Become a part of something bigger

Join the OSS Community

Bruno Georges
Director, Software Engineering
Red Hat Middleware

안녕하세요. 반갑습니다

A bit of history...

- Richard Stallman
  - 1977 – Added a feature to Xerox printer software
  - 1980 – Unable to change Xerox printer source
- 1983/5 – GNU / Free software foundation

Definition

- Open source software grants the rights to:
  - Modify source code (study/view...)
  - Copy
  - Redistribute – as long as license terms are kept
    - Original version
    - Modified copies
- Open-source software is very often developed in a public, collaborative manner.

Why is it good?

- Fix bugs – no need to wait for others
- Enhance it (add features)
- Reuse it – no need to “invent the wheel”
- Transparency - know your application
- More people involved
  - better tested
  - better code quality
  - less bugs

Community

- Usually done in communities
  - Very global in nature
- Communities
  - Formed around a common purpose.
  - Sharing ideas and work
  - Meritocracy = best ideas win
  - Transparency – wiki/mailing list/irc

Who can participate?

- Anyone!
- Classic model – Individuals
- Recent trend – Companies

Companies participation

- Integration with companies' products
- Charge for consulting/customer support
- Reuse open source products
- (must follow license terms)
- Report bugs/send patches

Benefits for contributors

- Write code that is used by more people
- Get personal credit
- Great way to learn a new language/framework
- **Global** collaboration

The downside

- No merge rights
- Longer review cycle / feature design
- Might be different/strict code policies
- Sometimes consensus is impossible and projects fork.

A closer look inside

Typical project structure

- Landing page
  - See: okd.io, vertx.io
- Documentation
- Community page
  - See commons.openshift.org, vertx.io/community
- Source code, Continuous Integration, Issue Tracker
  - See https://github.com/vert-x3

Roles

Contributor

- Anyone can contribute
  - Asking a question
  - Reply to a question
  - Request a feature
  - Sharing an idea
  - Reporting a bug
  - Adding/updating documentation
  - Sending a bug fix

Code reviewer

- Reviewers (code)
  - Review code contributions
  - Provide feedback (ack / nack / score if possible)
  - It is highly recommended to set a review policy

Commiter

- Has the permission to commit/push code changes
- In most cases technical only
- May be done automatically (Gerrit, Pull Requests)

Maintainer

- Well acquainted with the project/code
- Understands the project architecture
- Commit patches (sometimes)
- Build the *releases* for distribution
- Different projects - different models!
  - Single vs. multiple maintainers

Bug wrangler

- Scrub existing bugs
- Look for bugs in known places (public reports/ security)
- Check labels
  - Help Wanted - example
  - First Contribution, Quick Win….

How do I get a role ?

- Contributor - Just do it!
- Bug wranglers - Simply volunteer
- Reviewers - git clone, register if needed
- Committer - Merit based, vote might be needed
- Maintainer - Merit based, vote needed
- Roles and eligibility depend on the project governance

Defining a Healthy Community

- Many contributors and users
- Diversity!
- Users help other users
- Active participation (discussions, conferences)
- More contributions
- More installations

Tools

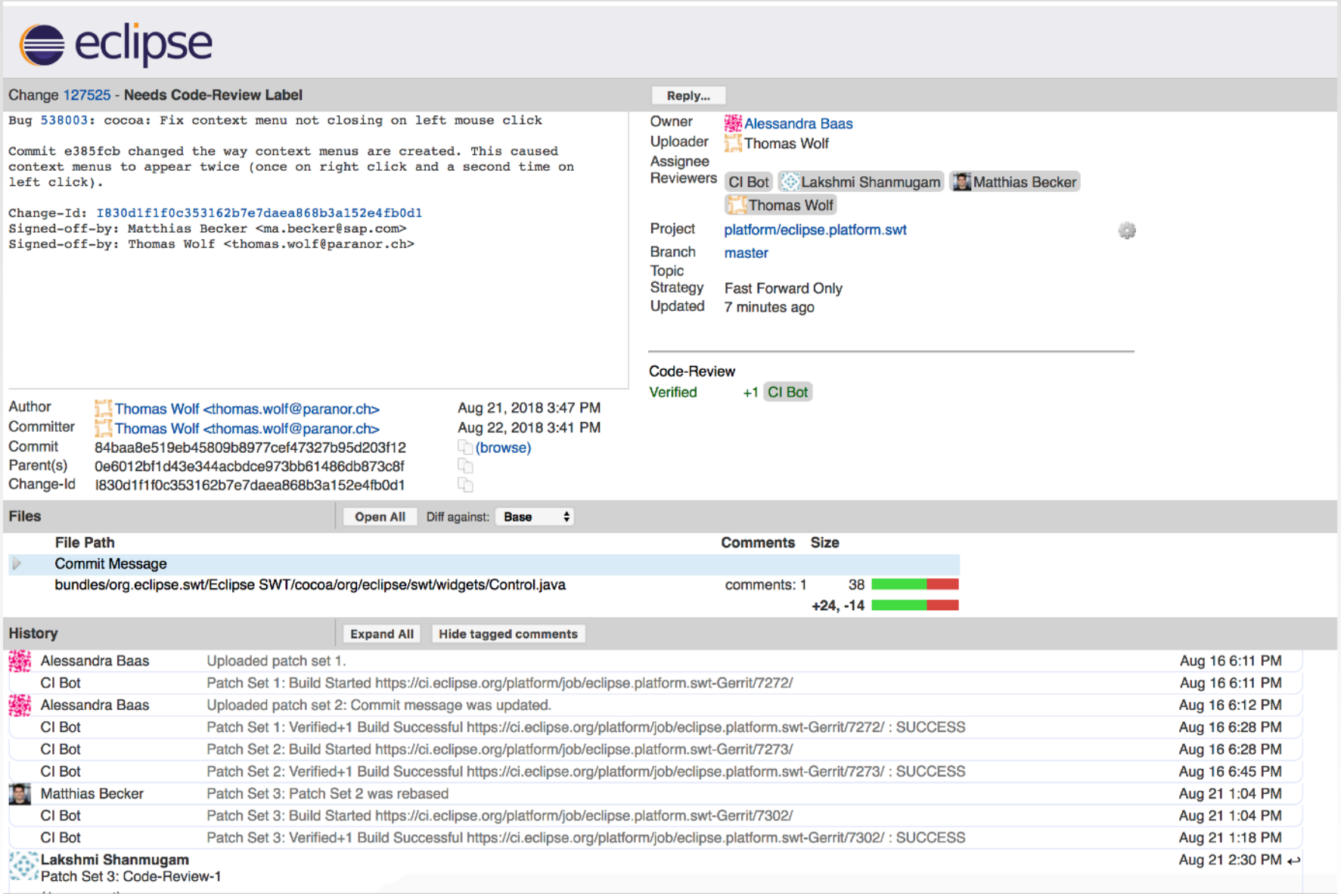
Communication

- IRC, Slack channels, Gitter, Trello, ..
- Mailing lists, Google Groups
- Forums
- Issue trackers (Github issues), Jira
- Social media

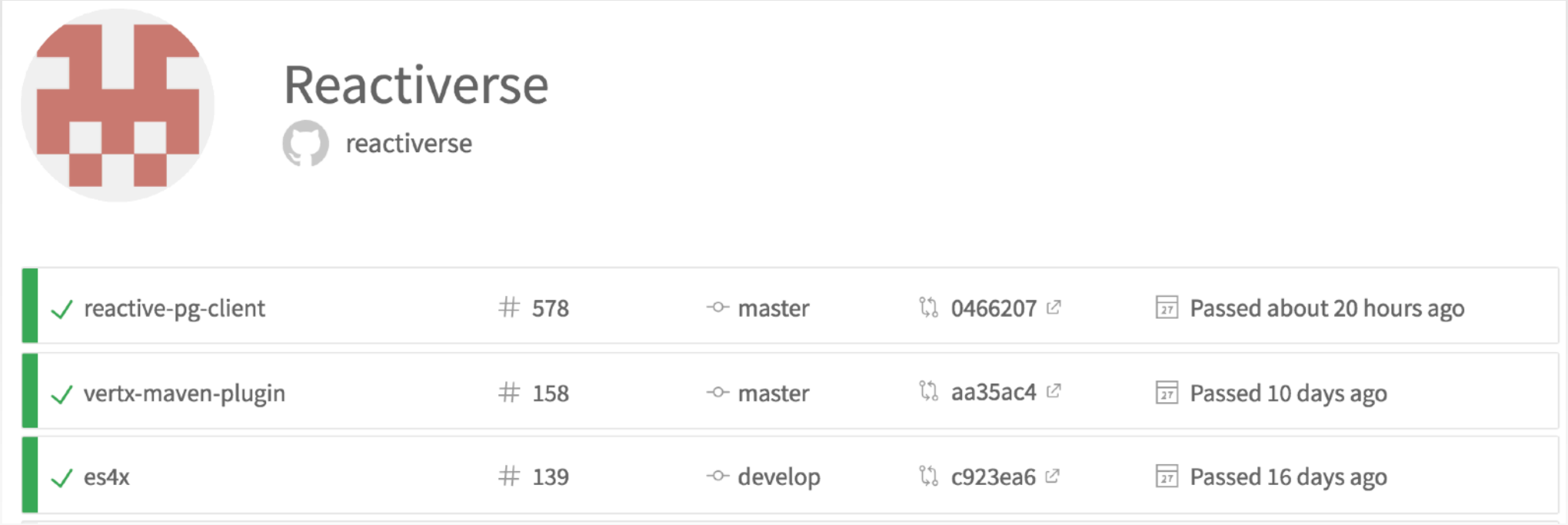
Dev tools

- Git/Mercurial/SVN etc.
- Gerrit
- Travis/Jenkins
- Github/Bitbucket/GitLab
- Github Issues/Bugzilla/Jira, etc.

Gerrit example



Continuous Integration (CI)



Licences

- Be aware of license issues
  - Not all Open Source Licenses are equivalent
  - Not all Open Source Licenses are compatible
  - https://tldrlegal.com/
- Example
  - A project using an Apache license cannot use a library released under GPL
- Tools
  - https://www.fossology.org

Licenses Example

FEATURED

"Commons Clause" License Condition v1.0
Code License managed by kevin
1988 0
2 Rules 1 Rules 2 Rules

YouTube Terms of Service
Terms of Service managed by seldon
24041 6

Sleepycat License
Code License managed by valeriedouglas
11723 4
3 Rules 2 Rules 2 Rules

Fair Source License 0.9 (Fair-Source-0.9)
Code License managed by kevin
7764 5
6 Rules 3 Rules 3 Rules

Mozilla Public License 1.0 (MPL-1.0)
Code License managed by bfitzg
12151 5
5 Rules 1 Rules 4 Rules

GNU Lesser General Public License v3 (LGPL-3.0)
Code License managed by kevin
108926 14
5 Rules 2 Rules 6 Rules

MOST POPULAR

MIT License (Expat)
Code License managed by kevin
508910 82
5 Rules 1 Rules 2 Rules

Apache License 2.0 (Apache-2.0)
Code License managed by kevin
426694 69
7 Rules 2 Rules 4 Rules

GNU General Public License v3 (GPL-3)
Code License managed by kevin
257132 44
5 Rules 2 Rules 6 Rules

BSD 3-Clause License (Revised)
Code License managed by kevin
126882 21
4 Rules 2 Rules 2 Rules

GNU General Public License v2.0 (GPL-2.0)
Code License managed by kevin
118123 15
4 Rules 2 Rules 5 Rules

GNU Lesser General Public License v3 (LGPL-3.0)
Code License managed by kevin
108926 14
5 Rules 2 Rules 6 Rules

NEWEST

PokerStar
Code License managed by Najmeh
6061 0

Dell Mohali
Code License managed by gurjot
4034 1
1 Rules 0 Rules 0 Rules

Homepage slider
Code License managed by gurjot
3709 0

kfkf
Code License managed by PulabaSravanthi
4044 0

Tunecore's Music Publishing Administration
Code License managed by Hydra9268
3505 1

Simple Non Code License (SNCL) 2.1.0
Code License managed by MysteryDash
4293 4
2 Rules 2 Rules 4 Rules

Processes and workflow

Development cycle & Patches

- Different projects – different process
- The basic concepts are the same
  - Anyone can send a patch
  - Patches that introduce new feature or new concepts are usually discussed on a mailing list
  - Patch submission should adhere to the project standards
  - Code review process
  - Ack/Nack
  - Commit

Code contribution standards

- Formatting
- Conventions
- Patch size
- Patch scope
- Documentation
- Unit tests
- Patch expiration date
- Signed-off

Feature design

- Create a draft of the feature
  - Wiki feature page
  - Blueprint
  - eMail
- Notify the mailing list, ask for opinions
- Have a discussion
  - Consensus may be needed
  - Start sending patches

Feature page example

Vert.x 4.0

- Vert.x 4 modernizes a few important aspects of Vert.x 3 and breaking changes can be expected
- Breaking changes should be kept minimal and used when it is necessary
- 3.6 users will have an easy and documented migration path

4.0 main themes

- dual asynchronous programming model `CompletionStage` / callback
- split package removal
- focus on backend clients usability and offering

CompletionStage support

The asynchronous programming model will evolve to support `CompletionStage` in addition of the callback approach. It is a fundamental change for users and thus the callback model will remain (in addition a portion of our users are fine with callbacks according to a survey).

Asynchronous methods are overloaded with a `CompletionStage` variant:

```
Future<T> asyncMethod(String s);
void asyncMethod(String s, Handler<AsyncResult<T>>);
```

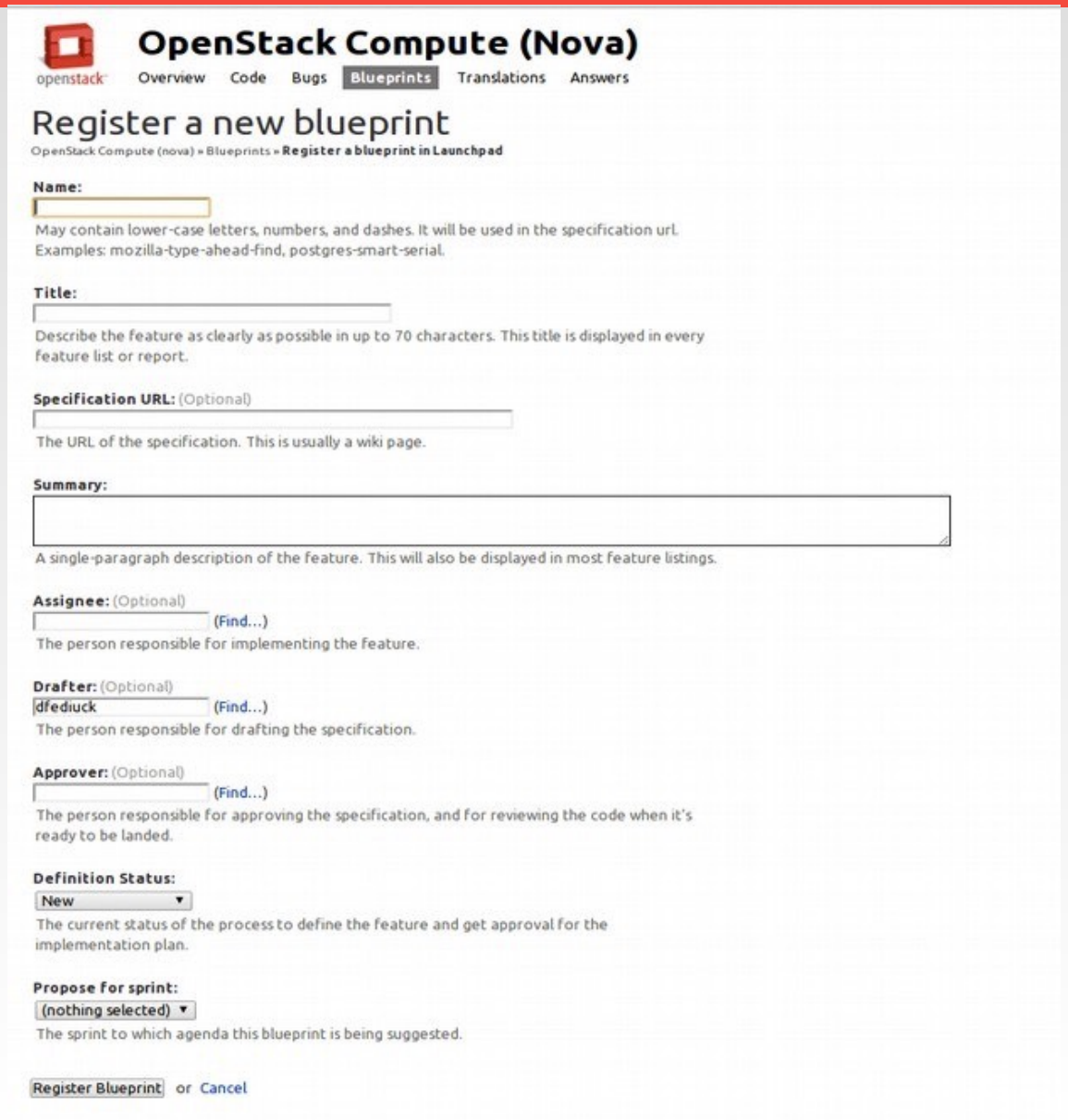
Most Vert.x asynchronous methods can be migrated to this. There are a few edge cases to care about:

- `EventBus#send(String, Object)` and `EventBus#send(String, Object, Handler<AsyncResult<Message>>)` don't have the same semantic
- `HttpClient` methods are `Handler` based and not `Handler<AsyncResult>` based

Composing a resolved future should trampoline on the event loop instead of doing a direct execution, in order to provide a non racy model that people can reason about.

Polyglot is affected since the new programming model needs to be translated into other languages.

Blueprint example



OpenStack Compute (Nova)

openstack

Overview Code Bugs Blueprints Translations Answers

Register a new blueprint

OpenStack Compute (nova) » Blueprints » **Register a blueprint in Launchpad**

Name:

May contain lower-case letters, numbers, and dashes. It will be used in the specification url.
Examples: mozilla-type-ahead-find, postgres-smart-serial.

Title:

Describe the feature as clearly as possible in up to 70 characters. This title is displayed in every feature list or report.

Specification URL: (Optional)

The URL of the specification. This is usually a wiki page.

Summary:

A single-paragraph description of the feature. This will also be displayed in most feature listings.

Assignee: (Optional)

(Find...)

The person responsible for implementing the feature.

Drafter: (Optional)

dfediuck (Find...)

The person responsible for drafting the specification.

Approver: (Optional)

(Find...)

The person responsible for approving the specification, and for reviewing the code when it's ready to be landed.

Definition Status:

New

The current status of the process to define the feature and get approval for the implementation plan.

Propose for sprint:

(nothing selected)

The sprint to which agenda this blueprint is being suggested.

Register Blueprint or Cancel

How to become a contributor

Process

- Find a project
- Start using it, build example
  - Propose documentation improvements
  - Propose examples
  - Start interacting with the community
- Look at the issue tracker
  - Pick an issue and try to propose a fix
  - Ask for help and feedback

CentOS Project

A solid, predictable base to build upon, with extensive resources to build, test, release, and maintain code.

Website | Repo

Eclipse Vert.X

A tool kit for building reactive applications on the JVM.

Website | Repo

Gluster

A free and open source software scalable network filesystem.

Website | Repo

OpenShift Origin

Enterprise Kubernetes for Developers.

Website | Repo

Pulp

A platform for managing repositories of software packages and making it available to a large numbers of consumers.

Website | Repo

WildFly

A flexible, lightweight, managed application runtime that helps you build amazing applications.

Website | Repo

All projects

Search by project name... SEARCH

Categories -- Filter by category --

A few more things

- Ask questions
  - Don’t expect synchronous answers
- Follow project conventions
  - Code style, Processes
  - Check contribution guide
- Don’t give up if your first attempt does not go through
  - Ask for feedback

Legal

- Depending on your country, you may need to ask your company for permissions
  - Contributions can be seen as a donation
  - Ownership transfer
- Software foundations have a form to sign before the first contribution:
  - CLA - https://www.eclipse.org/legal/ECA.php

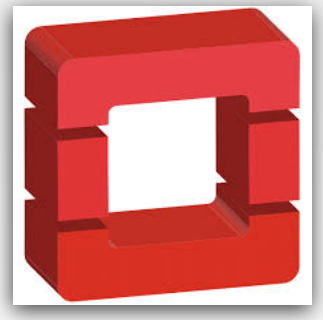
Summary

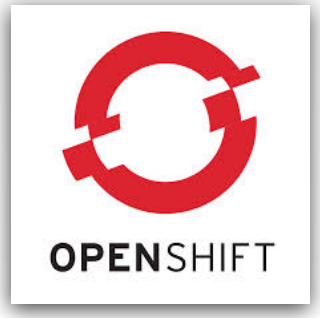
It’s all around us

Famous examples

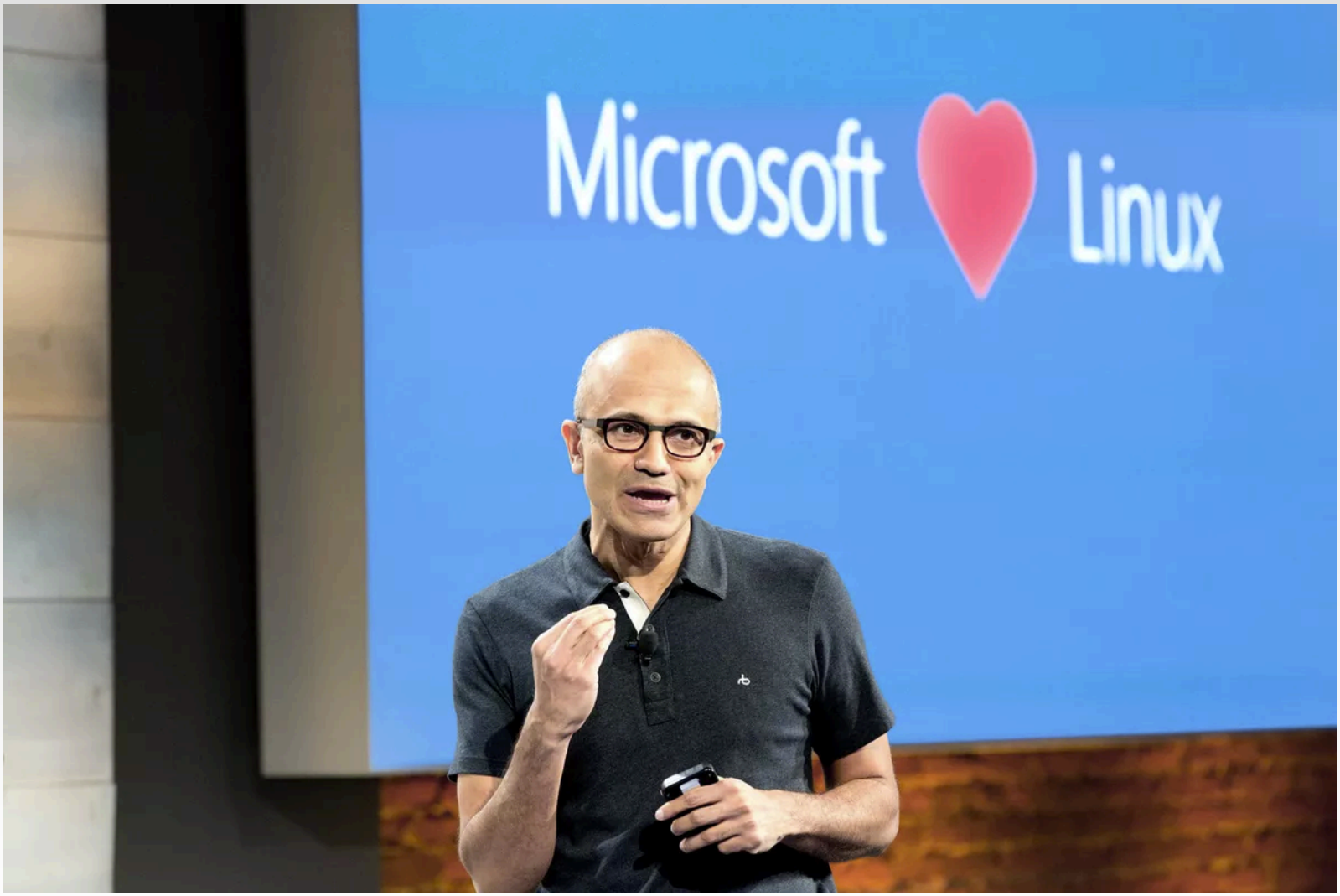
- Linux
- Mozilla- Firefox, Thunderbird, ...
- Jenkins
- Eclipse
- Git
- PostgreSQL
- Apache- Hadoop, Tomcat, Commons, ...

What's .Next ?
Going Mainstream !





MS Embraces OSS



고맙습니다. 다음에 또 만나요.

More Questions:
bruno@redhat.com