Practical License Detection for Organizations

Philippe Ombredanne
AboutCode & nexB
https://aboutcode.org

Abstract

Does Richard Stallman work for your organization? Can’t use the Anyone But Richard Stallman License then…

Vanity licenses exist. Obscure licenses exist. We may laugh or argue about why they were published, But how do we handle these license exceptions? If we don’t acknowledge and deal with all the licenses that exist in published software, we will keep having license exception headaches.

Many software license detection tools have two key problems:

- A too-narrow focus on the subset of common open source licenses while ignoring the high volume of less common licenses (the long tail) that exist in the real world.
- Exclusion of proprietary or commercial licenses that intersect or overlap with open source licenses, such as “dual” licenses.

For OSPOs to work with real data – not just theoretical assumptions – we need FOSS license detection tools that cast a wide net to handle all open source licenses, and future versions too. And we need FOSS tools to efficiently review these licenses and apply license policies across product codebases, updates, and versions.

The goal: Identify all licenses fast in a clean process with minimal license detection exceptions. By using FOSS tools and setting clear policies, OSPOs can empower organizations to quickly identify licenses and apply policies without asking developers to read license texts.

The result: Triaging license exceptions is easy.

Philippe Ombredanne

- On a mission to make it easier to reuse more FOSS, safely and efficiently
- Project lead and maintainer for **VulnerableCode**, **ScanCode** and AboutCode
- Creator of Package URL, co-founder of SPDX & ClearlyDefined, contributor to CycloneDX
- FOSS veteran, long time Google Summer of Code mentor
- Co-founder and CTO of nexB Inc., makers of DejaCode
- Weird facts and claims to fame
  - Signed off on the largest deletion of lines of code in the Linux kernel (but these were only license comments)
  - Unrepentant FOSS code hoarder. Had 60,000+ GH forks, now only 20K forks
- pombredanne@nexB.com irc:pombreda

Agenda

- ▷ Why should we care?
- ▷ In the beginning, there were licenses
- ▷ The situation
- ▷ The plan to fix this
- ▷ The many curious ways of licensing
- ▷ Licensing Easter eggs
- ▷ Scanning your way out
- ▷ Other excellent tools
- ▷ Scoring the license clarity
- ▷ Automating the package license review
- ▷ The many curious ways of copyright
- ▷ License compliance in practice: Five simple rules
- ▷ Future steps: Fix all the licenses, Open all the data

Why should we care?

- When I re-use third party code, I want to ensure that:
- **I am authorized to use it**
- **I comply with licensing conditions**
- **And also:**
    - *I am not using vulnerable code that can be exploited*
    - *I am not using buggy or low quality code that can crash*
    - *I am not using unmaintained or unsustainable code*

    - *And I can share this downstream **(SBOMs)***
    - *And I can proactively and continuously manage these*

In the beginning, there were licenses

open source is defined by licenses

In the beginning, there were licenses

open source is defined by licenses

- ▷ The big advantage of FOSS licenses over proprietary is ...
  - Not because this is open
  - But because there are few, well known terms:
    - Few like a couple 1000's, and 10 to 20 common
    - codified identifiers with **SPDX**
  - In contrast, every proprietary license is unique
- ▷ **Yet FOSS is still a chaotic mess**
- ▷ **Chaoss is hard to automate**

The situation(1): A jolly babel mess

- ▷ Ever **more FOSS software packages** are reused
  - *10x to 100x more than a few years ago*
    - *Apps with 10K packages are common*
  - *Yes! we can really build applications from components!*
- ▷ **Yet clear and accurate metadata** are direly missing
  - Software metadata is a messy babel
  - No metadata means no automation possible
- ▷ **Clarity in licensing** is far from being there

The situation(2): Light at the end of tunnel?

- ▷ We have defined a decent way to talk licensing
  - SPDX license expressions
- ▷ Evolution of license detections tools and techniques
  - Start to reach the point of being helpful
- ▷ The hype around SBOMs is for security
  - And this positively highlights the need for clear licensing in SBOMs

The plan to fix this

- ▷ Assess the many curious ways of FOSS documentation for
  - license
  - copyright
- ▷ Detect all the licences accurately
- ▷ Follow simple rules
- ▷ Beyond: Fix all the licenses

The many curious ways of licensing (1)

- **Python**: license string and/or a list of classifiers, ambiguous.
- **Ruby**: license string; or a list; should be SPDX, not enforced.
- **Apache Maven**: list of (name, URL and comment). List is ambiguous. No defined ids or names.
- **CPAN**: a string or a list. List is ambiguous. Custom ids.
- **Debian**: structured copyright files, but not all. Few own ids custom otherwise.
- **Fedora**: own license ids and license file RPM tags.
- **npm**: SPDX expression with custom extensions. Or a list. Or an object.

The many curious ways of licensing (2)

- **FreeBSD**: own license field, switching to SPDX expressions
- **ArchLinux**: own license ids. No expressions.
- **NixOS**: SPDX ids with custom extensions. No expressions.
- **Alpine Linux**: recommends SPDX expression, but not enforced. Use custom ids.
- **R/CRAN**: use own ids and notation for expressions
  - with a mathematical approach to expand license versions
  - such as LGPL (>= 2.0, < 3)
- **Android**: use MODULE_LICENSE_BSD empty tag files and NOTICE files.

The many curious ways of licensing (3)

- ▷ **Haskell**: SPDX expressions and license files
- ▷ **Erlang/Elixir**: license list, recommends SPDX.
- ▷ **NixOS**: SPDX ids with custom extensions.
- ▷ **Go**: no license metadata. At least, no new mess!
- ▷ **openSUSE**: SPDX expressions with custom extensions
- ▷ **Gentoo**: own license expression syntax and custom ids
- ▷ **Rust**: slash-separated SPDX ids and license-file field
- ▷ **PHP Composer**: SPDX ids with custom extensions
- ▷ **NuGet**: license URL or SPDX expression or file path

Linux kernel licensing Easter eggs

- **There used to be 600+ ways to write a GPL notice!**
- HOT: In the kernel drivers/w1/slaves/w1_therm.c thermal drivers code

```
This program is free software; you can redistribute it and/or
modify it under the therms of the GNU General Public
License...
```

- Shy or obfuscated: Seen in a GPL-shy external kernel module

```
MODULE_LICENSE("\x47\x50\x4c\x20\x76\x32");
```

- (e.g. GPL v2 in ASCII)
- Terse: In drivers/mtd/chips/cfi_cmdset_0001.c:

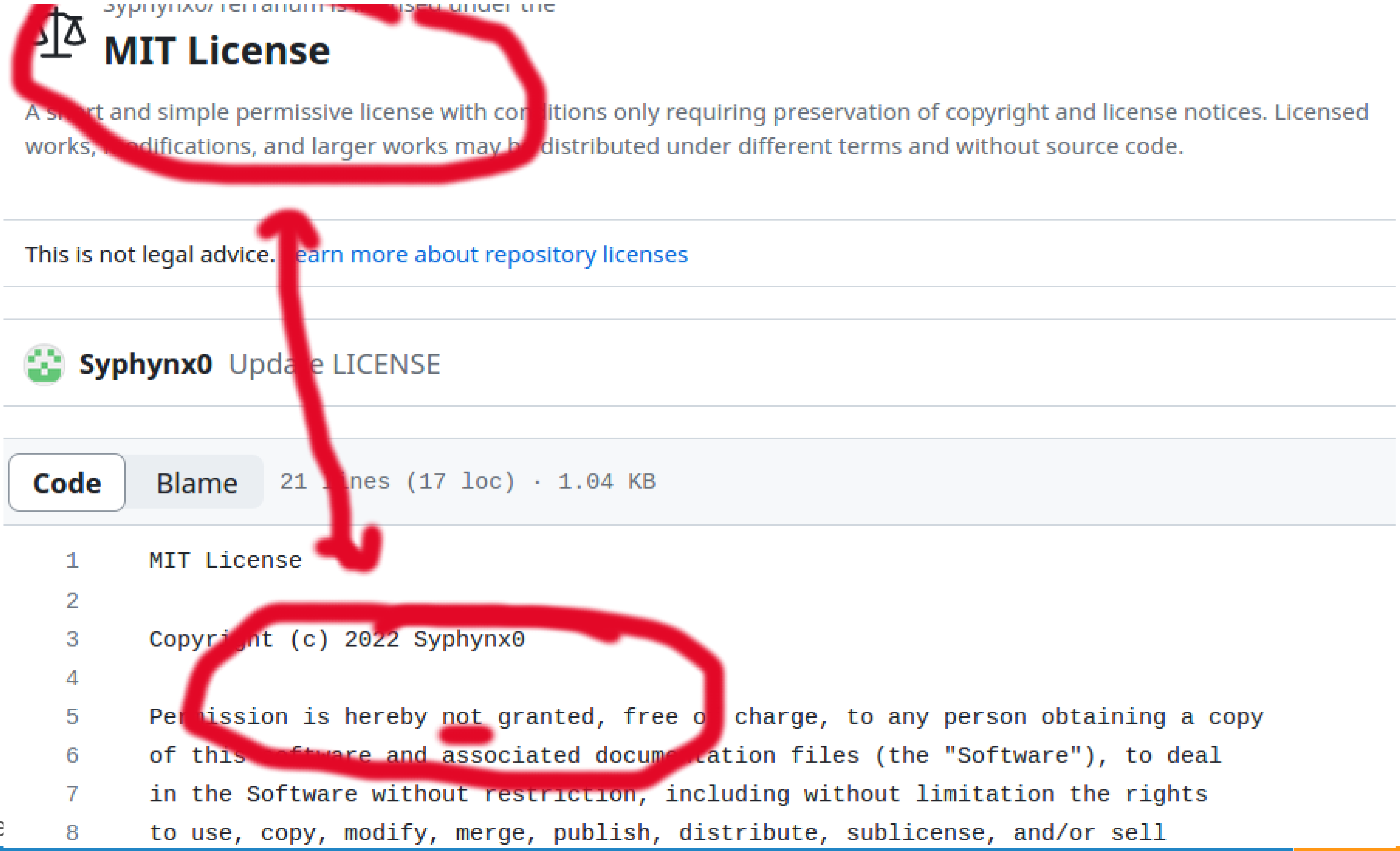
```
(C) 2000 Red Hat. GPL'd
```

Scanning your way out

- We have created five generations of detection engines
- Creating mappings for all of custom license ids is hard
- Making sense of all the subtly different conventions is hard
- Ambiguity is not a great idea for licensing
- The solution was eventually to write a license scanner with 2,000+ licenses and 32,000+ notices, mentions and tags combining multiple advanced text matching techniques
- **Problem mostly solved, now ScanCode toolkit is the industry leading tool**

But wait there are plenty of excellent tools!

- ▷ Scanning the ~70K files of the Linux Kernel
  - ○ Over 16K files **not** detected by another FOSS scanner
- ▷ Or this:



Syphynx0/Terrarium is licensed under the

MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

This is not legal advice. Learn more about repository licenses

Syphynx0 Update LICENSE

Code Blame 21 lines (17 loc) · 1.04 KB

```
MIT License

Copyright (c) 2022 Syphynx0

Permission is hereby not granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

Scoring the license clarity

- ▷ Declared license? ✓
- ▷ Identification precision? ✓
- ▷ License text present? ✓
- ▷ Declared copyrights? ✓
- ▷ No Ambiguous compound licensing? ✓
- ▷ Non Conflicting license categories? ✓

Automating the package license review

- License clarity scoring
- Automated false positive analysis and fixing
- Upcoming TODO option that tells what would needs review

The many curious ways of copyright

```
Copyright (c) 2006 Jane Doe
```

- ▷ **Looks simple, what could go wrong with this?**

```
Copyright (c) Jane Doe, 2004-2006
Jane Doe (c) copyright 2006 and others
```

- ▷ **Yet all the possible permutations exist in the wild**
- ▷ The solution was eventually to write a natural language parser and a grammar with 1,500+ lexing rules and 300+ parsing rules
- ▷ ScanCode is the industry standard tool

License compliance in practice: Golden rule #1

- Golden rule #1
  - Keep all original license, copyright and notices
  - Add them back if they are missing
  - You cannot detect what does not exist

In practice: rule #2 - Policy

- Define your license policy. The simplest is based on category:
  - Copyleft: OK or not and when
  - Limited Copyleft: or not and when
  - Permissive: OK
  - Funky and proprietary: not OK
- Every ScanCode license comes with a license category
- If you start from scratch, keep it simple and short

In practice: rule #3 - Use the source, Luke

- Ensure that you keep a copy of the source code of everything
  - All the source of all the FOSS packages you use
- Beyond being good engineering this is essential for compliance
- Do not think you will be able to efficiently chase these later
  - It is no fun and in some cases impossible
- This open "SOURCE" after all!
- This can be automated as part of the builds for package repos
- Just do it and you will thank me someday

In practice: rule #4 - Scan for licenses

- Use ScanCode to scan for licenses
- Use the license clarity score
- Apply your policy
- Trigger review of things that are out of the originary
  - Funky licenses
  - Proprietary, Dual proprietary or copyleft
  - Inaccurate detections, low clarity
  - License changes
- Developers may not know how do this review, and legal or engineering or an OSPO can help

In practice: rule #5 - Comply!

- Generate an attribution notice and/or SBOM
  - Use ScanCode for this (or DejaCode or AboutCode TK)
  - Automate as part of your release builds
  - Over-attributing a little is not a big problem
- Redistribute the sources
  - Simplest is to redistribute ALL the sources

Fix all the licenses! **GO UPSTREAM**

- ▷ Build a **babelfish universal translator for licenses**
  - Done with ScanCode toolkit
  - Output are clear, normalized licenses
  - Think of it as a **license and copyright linter**
- ▷ Scan all the code, then review, then fix **upstream**
  - In progress. Need to find a way to scale up. Apply ML to spot inconsistencies
- ▷ Work with package ecosystems **towards shared ids and conventions**
  - In progress for instance at Python
- ▷ Organize improvements campaign to fix everything **upstream**
  - In progress, planning some for **Debian and alpine**

Open all the data!

▷ VulnerableCode - **All the vulnerabilities**

- free correlated vulnerabilities DB (with support from the EU and NLnet.nl)

▷ LicenseDB - **All the licenses**

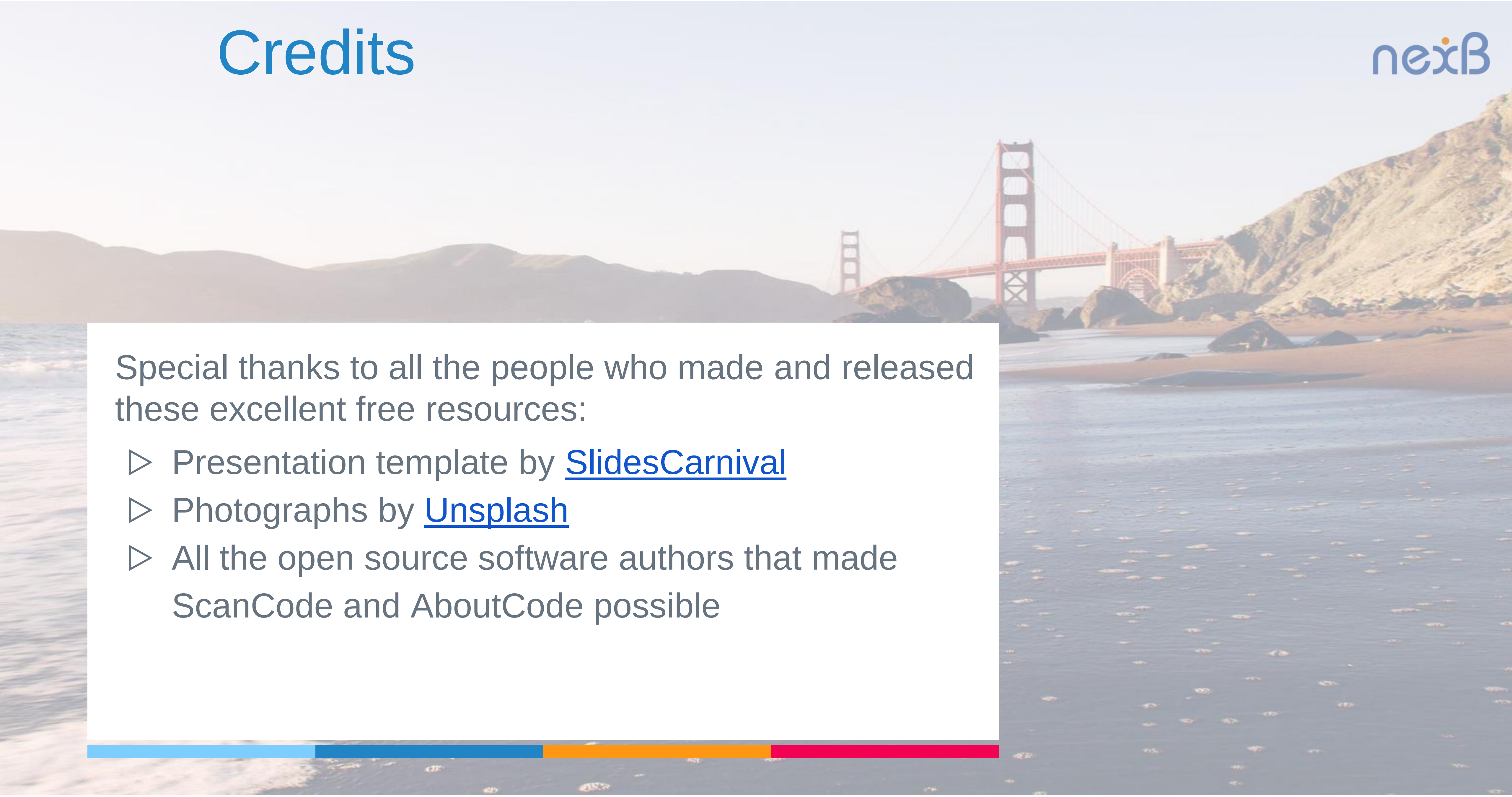
- FOSS or proprietary - https://scancode-licensedb.aboutcode.org/

▷ PackageDB - **All the packages** and dependencies metadata

- Upcoming and will include ClearCode - Extract all the data and ScanCode scans from ClearlyDefined, deployed also at Software Heritage

If you want to help

- ▷ You can contribute code, time, money
- ▷ Join the conversation at
  - https://gitter.im/aboutcode-org
- ▷ Donate money at
  - https://opencollective.com/aboutcode



Credits

Special thanks to all the people who made and released these excellent free resources:

- ▷ Presentation template by [SlidesCarnival]
- ▷ Photographs by [Unsplash]
- ▷ All the open source software authors that made ScanCode and AboutCode possible