Understanding Open Source

Sebastian Mancke

Agenda

- History and definition of Open Source Software
- Basic knowledge about OSS licenses
- Working together

History and definition of Open Source Software

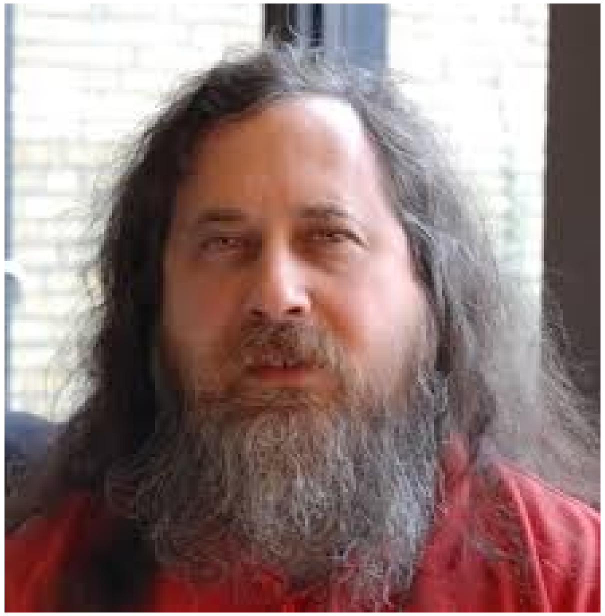
In the beginning ..

- BSD Unix

  The first free unixoid software distribution started.

- In 1980 the
  - BSD License

.. and ..
(in contrast to that)



- Richard Stallman got the Vision of free software
- In 1983 the
  - GNU Projekt and the
  - GNU Manifesto
- In 1989 the
  - GNU General Public License

.. and then

- In 25 August 1991

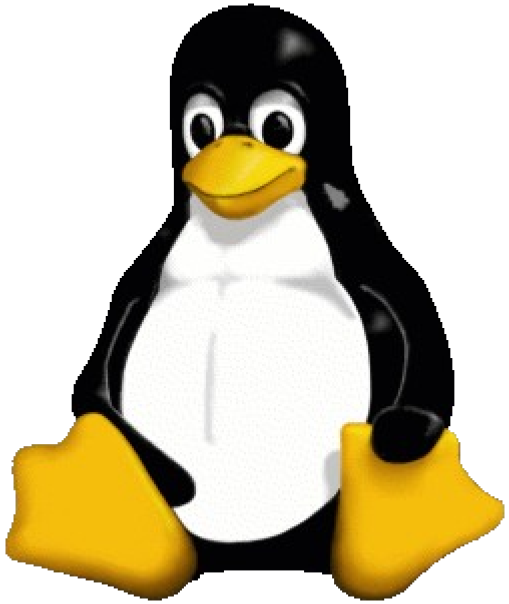
  Linus Torwalds joined the party

- He invented linux and

  put it unter GPL

- So:

  GNU/Linux was born!



Motivations for doing Open Source

market competition

pragmatism

ideology

costs

availability

velocity

transparency

independence

Why do I do Open Source?

pragmatism

As an engineer I always wanted to have the full power in my hands:

- Looking inside the engine
- Build new systems without limitations

Why do I do Open Source?

ideology

After some years of doing so,

I decided, that this is the right way for me and I don't want to use proprietary software any more.

Why do I do Open Source?

earning money

Open Source helps you being a good software developer and that's a perfect foundation for earning money!

Open Source Software hast many aspects and flavours ..

- Free of charge?
- Public available?
- ..?
- Copyleft <-> proprietary extensions?

Definition: Open Source Software

↳ tarent

- Free redistribution
- Availability of source code
- Derived Works
- Integrity of 'The Author's Source Code'
- No Discrimination
- Distribution of License
- Not Be Specific to a Product
- Not Restrict Other Software
- Technology-Neutral License

open source ™

(http://www.opensource.org/osd)

Basic knowledge about OSS licenses

- Software licenses are always a bisg issue.
- OSS Licenses are standardized and therefore give you an easy understandable framework.
- Even if a lot of different licenses exist, they can be clustered in a few license types (apart to proprietary licenses)
- It's hard to build software without OSS parts, today

=> OSS Licenses are very relevant

Definition: Open Source Software

- The
  Open Source Initiative
  lists all approved licenses:

  http://opensource.org/licenses

Most Popular

- MIT
- Apache License
- GPL
- BSD
- LGPL
- Eclipse Public License
- Mozilla Public License

Consequences of OSS license

- Depending on the individual license
  - strong copyleft
  - weak copyleft
  - non copyleft
- Depending on the usage type
  - Internal usage (e.g. build toolchain)
  - Network services
  - Unmodified delivery
  - Modified delivery
  - Dependencies in own components

Copyleft

All derivative work of code under a copyleft license must themselve be under the same or at least comparable license.

Copyleft

- When is a work 'derived'?
  - Modification of the base project
  - Substantially dependency to the base project
  - Statically linked
- What's about dynamic linking?
  - Borderline case!
  - Depends on the type of dependency
  - Conservative approach: derived work!

Copyleft

- Weak copyleft:
  - Modifications of the code themselves has to be the same license
  - Components who use the software are free to get another license (e.g. linking libraries)
- No copyleft:
  - Derived work can be released under a different license
  - Even proprietary software based on those software is possible.

Usage: Internal

Usage example: build tools, compiler, ...

- Normally: No consequences
- Exception: Tools which produce source code (e.g. MDA processors). If the created source is more then a simple template, it's library has to be considered.

Usage: Unmodified Delivery

Usage example: E.g. delivering OSS software packages.

- Non copyleft licenses
  - Attributation has to be preserved
- Copyleft licenses
  - Detailed inclusion of license information
  - Provision of the source code has to be ensured

Usage: Modified Delivery

Usage example: Modified open source software

- Non copyleft licenses
  - No or small consequences
  - Attributation has to be preserved
- Copyleft licenses
  - Releasing the whole work under the free license
  - Detailed inclusion of license information
  - Provision of the complete modified source code

Usage: Dependencies in own components

Usage example: Using libraries in your software

- Weak or non copyleft licenses
  - Free to choose a license
  - Attributation has to be preserved
- Licenses with strong copyleft & derived work
  - Releasing the whole work under the free license
  - Inclusion of license information
  - Provision of the complete source code of all parts of the component

BSD-Style & MIT License

- Free distribution
- No copyleft
- Free choice of the license for a derived work
- No duty to publish source code

Obligations:

- Preservation of copyright information
- For source redistribution of modified work:
    - License information
    - Marking of modifications
    - Changing the project name

Apache License v2

- Free distribution
- No copyleft
- Free choice of the license for a derived work
- No duty to publish source code
- Patent clause

Obligations:

- Preservation of copyright information
- For source redistribution of modified work:
  - License information
  - Marking of modifications
  - Changing the project name

GPLv2

- Free distribution
- Strong copyleft

Obligations:

- Preservation of copyright information
- Provision of the complete source code
- Changing the project name (e.g. trademark issues)

GPLv3

- Similar to GPLv2
- Saves the users of a software against patents from the originator
- Anti DRM Clause
- Compatibility with the Apache License V2

Problems:

- Through the copy left, the GPLv2 and GPLv3 are not combinable. So better publish GPL software as

  'GPLv2 or any later'

LGPLv2

- Free distribution
- Weak copyleft:
  - Full copyleft for modifications on the software
  - No copyleft for software that „uses“ the library or component. (e.g. static and dynamic linking is allowed)

Obligations for „usage“:

- For the LGPL library, same as with GPL
- For the dependent software: providing possibilities to modify the original library (e.g. object code for re-linking).

Afero GPL (AGPL)

tarent

Similar to GPL, but

- Expand the copyleft also to users of the software within the same network.

Obligations:

- The source code has to be available for every user of a services
- Effective protection against proprietary usage

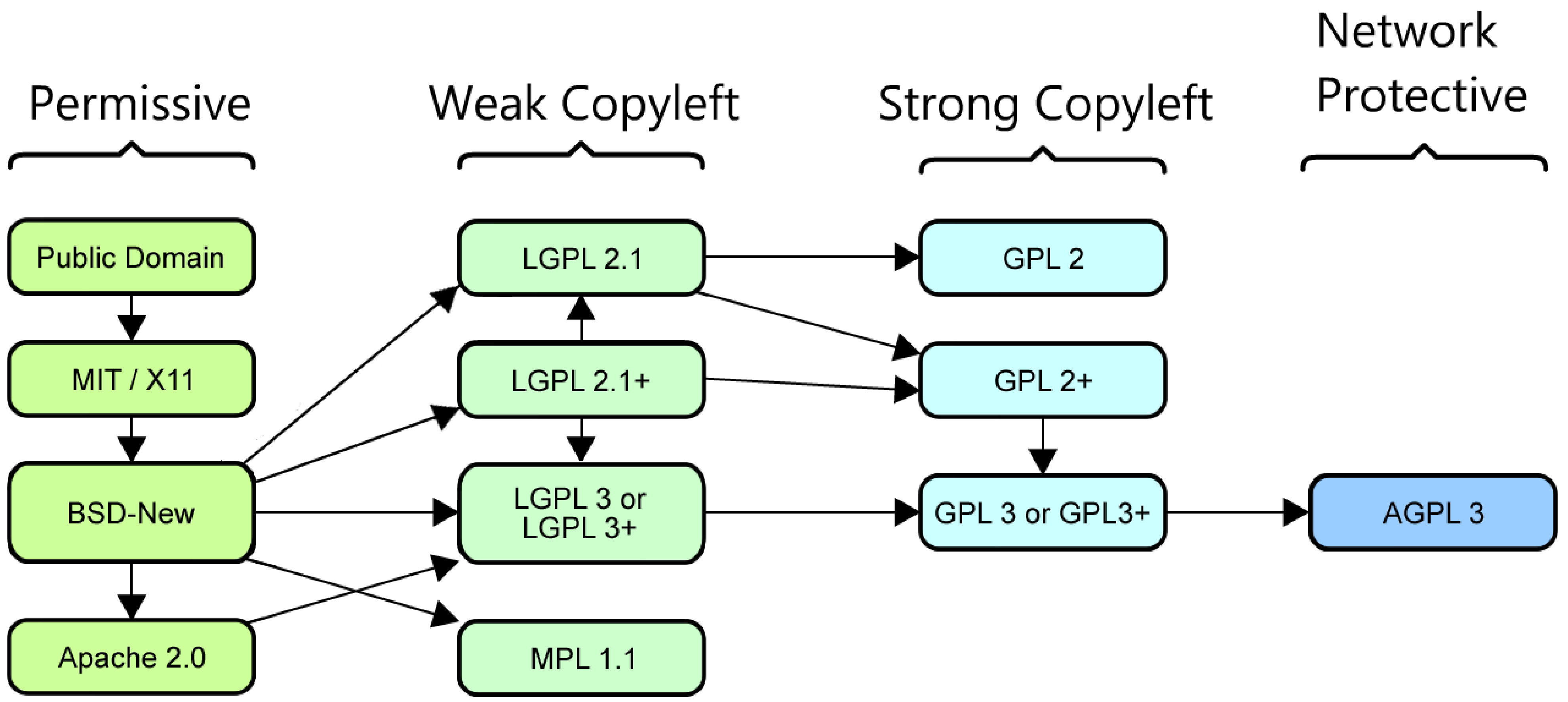
Eclipse Public License

- Free distribution
- Strong copyleft
- No derivative works through usage or linking.
- No derivative works through addition.

Obligations:

- Preservation of copyright information
- Provision of the complete source code
- Patent grants for contributions
- Changing the project name (e.g. trademark issues)

Overview and compatibility



Providing source code

- Which parts have to be provided?
  - Depending on the license: all parts with copyleft
  - The exact version for the binary!
  - All special tools which are necessary to build the binary (e.g. build script, special compiler, ..)
- To whom?
  - Often misunderstood: To the person, which has obtained the binary, only.
  - No duty to publish the source to the whole world if you build an application for a customer!

Working together ..

GitHub

GitHub has become the main platform for open source collaboration:

- Git as foundation
- Forking as default cooperation model
- Markdown for documentation
- Issue tracker
- Hosting of static pages
- Open api for integration (e.g. in build services)

Public build services

Services, which offer build systems for open source projects: e.g. travis-ci.org

- Configuration by travis.yml
- Integration with GitHub
- Public viewable build status

Collaboration on systems

Container technologies bring collaboration to the next level:

- People are able to exchange whole software systems in an easy way.
- Other than linux distributions, they may be preconfigured for special usecases.

.. and what's next?

What has happened since 2005?

The (software) world has changed:

- Workstations and desktop software don't play a role anymore.
- Web based services have made the race!

Open Source has changed

In the beginning		Today
Individuals	→	Big Companies
Applications	→	Libraries, tools
User focussed	→	Developer focussed
GPL		MIT

Open Source has changed

On the good side:

Today no company can compete without using open source!

On the bad side:

The evolution to use cloud based services makes it easy to ..

.. use open source without contribution

.. lock users

Questions for the future

How to avoid vendor logins:

Open standards!

Open APIs!

The freedom to stay owner of your data!

Thank you!