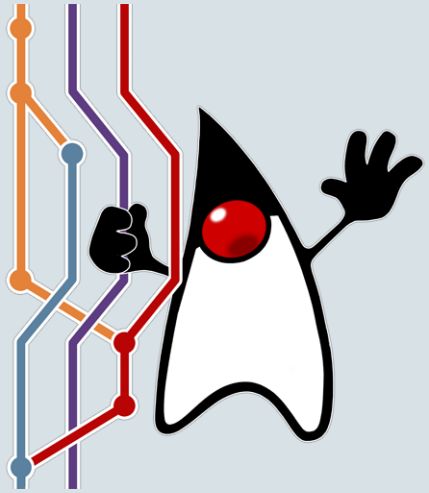
Update on Project Skara and Git

Source code management options for the JDK



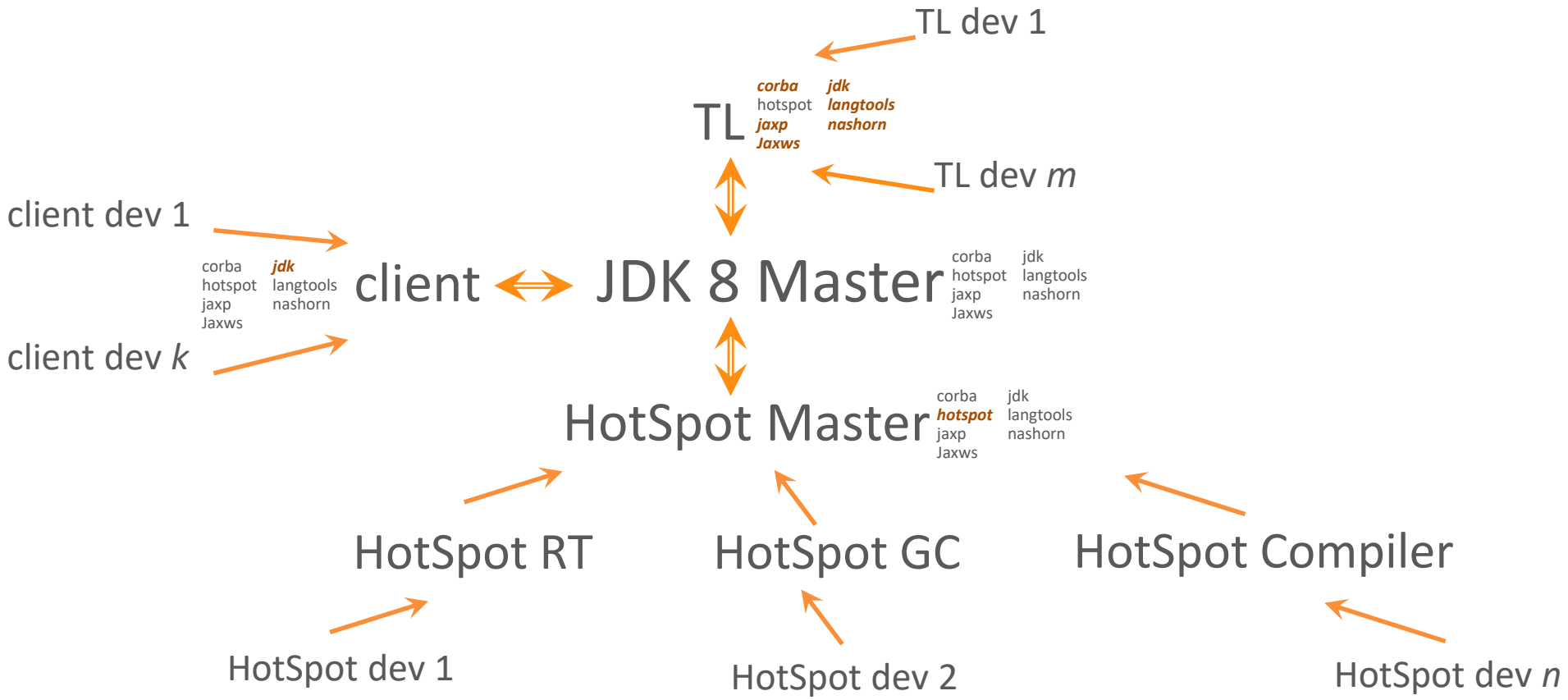
Joseph D. Darcy (darcy, @jddarcy)
Joint work with Erik Duveblad (ehelin) and Robin Westberg (rwestberg), among others
Java Platform Group, Oracle
OpenJDK Committers' Workshop
August 2019, Santa Clara, CA

Image from https://www.conservation.ca.gov/cgs/PublishingImages/Minerals/asbestos2b.jpg

Speed run of changes in JDK SCM usage, JDK 8 to present

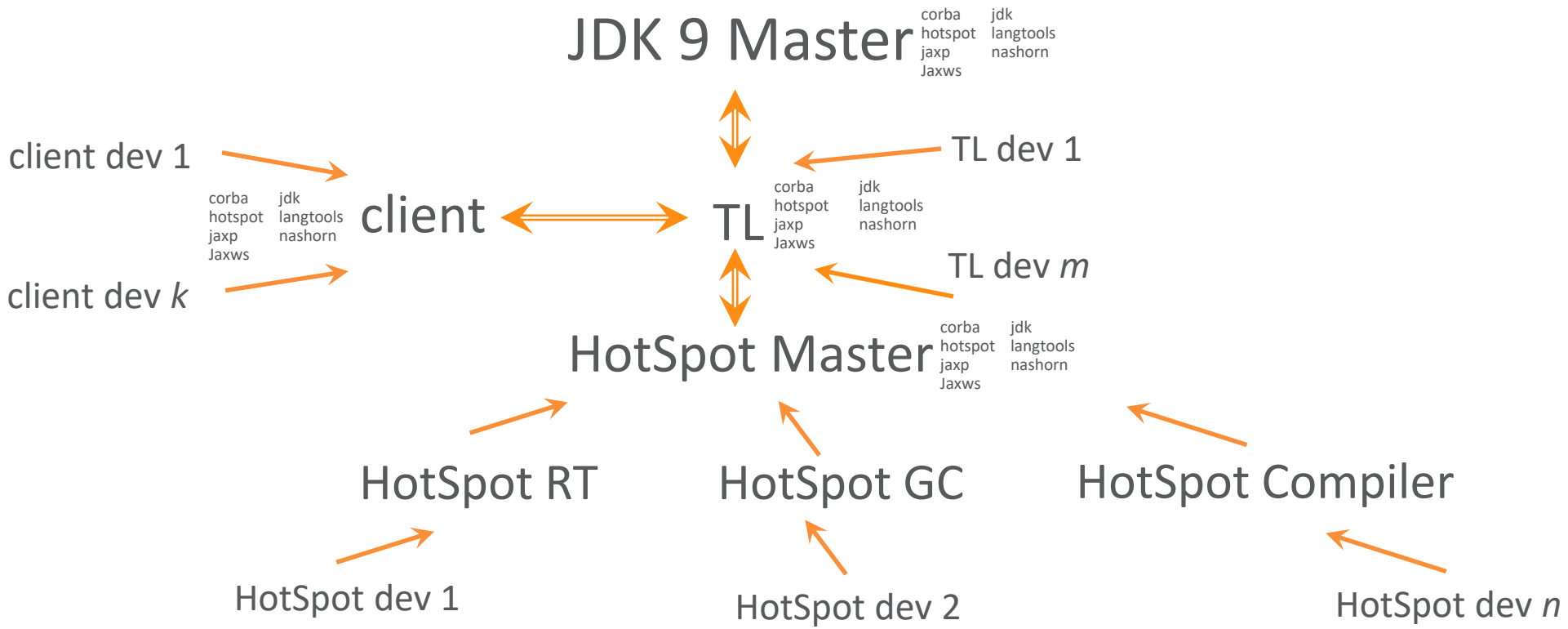
Integration topology of lines of development, JDK 8 (GA 2014)

A *graph* of integration hg *forests*



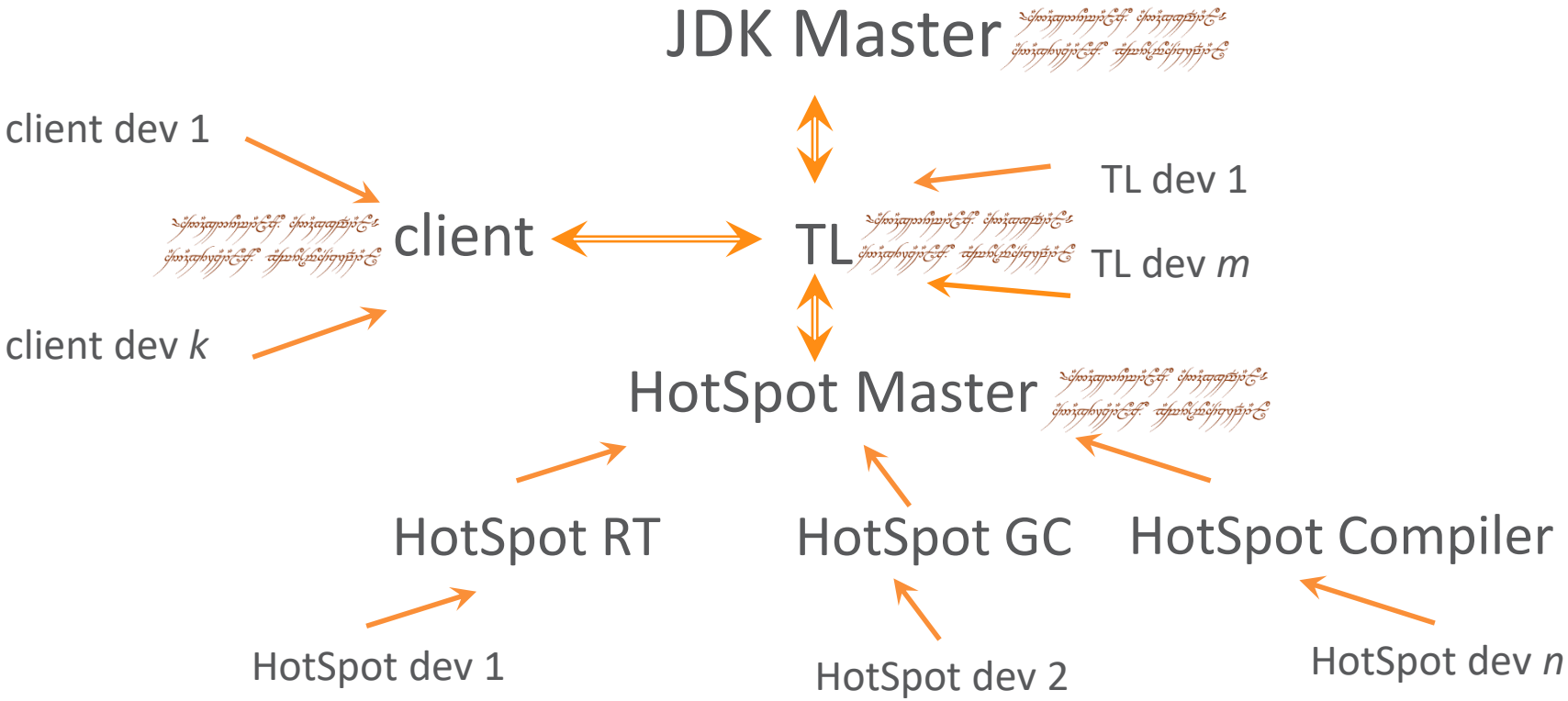
Integration topology of lines of development, JDK 9 (GA 2017)

A *tree* of integration hg forests



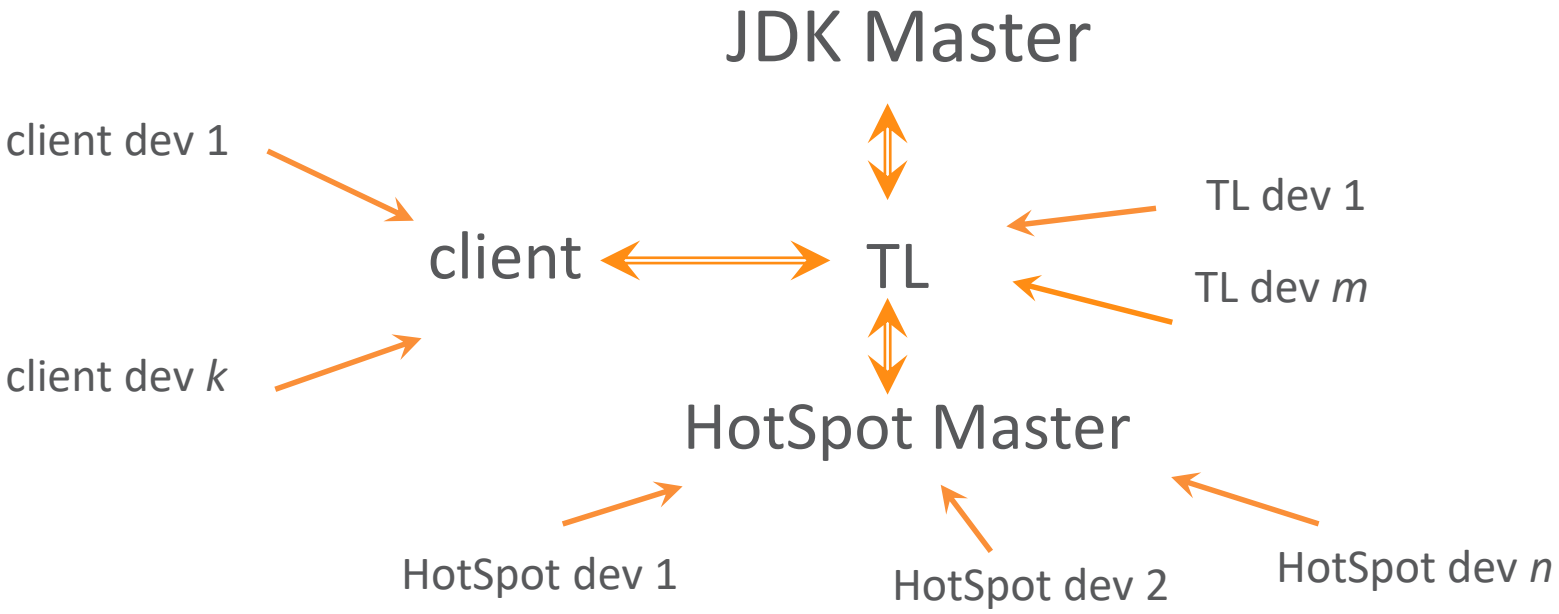
Integration topology of lines of development, JDK 10 (GA 2018)

Repo consolidation (JEP 296); a tree of integration *repos*



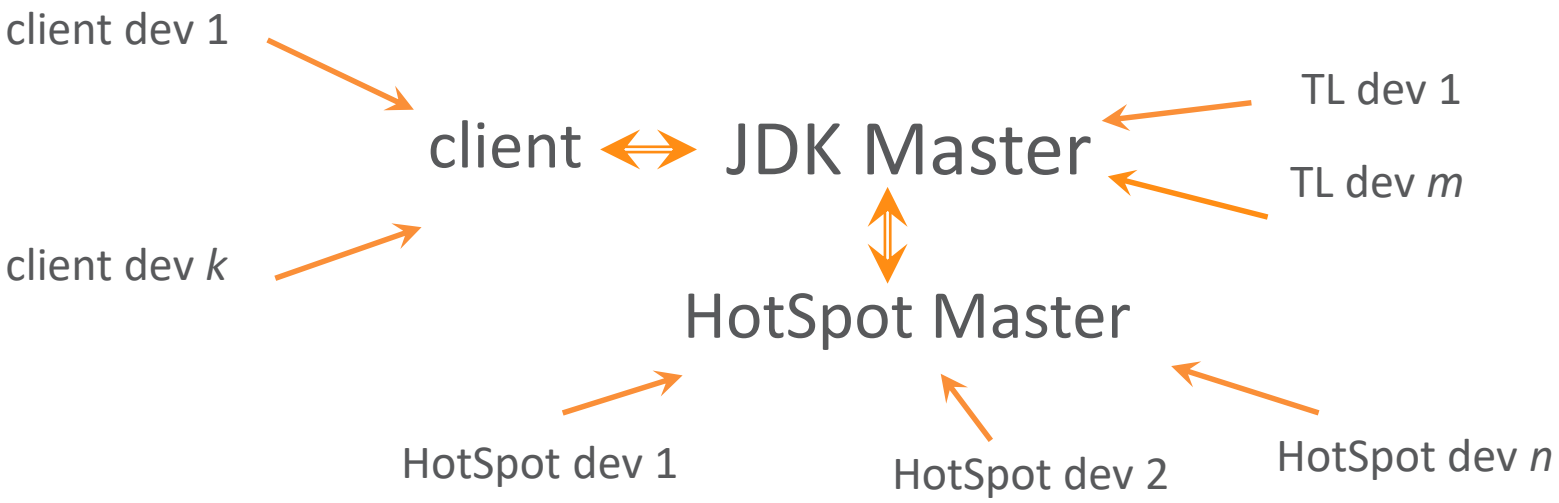
Integration topology of lines of development, JDK 10, 11, 12

Combining HotSpot lines of development



Integration topology of lines of development, JDK 10, 11, 12

Combining master and TL



Integration topology of lines of development, JDK 12 (GA 2019)

Combining HotSpot Master and master

Why bother?

- Better SCM architecture: single consolidated repo (mostly) shared by all
- Simpler and easier to make atomic commits which cross javac, core libs, hotspot boundaries
  - Making such a change in JDK 8 would require multiple pushes and take month or more to propagate cross the graph of integration forests
  - In JDK 12 and later, such a change can be pushed as a single update to master.[†]
    [†]Still needs to propagate to client repo.
  - Recent example: start of JDK 14 changeset updated all of the build system, javac, core libs, and hotspot
  - Enabled by improvements to testing and other development practices
- Allows (but does not require) shorter release cycles

What’s next?

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Project Skara, CFD July 2018

Call for discussion: investigating source code management options for the JDK sources

“In order to help OpenJDK contributors be more productive, both seasoned committers and relative newcomers, the Skara project proposes to investigate alternative SCM and code review options for the JDK source code, including options based upon Git rather than Mercurial, and including options hosted by third parties.

The Skara project intends to build prototypes of hosting the JDK 12 sources under different providers.

The evaluation criteria to consider include but are not limited to:

* Performance: time for clone operations from master repos, time of local operations, etc.
* Space efficiency
* Usability in different geographies
* Support for common development environments such as Linux, Mac, and Windows
* Able to easily host the entire history of the JDK and the projected growth of its history over the next decade
* Support for general JDK code review practices
* Programmatic APIs to enable process assistance and automation of review and processes

If one or more prototypes indicate a different SCM arrangement offers substantial improvements over the current situation, the Skara project will shepherd a JEP to change the SCM for the JDK.”

Along the way, many mirrors

https://github.com/openjdk/

- Git mirrors of hg repos of consolidated JDK lines of development (and a few other repos from codetools) including:
  - jdk/jdk
  - jdk13
  - jfx
  - amber
  - loom
  - panama
  - shenandoah
  - tsan
  - …

Added by April 2019

Imported repos reformatted commit messages

- Current:

8225035: Thread stack size issue caused by large TLS size
Summary: Adjust thread stack size for static TLS on Linux when AdjustStackSizeForTLS is enabled.
Reviewed-by: dholmes, fweimer, stuefe, rriggs, martin
Contributed-by: jeremymanson@google.com, fweimer@redhat.com, jianglizhou@google.com

- Reformatting works better with git tooling including `git log`
- Multiple authors supported
- Git has separate notions of author and committer; could be used for backports too

- Proposed

8225035: Thread stack size issue caused by large TLS size

Adjust thread stack size for static TLS on Linux when AdjustStackSizeForTLS is enabled.

Co-authored-by: Florian Weimer fweimer@redhat.com
Co-authored-by: Jiangli Zhou jianglizhou@google.com
Reviewed-by: dholmes, fweimer, stuefe, rriggs, martin

“The Skara tooling is now open source”, June 2019

https://github.com/openjdk/skara

“The Skara tooling includes both server-side tools (so called "bots") as well as several command-line tools:

- git-jcheck
- git-webrev
- git-defpath
- git-fork
- git-pr
- git-translate
- git-info
- git-skara”

New candidate JEP 357, July 2019

Migrate from Mercurial to Git

Summary

Migrate the OpenJDK Community's source code repositories from Mercurial (hg) to Git.

Goals

- Migrate all single-repository OpenJDK Projects from Mercurial to Git
- Preserve all version control history, including tags
- Reformat commit messages according to Git best practices
- Port the jcheck, webrev, and defpath tools to Git
- Create a tool to translate between Mercurial and Git hashes

Audience straw poll

Your three options

- Indifferent to which distributed SCM is used
- Stay on Mercurial (hg)
- Move to Git

I think we should use Git as the SCM for the JDK.

Why?

Reasons to migrate to Git include

- Nuts and bolts (repo on disk size, etc.)
- Availability of hosting providers
  - More contemporary review workflow
  - Automation with bots, OCA, jcheck, …
  - Fast download times, geo-caching
- Git's current scaling is sufficient for many, many more years of JDK development at current change rate.
- Much more prevalent usage
  - From 2018 stackoverflow, ≈90% for Git vs. ≈4% for hg)
  - From Snyk 2018 JVM Ecosystem report: 74% for git, 16% subv., 7% other, 3% none

Why not?

Reasons to stay on Mercurial

- Familiarity; accustomed and inured to existing limitations & workarounds
- Avoid migration costs
  - Retraining
  - Update existing boundary systems (build pipelines, CI, …)
  - …

Partial cheat sheet of equivalents of hg commands in git

See https://github.com/sympy/sympy/wiki/Git-hg-rosetta-stone

Hg	Git
hg clone *HgURL*	git clone *GitURL*
hg help *command*	git help *command*
hg status	git status
hg add	git add
hg rm	git rm
hg pull -u	git pull
hg export	git format-patch
hg push	git push
~/.hgrc	~/.gitconfig

Skara team can host “Git for JDK Developers” talk, etc.

Looking ahead, a 3rd party hosting provider?

Projects using Github.com

- Apache: https://github.blog/2019-04-29-apache-joins-github-community/
- Adopt OpenJDK: https://adoptopenjdk.net/
- Intellij IDEA: https://github.com/JetBrains/intellij-community
- Graal: https://github.com/graalvm
- OpenJ9: https://github.com/eclipse/openj9
- …

Skara development model

Being used for Skara itself!

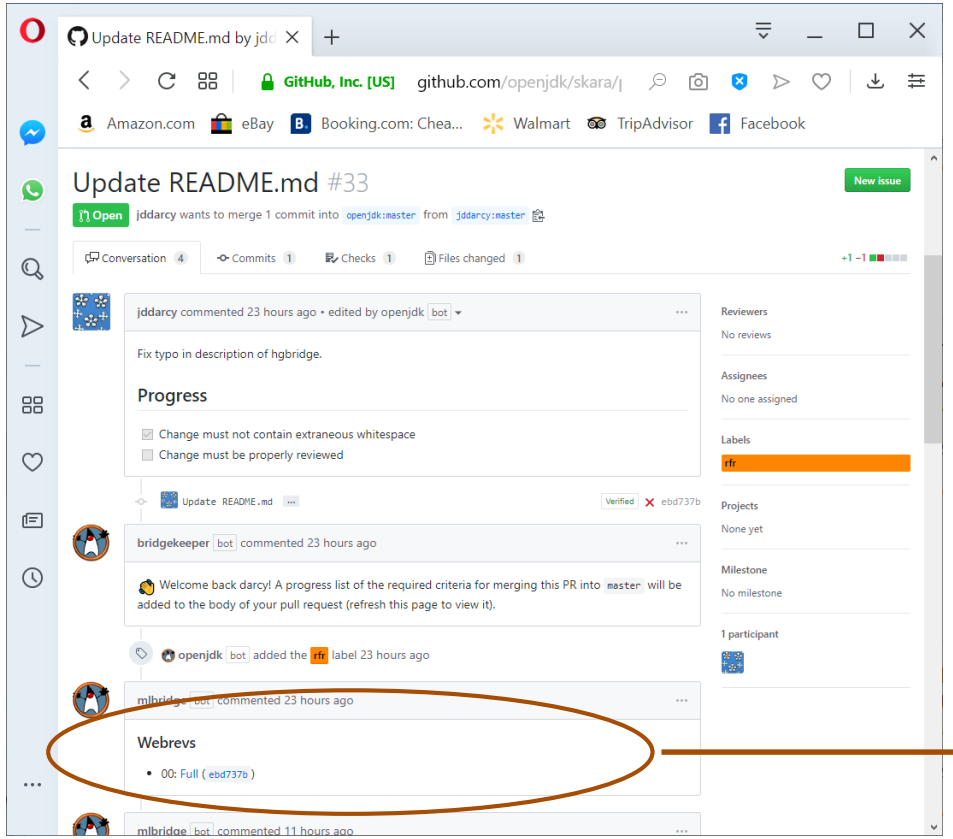
- Enable use of github/gitlab workflow while supporting historical JDK development practices (webrevs and mailing lists).

Model for using a hosting provider…

- A developer has one more clones on the hosting provider *server*; self-service, done quickly
- These can then be cloned locally and worked on
- Pushes are made to the personal clones
- A *pull request* (PR) is made from the personal clone to the master
  - Bots and people review the request
  - Bots then handle the actual push
  - Users would *not* push directly to master

Example, starting with a PR

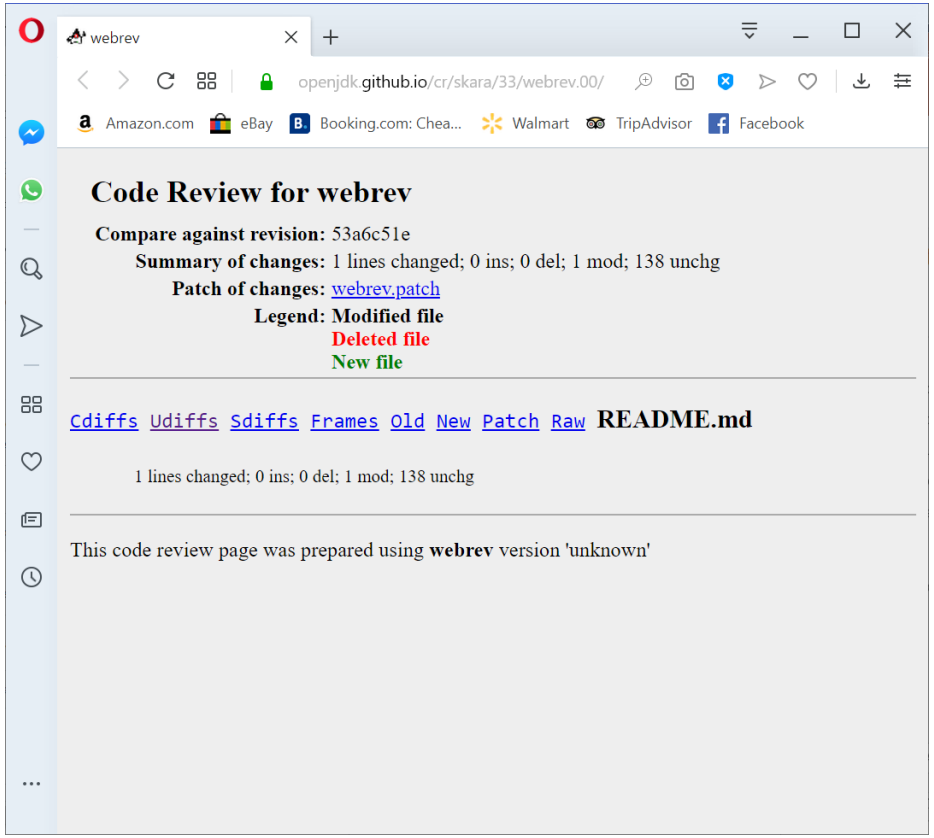
https://github.com/openjdk/skara/pull/33



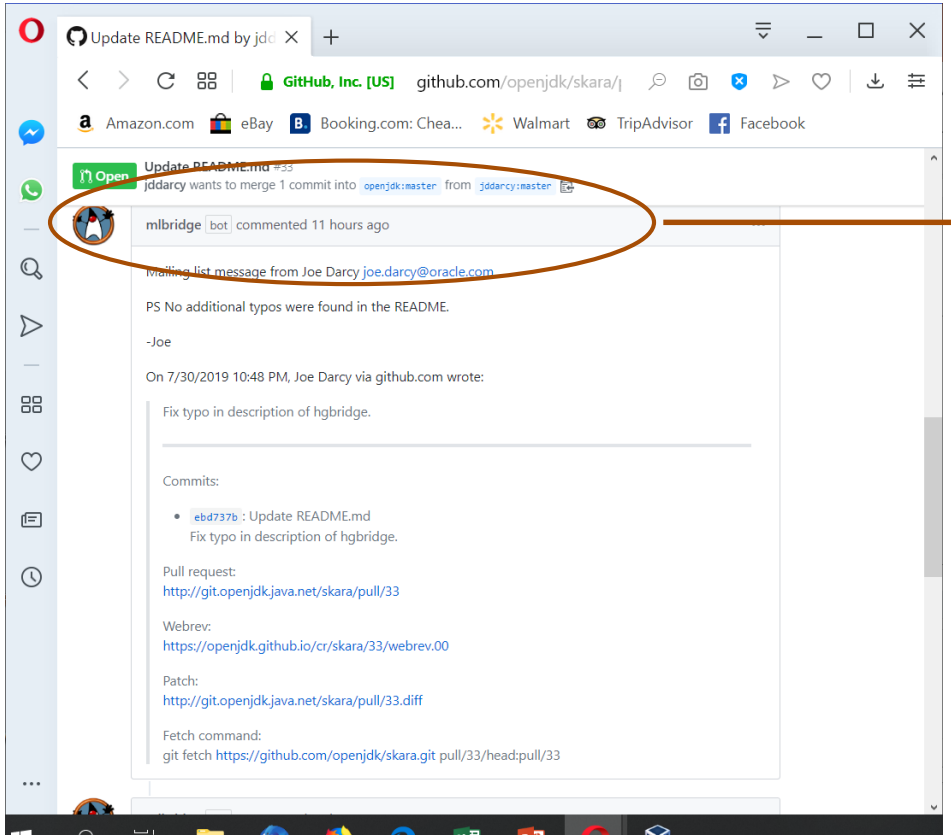
Bot generates and uploads a corresponding webrev.

The webrev

https://openjdk.github.io/cr/skara/33/webrev.00/



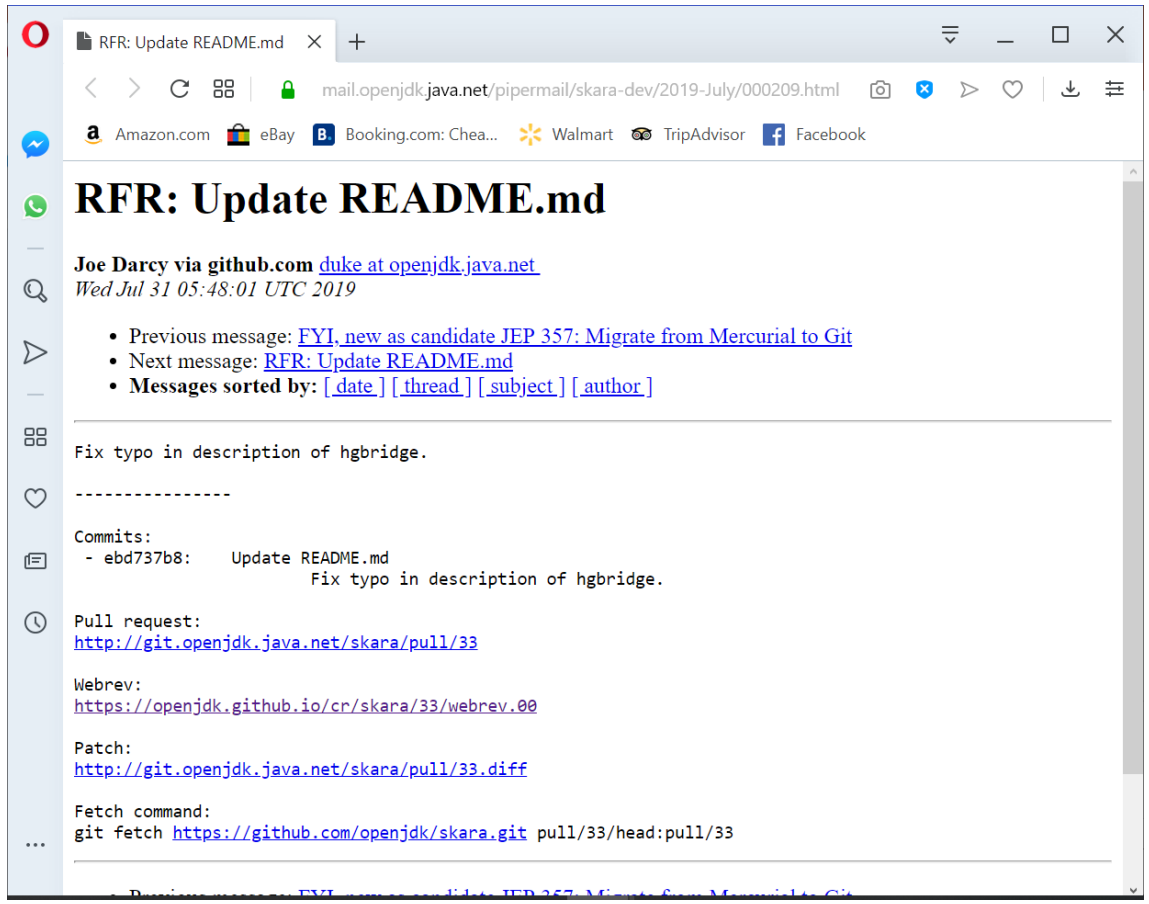
More bot tasks



Bot starts a thread on the mailing list.

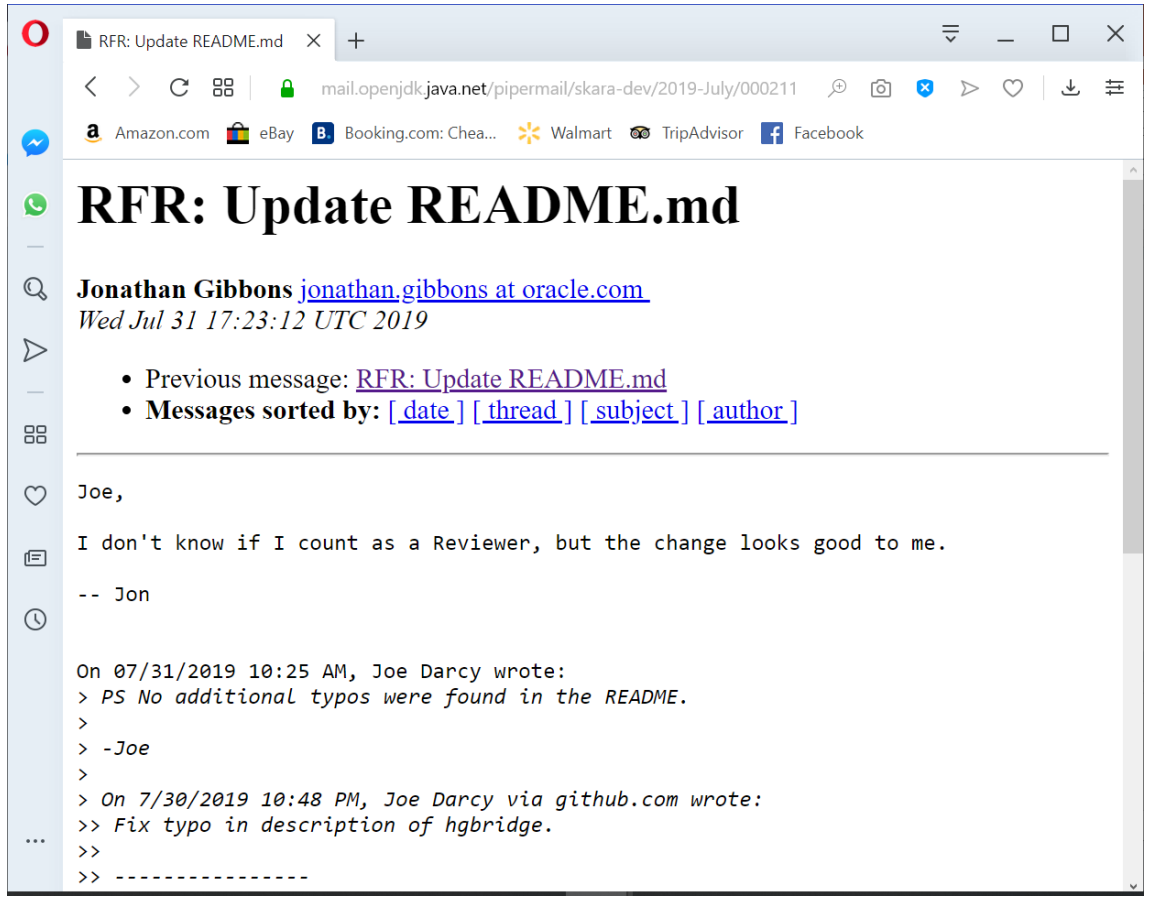
Mailing list thread

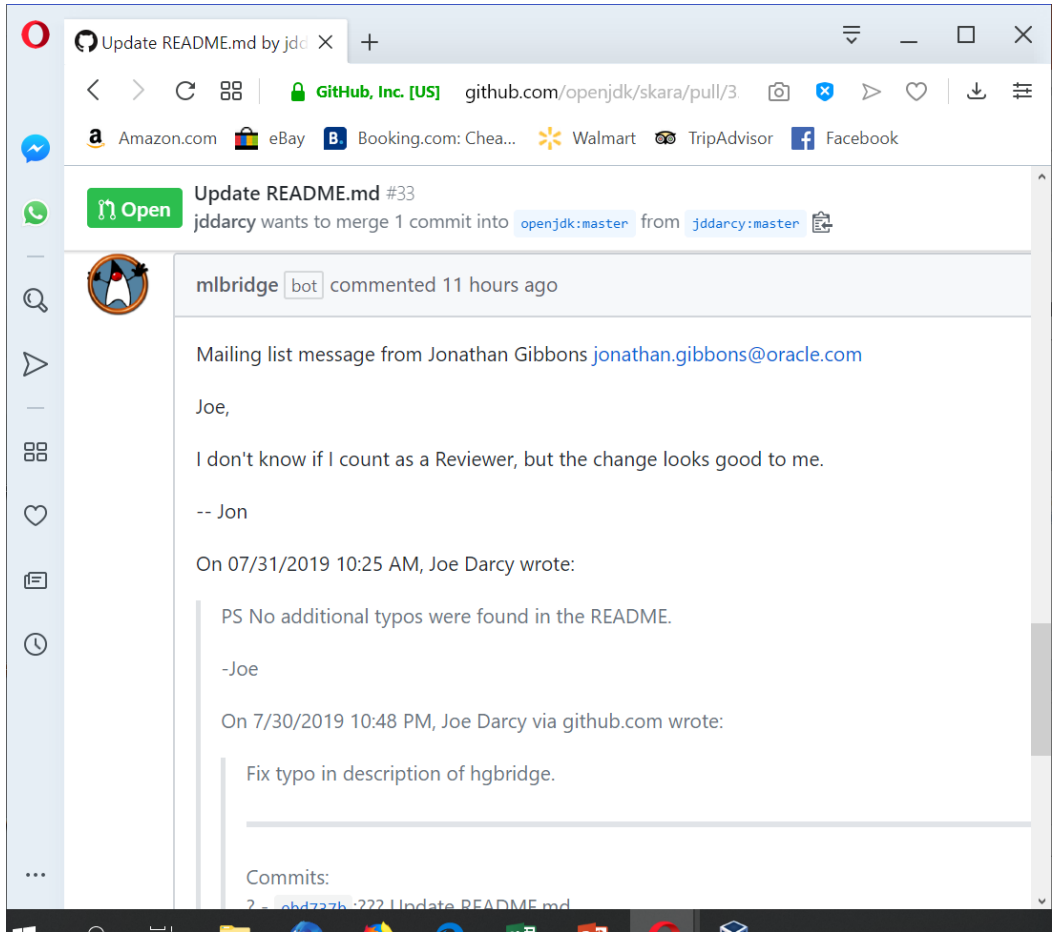
https://mail.openjdk.java.net/pipermail/skara-dev/2019-July/000209.html



- Includes links back to
  - Pull request
  - Webrev & patch

Mailing list comments reflected back in the pull request





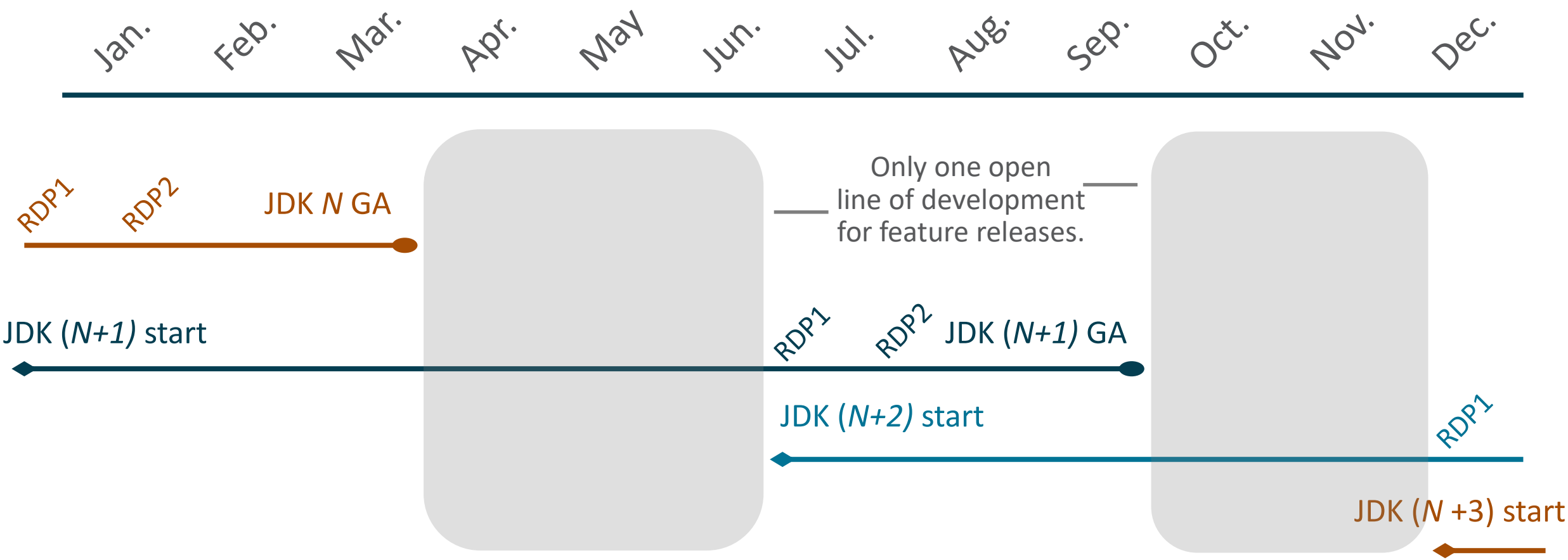
Invitation to use the git mirrors for downstream projects

- Good to have more users and gain experience with the model
- If interested, contact Skara team for scheduling, etc.

Transition Considerations

Schedule of JDK features releases under six-month schedule

Note: figure not drawn exactly to scale.



Feature release considerations on timing a switch

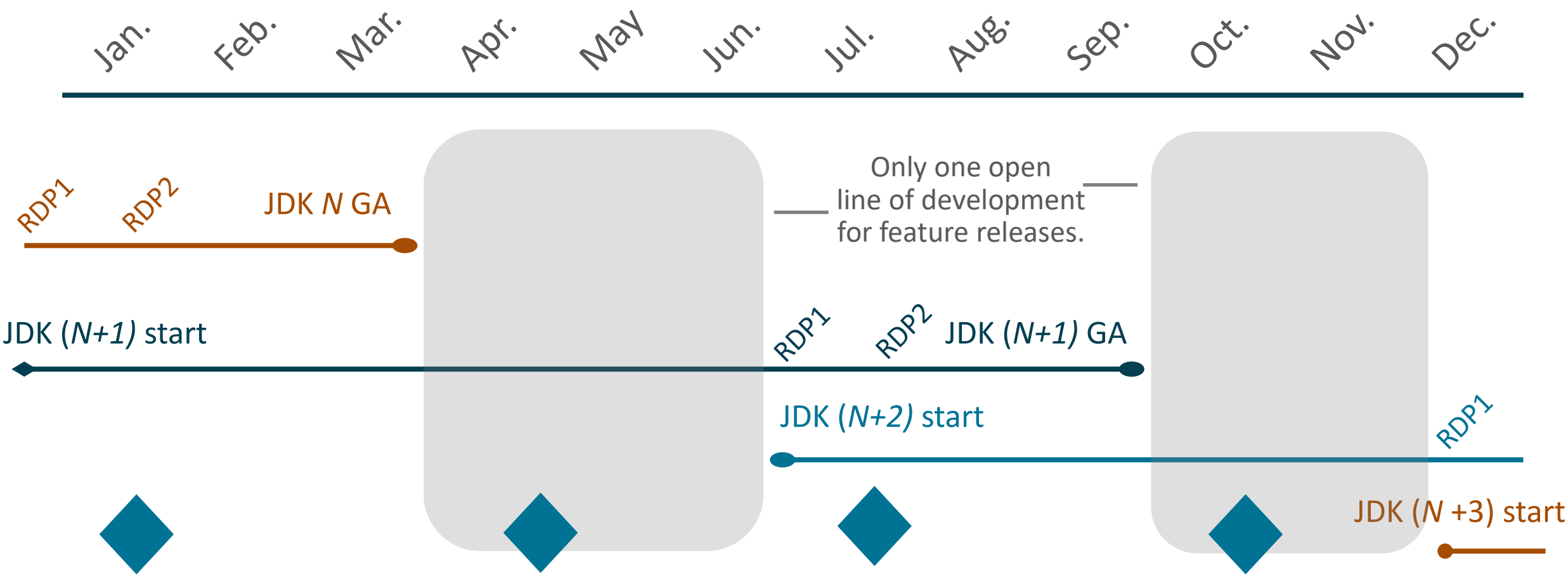
- With the six month release model, each JDK feature release is worked on for about 9 months, either:
  - Mid-December $YEAR to mid-September following year
  - Mid-June $YEAR to mid-March following year
- Implies *two* lines of development open
  - mid-December through mid-March
  - mid-June through mid-September
- And only one line of development open at other times starting after a GA:
  - mid-March through mid-June
  - mid-September through mid-December

Schedule for quarterly security update releases (CPUs)

- Mid-January
- Mid-April
- Mid-July
- Mid-October

Schedule overlay

Note: figure not drawn exactly to scale.



Recommended switching windows

- Assuming all eligible releases transition at the same time, two switching windows:
  - Mid-March through mid-June preferring mid-April to mid-June
  - Mid-September through mid-December, preferring mid-October through mid-Dec.

What about the update releases?

- If JDK *N* is moved to git, JDK *N*.0.*k*, $k \geq 1$ should be on git too.
- Consolidated releases are candidates to go to git; i.e. 11u, but *not* 8u in its present configuration
- Patches could be extracted from a JDK *N*, $N \geq 13$ git repo, (`git format-patch`) and manipulated with the same kinds of patch manipulations scripts used to move patches across consolidation and modularization boundaries.

Boundary systems

- Include, but are not limited to…
  - Bug system
  - Code review
  - CI systems
  - Submit repo
  - Personal scripts
- General approach: run new system in parallel with mocked/alternative boundary systems

Next steps

- More hg $\Rightarrow$ git transition tips from Skara team
- (Hopefully) feedback from early adopters on Skara tooling for JDK-related projects
- Second JEP on hosting options

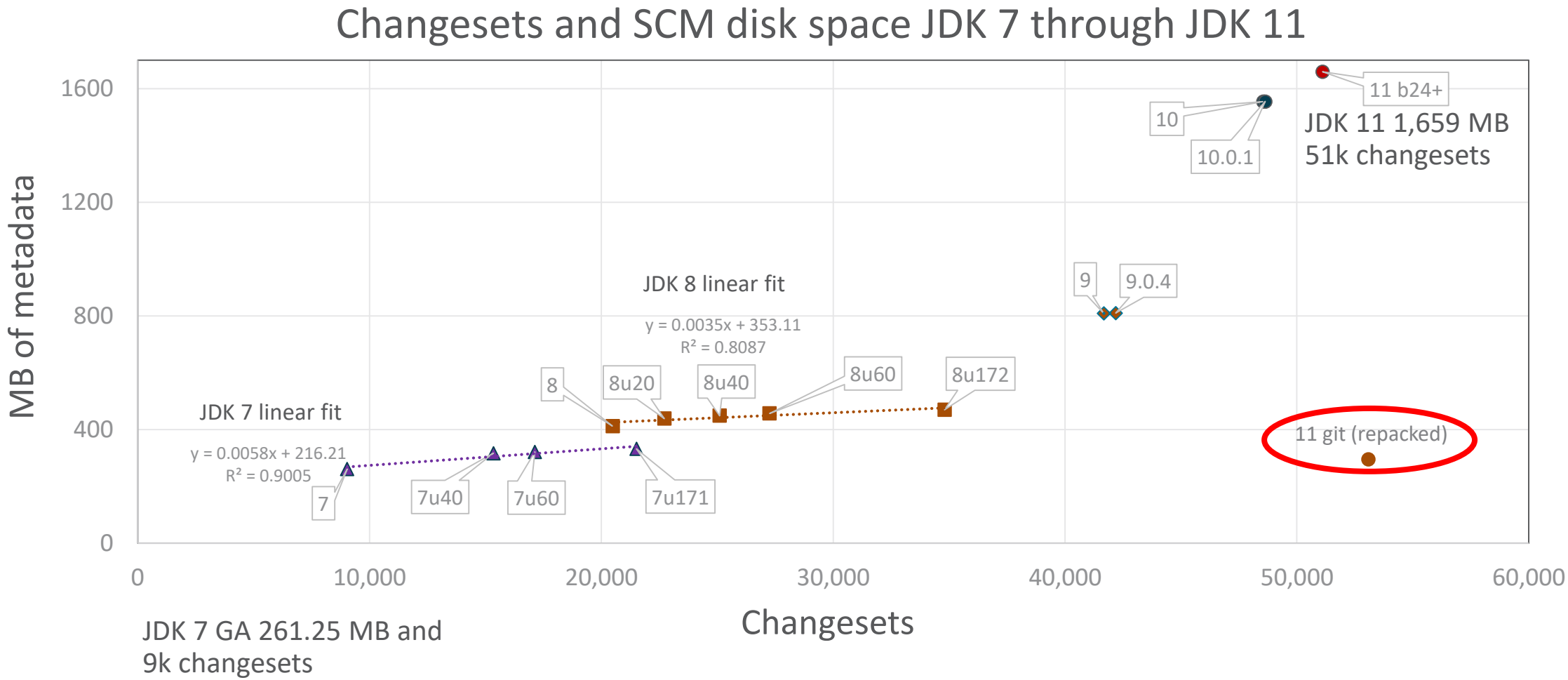
Discussion

Follow-ups to: skara-dev@openjdk.java.net

http://cr.openjdk.java.net/~darcy/Presentations/OCW/ocw-2019-08-skara.pdf

Backup slides

Changesets and repo metadata size;
E.g.: sum of `du -k --total `find . -name ".hg" -type d`



Summary of changes in JDK 10, 11, and 12

- Repo consolidation in JDK 10 (JEP 296)
- Bulk syncs from JDK (*N* – 1) into JDK *N* starting with *N* = 10
- Disabled duplicate bug id check in jcheck to ease bulk syncing into JDK 10
- Combined dev and master into a single entity “master” in JDK 10, now jdk/jdk; integrations are just tags
- Multiple HotSpot lines of development combined into one
- HotSpot engineers push directly into master, i.e. no dedicated HotSpot line of development (end of JDK 10, JDK 11 so far)