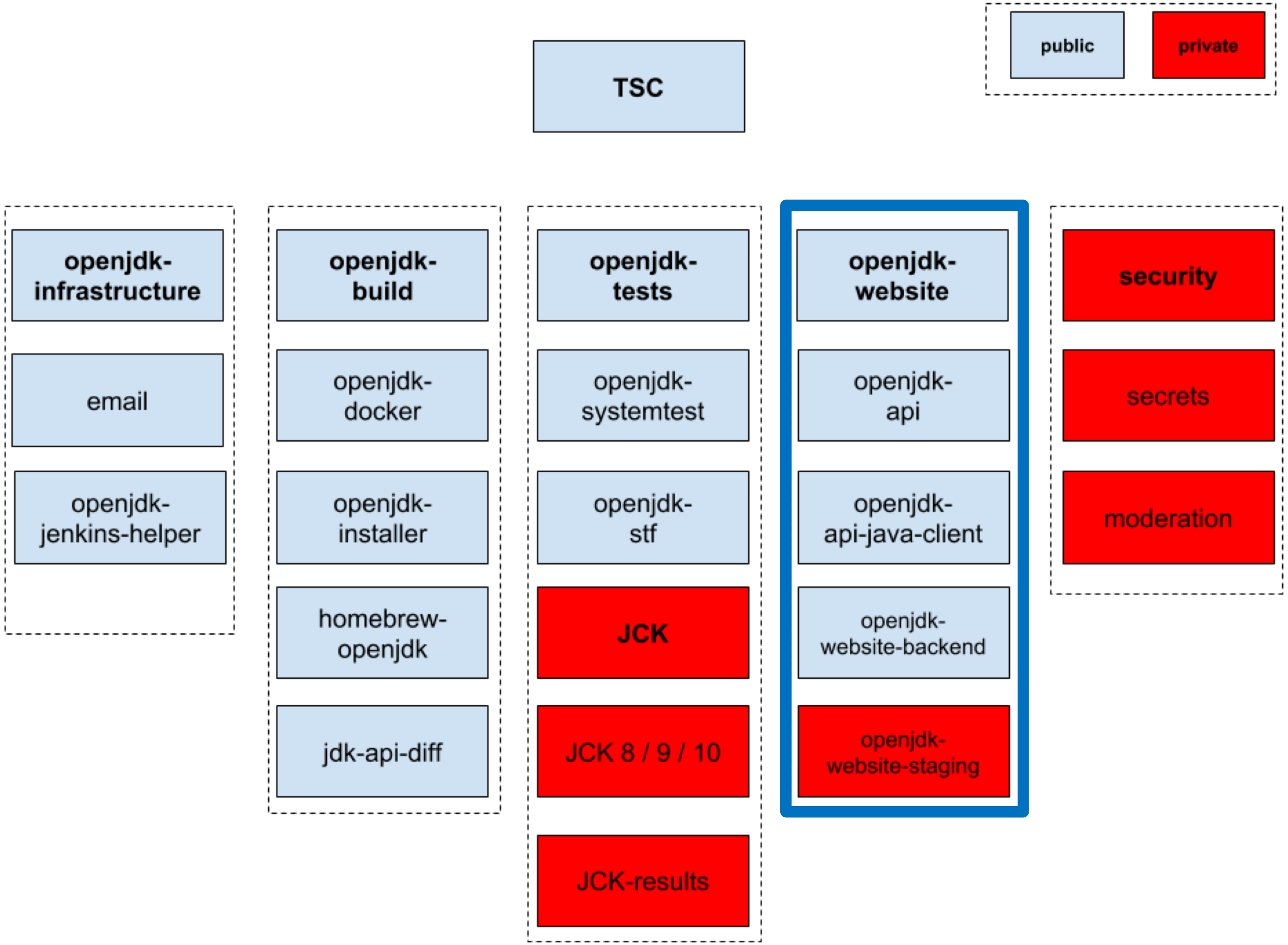
openjdk-website
openjdk-api
openjdk-api-java-client
openjdk-website-backend
openjdk-website-staging

security
secrets
moderation

Source: https://github.com/AdoptOpenJDK/TSC/blob/master/images/Adopt_Build_Farm_Repo_Relationships.png

AdoptOpenJDK Open Source Build Farm

Implements web site and Express.js REST API



public

private

TSC

openjdk-infrastructure
- email
- openjdk-jenkins-helper

openjdk-build
- openjdk-docker
- openjdk-installer
- homebrew-openjdk
- jdk-api-diff

openjdk-tests
- openjdk-systemtest
- openjdk-stf
- JCK
- JCK 8 / 9 / 10
- JCK-results

openjdk-website
- openjdk-api
- openjdk-api-java-client
- openjdk-website-backend
- openjdk-website-staging

security
- secrets
- moderation

Source: https://github.com/AdoptOpenJDK/TSC/blob/master/images/Adopt_Build_Farm_Repo_Relationships.png

Get involved at AdoptOpenJDK !

Want additional / different testing? Help to add it!

Want another platform? Update the scripts!

AdoptOpenJDK is 100% open source

and community driven!

https://adoptopenjdk.net/getinvolved.html

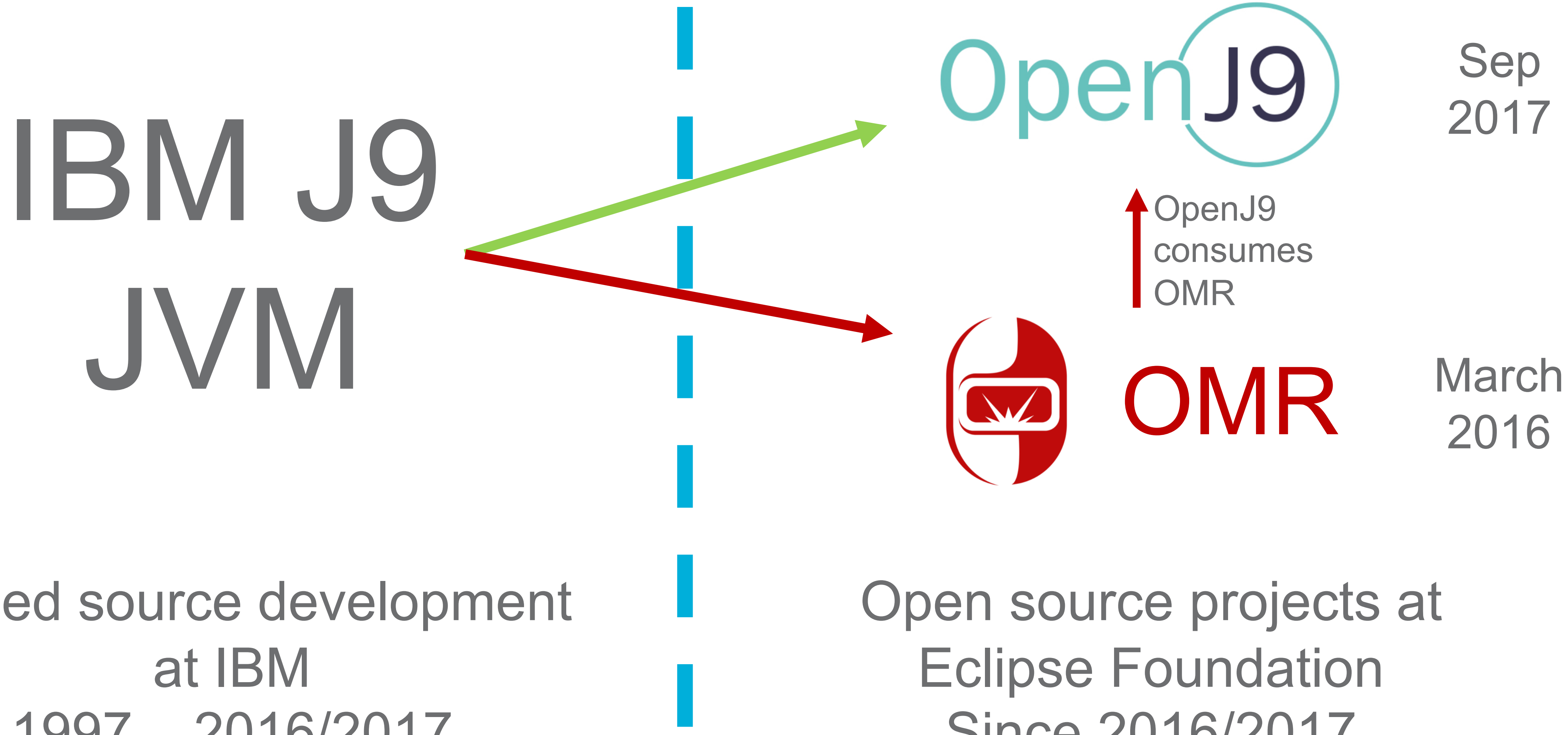
Who is participating ?
Here are some of the over 200 contributors!

packet
LJC
London Java Community
IBM
DigitalOcean
Joyent
Scaleway
CLOUDFLARE
etnetera
Linaro
MacStadium
GitHub
CLOUDCONE
Localize
slack
Installers by BITROCK
Lightbend
digicert
macincloud
Microsoft Azure

The Java Ecosystem

and OpenJ9's place in it

Eclipse OpenJ9 origin is IBM J9 JVM which produced **two** open source projects!



IBM J9 JVM

OpenJ9

Sep 2017

OpenJ9 consumes OMR

OMR

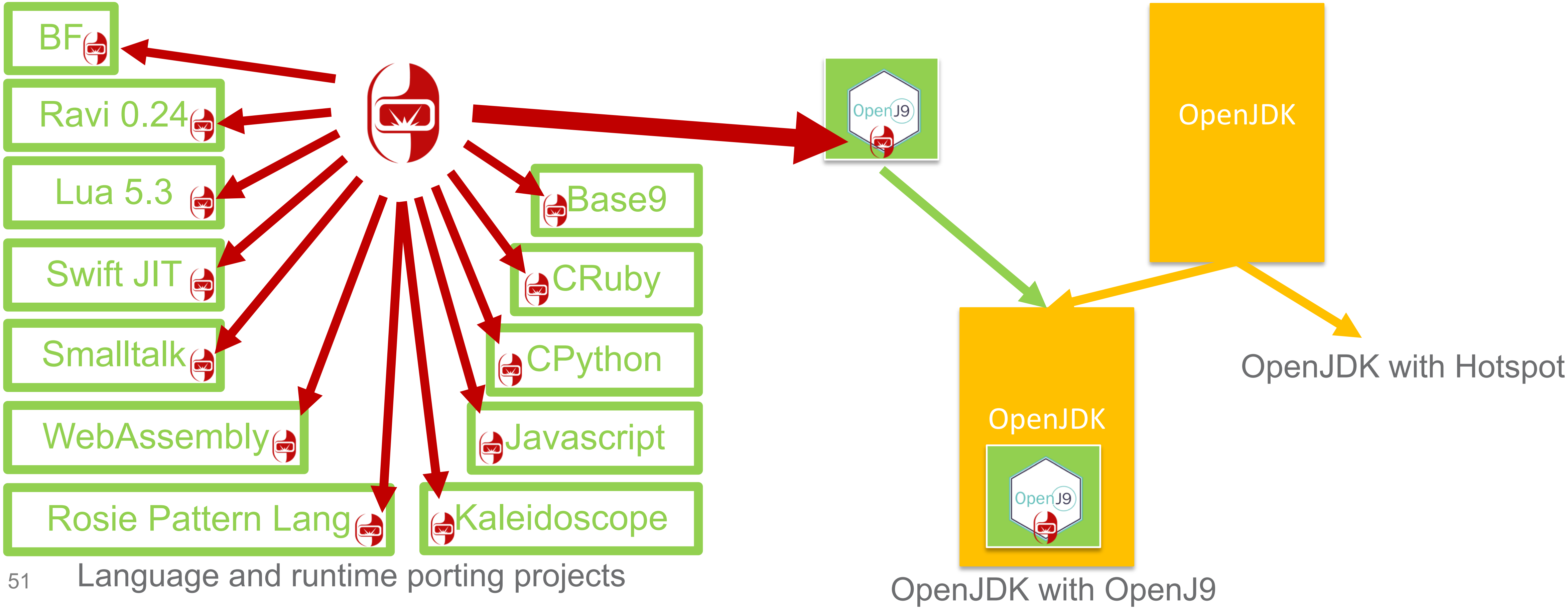
March 2016

Closed source development at IBM
1997 – 2016/2017

Open source projects at Eclipse Foundation
Since 2016/2017

Eclipse OMR : OpenJ9 : OpenJDK

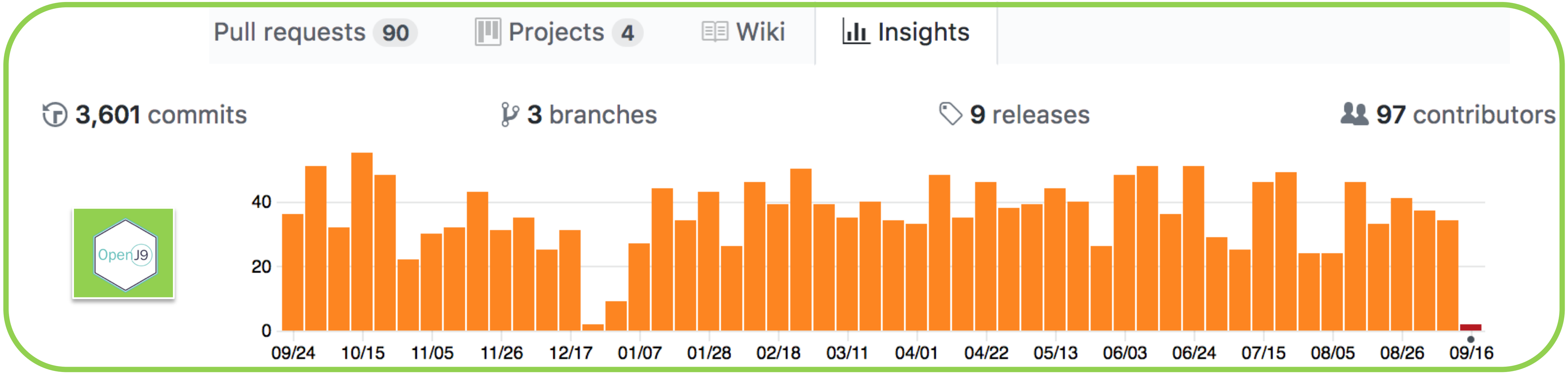
- Eclipse OMR is an open source project with reusable, reliable components (GC, JIT, etc.) for building all kinds of language runtimes



Language and runtime porting projects

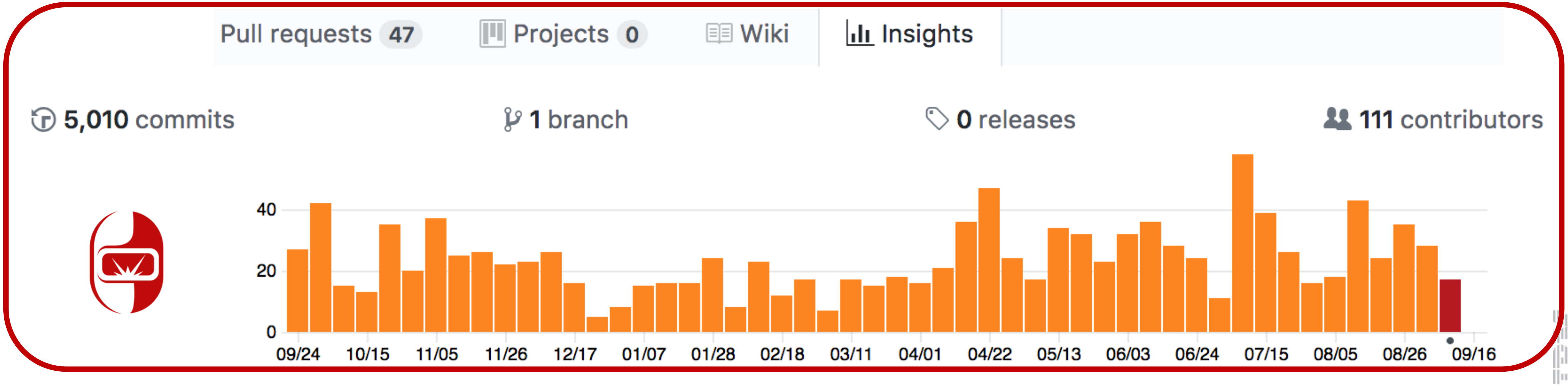
OpenJDK with OpenJ9

OpenJ9 and OMR very active projects



Pull requests 90 | Projects 4 | Wiki | Insights

3,601 commits | 3 branches | 9 releases | 97 contributors



Pull requests 47 | Projects 0 | Wiki | Insights

5,010 commits | 1 branch | 0 releases | 111 contributors

OpenJ9: 100% Open Source JVM Project but not associated with OpenJDK

- Under governance of the Eclipse Foundation
- Source code in `git` on GitHub: https://github.com/eclipse/openj9
- Issues and Features tracked at our GitHub project
- All changes are handled via Pull Requests:
  - Open CI testing performed before merge via “Jenkins” bot
  - Cross platform, cross JDK test results reported directly in PR
- Community communicates via:
  - Request slack invite: https://www.eclipse.org/openj9/oj9_joinslack.html
  - Hangout Wednesdays 11am EST: see `#planning` channel for details

What’s with the **0.**8, **0.**9, **0.**10 releases?

- Eclipse considers OpenJ9 to be in “incubator” phase:
  - *”After the project has been created, the purpose of the incubation phase is to establish a fully-functioning open-source project.”*
  - *Source: http://www.eclipse.org/projects/dev_process/development_process.php#6_2_3_Incubation*
- When we graduate from incubation, we can do our 1.0 release
- 0.x release does not reflect on quality of OpenJ9 JVM…
  - Simply that we are not yet a fully fledged Eclipse project
  - Vendor neutrality (all committers are from IBM) currently holds us back
  - **We welcome all kinds of contributors to the project**
- Proof of JVM Quality: IBM chooses an OpenJ9 nightly build with which to ship updates to the SDK for Java 8

License

- OpenJDK is GPLv2 with Classpath Exception
- Eclipse OpenJ9 is AL2 or EPLv2
  - EPLv2 has secondary license compatibility for OpenJDK's license
- OpenJDK + OpenJ9 can be used anywhere OpenJDK can be used

- OpenJ9's license is more flexible than OpenJDK's license so that OpenJ9 (and OMR) can participate in other language ecosystems

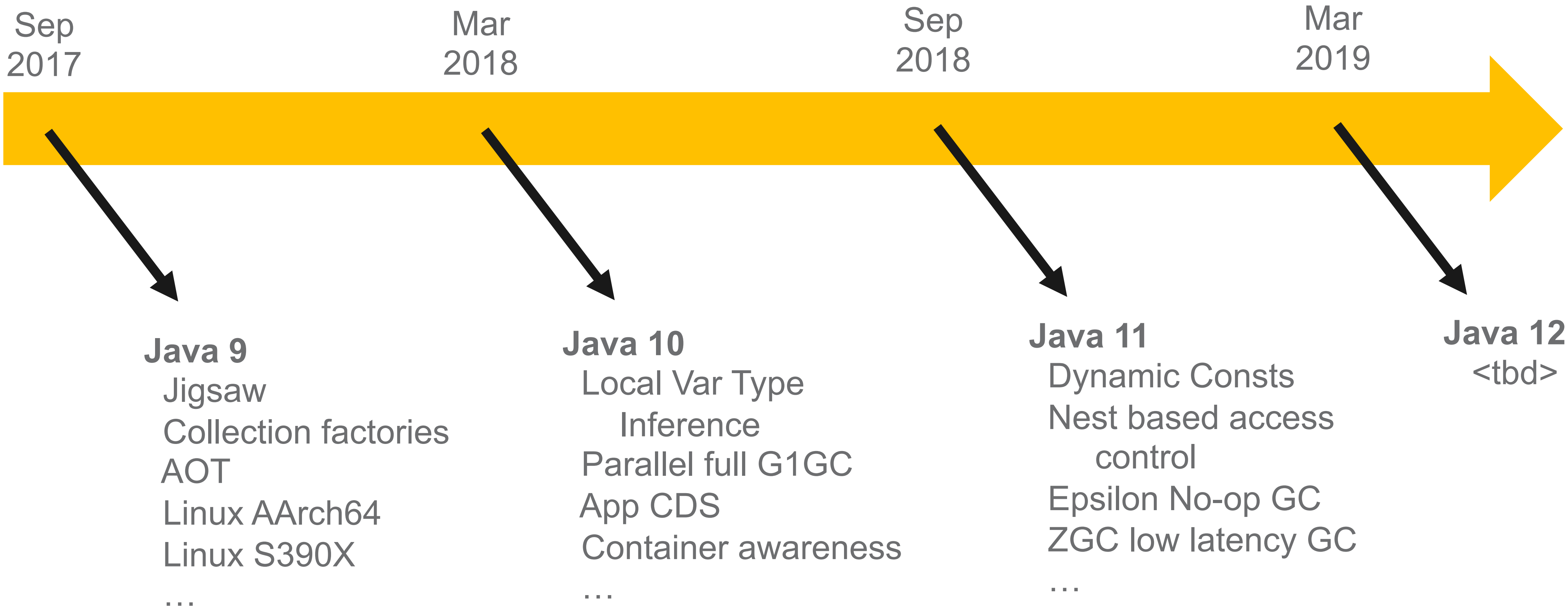
A JVM-only project has another advantage

JVM features need not be

linked to a specific JDK release

(OpenJ9 releases deliver even to JDK8!)

OpenJDK features now come every 6 months



Sep 2017

Java 9
- Jigsaw
- Collection factories
- AOT
- Linux AArch64
- Linux S390X
- …

Mar 2018

Java 10
- Local Var Type Inference
- Parallel full G1GC
- App CDS
- Container awareness
- …

Sep 2018

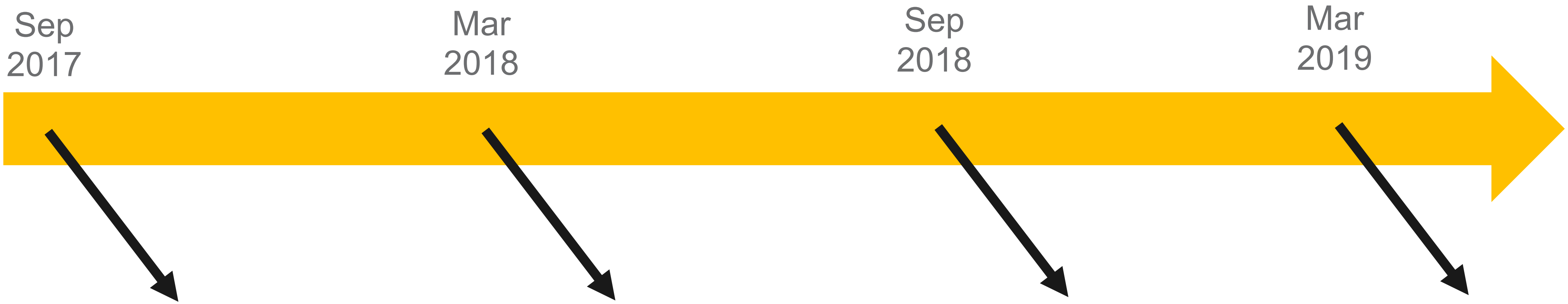
Java 11
- Dynamic Consts
- Nest based access control
- Epsilon No-op GC
- ZGC low latency GC
- …

Mar 2019

Java 12
- <tbd>

OpenJDK features now come every 6 months

Language / JCL / spec changes



Sep 2017

Mar 2018

Sep 2018

Mar 2019

Java 9

- **Jigsaw**
- **Collection factories**
- **AOT**
- Linux AArch64
- Linux S390X
- …

Java 10

- **Local Var Type Inference**
- Parallel full G1GC
- App CDS
- Container awareness
- …

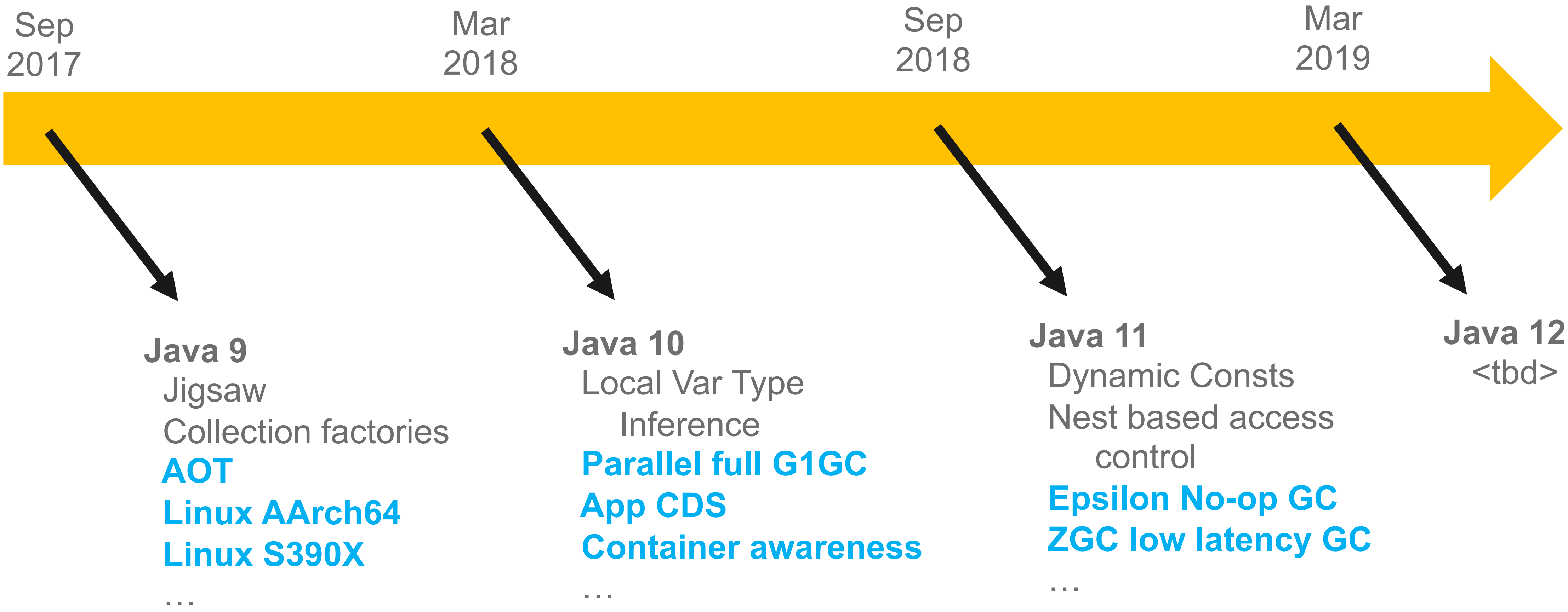
Java 11

- **Dynamic Consts**
- **Nest based access control**
- Epsilon No-op GC
- ZGC low latency GC
- …

Java 12

<tbd>

OpenJDK features now come every 6 months
JVM technology / platform changes



Sep 2017

Mar 2018

Sep 2018

Mar 2019

Java 9
- Jigsaw
- Collection factories
- **AOT**
- **Linux AArch64**
- **Linux S390X**
- …

Java 10
- Local Var Type Inference
- **Parallel full G1GC**
- **App CDS**
- **Container awareness**
- …

Java 11
- Dynamic Consts
- Nest based access control
- **Epsilon No-op GC**
- **ZGC low latency GC**
- …

Java 12
- <tbd>

OpenJ9 has frequent releases too

- Transitioning to 6 releases / year based on OpenJDK releases
  - 2 feature releases tracking new OpenJDK releases (JDK 9,10,11,12,…)
  - 4 update releases tracking OpenJDK quarterly update releases

- **BUT**: each OpenJ9 release can be built into **any** supported JDK
  - OpenJ9 0.8 (Mar 2018) builds into JDK8, JDK9
  - OpenJ9 0.9 (Aug 2018) builds into JDK8, JDK10
  - OpenJ9 0.10 (Sep 2018) feature release to build in JDK8, JDK10, JDK11
  - OpenJ9 0.11 (Oct 2018) will build into JDK8, JDK11

Red JDK = Long Term Support Release

Release		OpenJ9 Feature	JDK8	JDK9	JDK10	JDK11
0.8.0	Mar 2018	JDK8 certified by AdoptOpenJDK	🟩 ✓			
		JDK9 feature complete (since Sep 2017)		🟩 ✓		
		Linux 64-bit on X86, ppc64le, s390x, Windows 64-bit platform support	✓	✓		
0.9.0	Aug 2018	JDK10 feature complete			🟩 ✓	
		Windows 32-bit builds	✓		✓	
		Large heap builds for Linux x86-64	✓		✓	
		New GC policy “no-gc”	✓		✓	
		Idle tuning features	✓		✓	
		Container awareness	✓		✓	
0.10.0	Sep 2018 *	JDK11 feature complete				🟩 ✓
		Improved JNI performance	✓		✓	✓
		Increase default shared cache size	✓		✓	✓
		Improve option support for Hotspot migration	✓		✓	✓

Green box = Java language/spec work

Eclipse OpenJ9

is the only 100% open source JVM

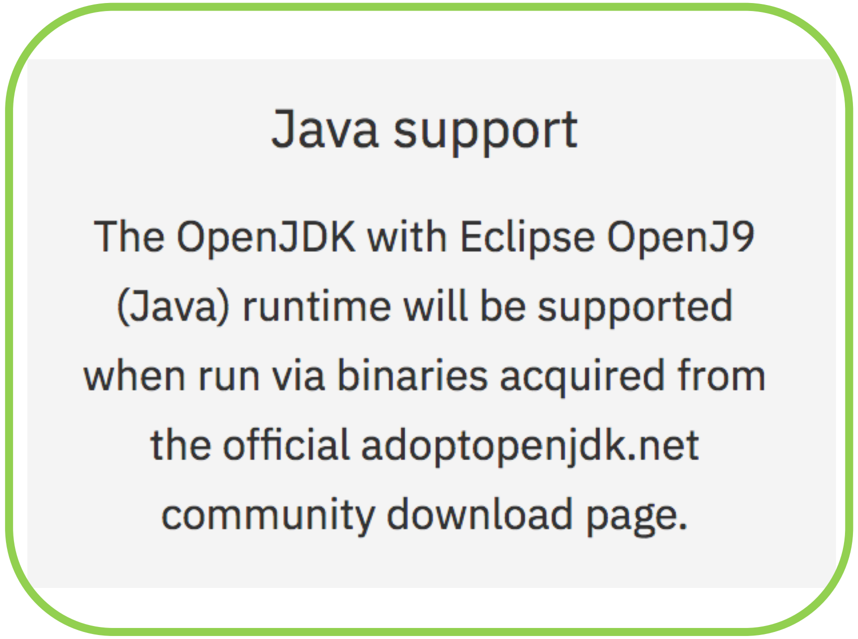
that **re-invests in the JVM platform**

independently of JDK releases

A quick word about commercial support

IBM offers commercial support for

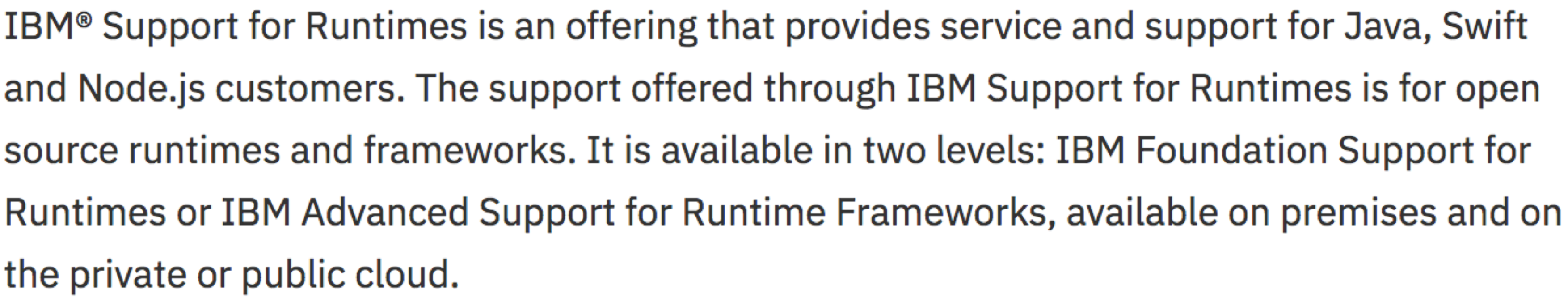
OpenJDK with OpenJ9 if you want it



Java support

The OpenJDK with Eclipse OpenJ9 (Java) runtime will be supported when run via binaries acquired from the official adoptopenjdk.net community download page.

What does IBM Support for Runtimes do?



IBM® Support for Runtimes is an offering that provides service and support for Java, Swift and Node.js customers. The support offered through IBM Support for Runtimes is for open source runtimes and frameworks. It is available in two levels: IBM Foundation Support for Runtimes or IBM Advanced Support for Runtime Frameworks, available on premises and on the private or public cloud.

https://www.ibm.com/ca-en/marketplace/support-for-runtimes

I believe that

1. OpenJDK with OpenJ9 is the best OpenJDK solution for your Java workloads
2. AdoptOpenJDK is the best place to get your OpenJDK (with OpenJ9!) binaries
3. Eclipse OpenJ9 is the best open source JVM project in the open Java ecosystem

- **Try it out! And you can start to believe it too!**

Try out OpenJDK with OpenJ9!

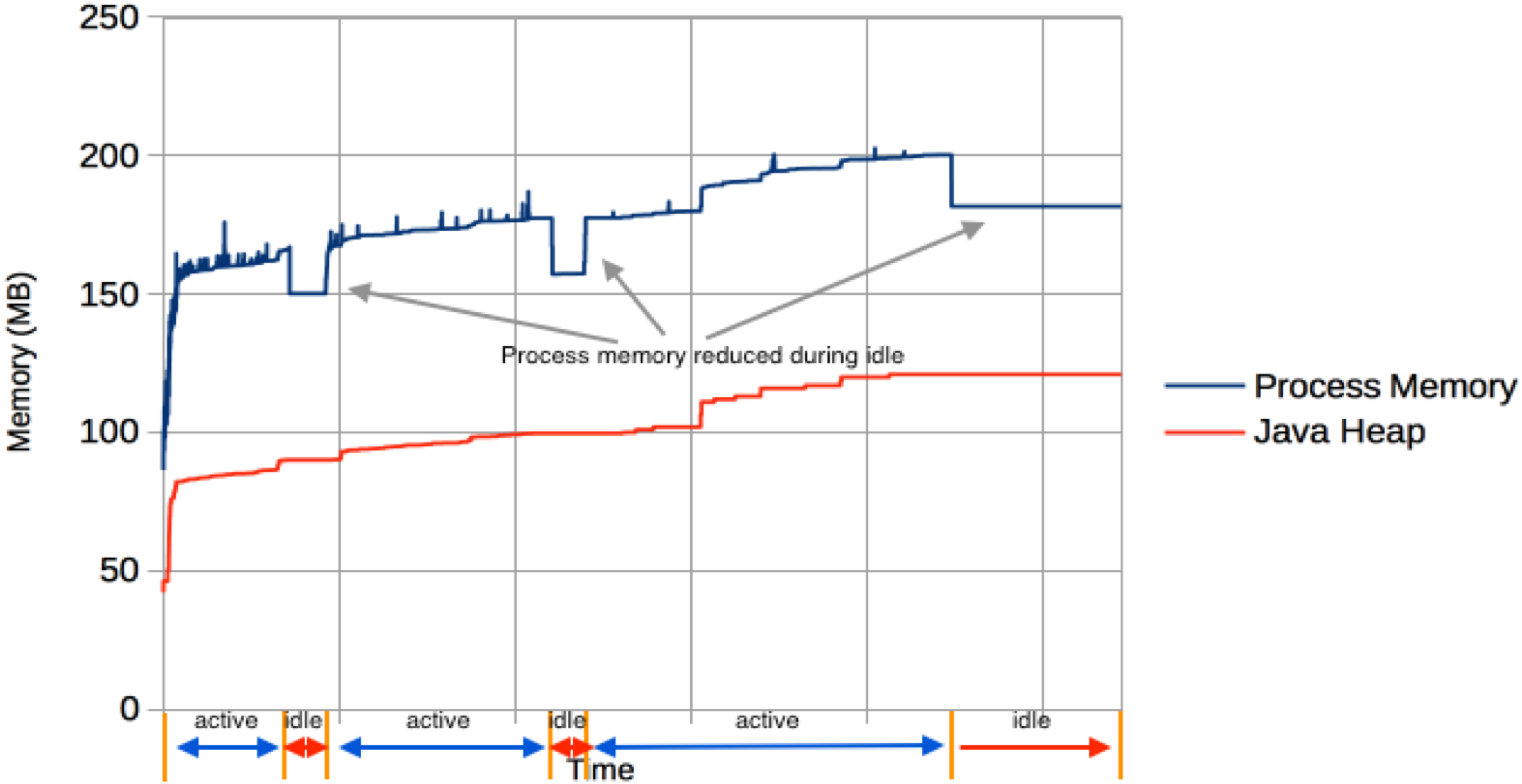
- Easy to get and integrate into your deployments
  - Download high quality, certified builds from AdoptOpenJDK!
- ~30% faster server start-up
- ~50% less physical memory use
- “Designed for Cloud” configuration options:
  - Idle mode tuning for your less active JVMs
  - Faster ramp-up in CPU constrained environments
- New JVM features (e.g. container support) with JDK8 and up!

Handy Links

▪ Eclipse OpenJ9	https://www.eclipse.org/openj9
– Source code (OpenJ9)	https://github.com/eclipse/openj9
– Source code (OMR)	https://github.com/eclipse/omr
– Slack invite	https://www.eclipse.org/openj9/oj9_joinslack.html
▪ AdoptOpenJDK	https://adoptopenjdk.net/
– Slack invite	https://adoptopenjdk.net/slack.html
– Source code	https://github.com/adoptopenjdk
▪ My contact info:	
– Mark Stoodley	mstoodle@ca.ibm.com, @mstoodle

Backup

Cloud configuration (`-XX:+IdleTuningGcOnIdle`) Enterprise workload: OpenLiberty with AcmeAir



Benchmark: https://github.com/blueperf/acmeair

More details: https://developer.ibm.com/javasdk/2017/09/25/still-paying-unused-memory-java-app-idle

Designed for the cloud: CPU and wakeups of Idle JVMs

- Analyze behavior of idle OpenLiberty server with powertop tool

OpenJDK9 with HotSpot – 0.168% CPU

```
Summary: 84.7 wakeups/second,  0.0 GPU
ops/seconds, 0.0 VFS ops/sec and 0.3% CPU use.
Usage        Events/s    Category       Description
0.9 ms/s       44.2         Process      /sdks/OpenJDK9-
x64_Linux_20172509/jdk-9+181/bin/java
119.5 µs/s      20.0       Process        [xfsaild/dm-1]
138.6 µs/s       7.4        Timer           tick_sched_timer
10.5 µs/s        1.6        Process        [rcu_sched]
190.4 µs/s       1.5        Timer           hrtimer_wakeup
```

OpenJDK9 with OpenJ9 – 0.111% CPU

```
Summary: 38.5 wakeups/second,  0.1 GPU
ops/seconds, 0.0 VFS ops/sec and 0.2% CPU use
Usage        Events/s    Category       Description
681.2 µs/s     19.2        Process       /sdks/OpenJDK9-
OPENJ9_x64_Linux_20172509/jdk-9+181/bin/java
58.3 µs/s        5.2        Timer          tick_sched_timer
21.9 µs/s        3.6        Process       [rcu_sched]
39.3 µs/s        2.0        Timer          hrtimer_wakeup
157.1 µs/s       1.0        kWork          ixgbe_service_task
```

- OpenJ9 triggers ~55% fewer wakeups than HotSpot