JGit under EDL at Eclipse.org

- Benefits of Git to Eclipse.org:

- Committers want Git support on eclipse.org.

- Provides all contributors equal tool support
  - Non-committers equal citizens
  - Reduces contributor startup time; have all committer tools at start
  - Reduces contributor -> committer conversion bump

- More accurate recording of activity
  - Author recorded separately from committer; better IP tracking.
  - Code movement detection; better IP tracking after-the-fact.

- Open source implementations; meets mandate to avoid vendor lock-in.

- Widely popular.
  - github.com has become very high traffic site.
  - Staying power; many large projects use Git, its not going away.
  - Big current users: Qt, KDE, Linux, Android, X.org, Wine, OLPC, OpenAFS, Ruby, Perl5
  - More considering: GNOME, ASF

- Easier to move projects to eclipse.org
  - Maintain history, audit trails
  - Easier to start projects before moving to incubation status

- Current JGit Users:

- EGit; Eclipse Team Provider (EPL)
- nbgit; NetBeans Plugin (GPLv2)
- Gerrit Code Review; (APLv2)
- Apache Maven; (APLv2)
- Hudson Continuous Integration; (MIT)
- Jira Git Plugin; (BSD, EPL, Commerical)
- JetBrains TeamCity; (APLv2, Commerical)
- SCuMD; (MIT)
- gimd; (APLv2)
- Git#; Git for .NET and Mono (EDL)

- Benefits of JGit at Eclipse.org:

- IP team can track contributions, ensure project maintains IP clean code.
- IP clean code permits broad redistribution by members.
- JGit and EGit team provider can be shipped "out of the box".

  Member companies who want to provide team based data storage functionality as part of the products may benefit from redistributable team provider for Git, as their customers can rely on this stable version control system for data storage.

- Committers want Git support on eclipse.org.

Redistribution capability of EGit team provider makes redistribution of tooling for eclipse.org supported projects trivial. Meets eclipse.org mandate to only use open tools and processes which prevent vendor lock-in, or committer lock-out due to difficulties meeting required tool licenses.

- Why JGit remains EDL
  - Existing contributions:
    - 72%/~109k lines by Shawn Pearce/Google
    - 16%/~ 23k lines by Robin Rosenberg
    - remainder by 26+ others

  - Contributors won't relicense under EPL:
    - Both EPL and EDL require copyright notice to be given.

      Any copied code must retain notices, if copied from a work under either license. An argument that licensing under the EPL makes it easier for committers to copy code is mostly irrelevant here, the notice still has to be preserved across the copy. The terms of the EDL permit copying, provided that the notice is retained. As a similar requirement is in the EPL, copying code under either license has the same burden on the copier.

    - Dual EPL/EDL permits an EPL-only fork.

      By licensing code under a dual EPL/EDL relationship the combined license permits the recipient to completely ignore the other license, if they so choose. Hypothetically, a 3rd party could take a EPL/EDL JGit, make modifications, and redistribute it under only the EPL. Code from the fork cannot be brought back into the main project. A pure EDL still permits the 3rd party to create their own fork, but any modifications are more likely to be under the project license, EDL, and are more easily included back.

    - Dual EPL/EDL is a complex license.

      Some contributors find the dual EPL/EDL license to be complex, and difficult to understand. What is being asserted? What rights does the recipient have? By comparison the EDL is a very simple license, with relatively clear terms, and its use with other licenses is well understood. A change to the more complex dual license provides no tangible benefits to current contributors, and thus does not offer them an apparent reward for dealing with this more complex situation.

    - Contributors favor freedom of reuse.

      EPL is known to be incompatible with the GPL, preventing linking of EPL code into a GPL program. EDL is known to be compatible with basically every license out there, including any proprietary commercial licenses, provided notice is given on redistribution. Contributors favor giving redistribution freedom to all recipients, to promote reuse of a single implementation, rather than redevelopment of competing implementations. An EPL/EDL dual license may spawn an EPL-only redistribution channel, potentially limiting recipients from reusing under EDL.

    - Backports to C Git

      C Git is licensed under GPLv2 only. Contributors to JGit want to ensure that innovations made in JGit can be ported to C Git to continue to maintain full compatibility between the two implementations of the Git version control system.

An EPL/EDL license may make it more difficult to strip the EPL terms and backport the functionality to C under the EDL for inclusion in C Git distributions. Improvements made in C to such a backported feature might remain licensed under EDL, and thus be able to be brought back to JGit under EDL. C Git contributors are very unlikely to license any code under the terms of the EPL, but are receptive to the EDL.

- Google's Contributions:
  - Significant portion of Shawn Pearce's contributions are copyright Google.
  - Google uses Apache License 2.0 whenever possible.
    - Promotes reuse of code in all contexts.
    - Promotes contributions returning back to upstream
    - Permits reuse in commercial products where not all source is open-sourced.
    - Relatively easy to understand license.
  - BSD/MIT style is next preferred license.
    - Pretty much the same as ASLv2.
    - Lacks patent clauses, but still offers broad reuse freedom.
  - Avoid copy-left licenses in created works.
    - Sometimes restricts redistributions.
    - Sometimes restricts combination with commercial products.
    - May promote wasted effort due to reimplementations arising.
  - Avoid dual licenses in created works.
    - Complex cases created, sometimes unintended, from dual license use.
    - Risk of license forks; same concern other JGit contributors have.
    - Complicates contributions, need agreement under both licenses.
  - Avoid license proliferation.
    - Too many open source licenses, difficult to navigate space.
    - Dual license pairs create wholly new licenses (the combined license).
  - Google is very unlikely to agree to dual-license.

- **Propose JGit @ Eclipse under EDL**
  - Host JGit development at eclipse.org.
  - License only under EDL.
  - Breaks from EPL and EPL/EDL tradition.

- **When should non-EPL be permitted?**
  - Of strategic importance to the foundation's goals:
    - Provides critical services to foundation staff, committers, members.
      - Development tools for committers/members are clearly "critical services".
    - Supports the Foundation's mission of open and vendor-neutral.

> Section 1.1 Purposes. The Eclipse technology is a vendor-neutral, open development platform supplying frameworks and exemplary, extensible tools (the “Eclipse Platform”) [...] The purpose of Eclipse Foundation Inc., (the “Eclipse Foundation”), is to advance the creation, evolution, promotion, and

> support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services.

- Benefit of hosting outweighs loss:
  - Permits foundation to maintain IP cleanliness standards.
  - Permits foundation to be actively involved in future of project.

- Established, existing project:
  - Sizeable code base
  - Sizeable and active contributor base
  - Evolved several years outside of Eclipse
  - Community supports Eclipse Foundation stewardship of project

- Membership support:
  - Project has significant contributions from Eclipse members