Beyond SBOMs

Making license data actionable with ClearlyDefined

Jamie Magee · Microsoft

Open Source Summit North America · 2026

The NOTICE file you cannot write

A real situation, lightly disguised

1. You are about to ship a product.
2. Legal asks for the third-party notices.
3. You generate an SBOM. Hundreds of dependencies.
4. You paste the license fields into a template.
5. Half of them say `NOASSERTION`.

SBOMs answer
"what is in it."

They rarely answer
"what am I allowed to do with it?"

SECTION 1

Why SBOMs fall short on licensing

Gap 1 · Missing

```
{
  "name": "some-package",
  "versionInfo": "2.4.1",
  "licenseDeclared": "NOASSERTION",
  "licenseConcluded": "NOASSERTION"
}
```

The SBOM generator only knows what the package manager metadata told it. If the package's `package.json` or `pom.xml` has no license field, the SBOM has no license field.

Gap 2 · Ambiguous

`package.json`	**"license": "MIT"**
`LICENSE file`	**Apache License 2.0**
`src/*.js headers`	**SPDX-License-Identifier: MPL-2.0**

Three sources, three answers. The SBOM generator picks one and moves on.

Gap 3 · Attribution

Declared license is correct. The package vendors three other libraries. Each ships its own copyright notice.

Your SBOM has one license field.

SPDX and CycloneDX *can* express this. SBOM generators rarely populate it, because nothing scanned the source.

Why this happens

SBOM generators read package metadata. They do not scan source code.

Source scanning is slow, expensive, and produces messy output that needs a human to interpret. So most generators skip it. The license column gets filled from whatever the package metadata returned.

SECTION 2

Enter ClearlyDefined

What it is

An open, crowdsourced database of license and attribution metadata for published open source.

An OSI project. Started in 2017. Operated today with infrastructure and contribution from Microsoft, GitHub, SAP, Bloomberg, and others.

Two flows

Harvest

How the data gets *in*.

Scanners run automatically against every published package. You do not have to do anything.

The machine half.

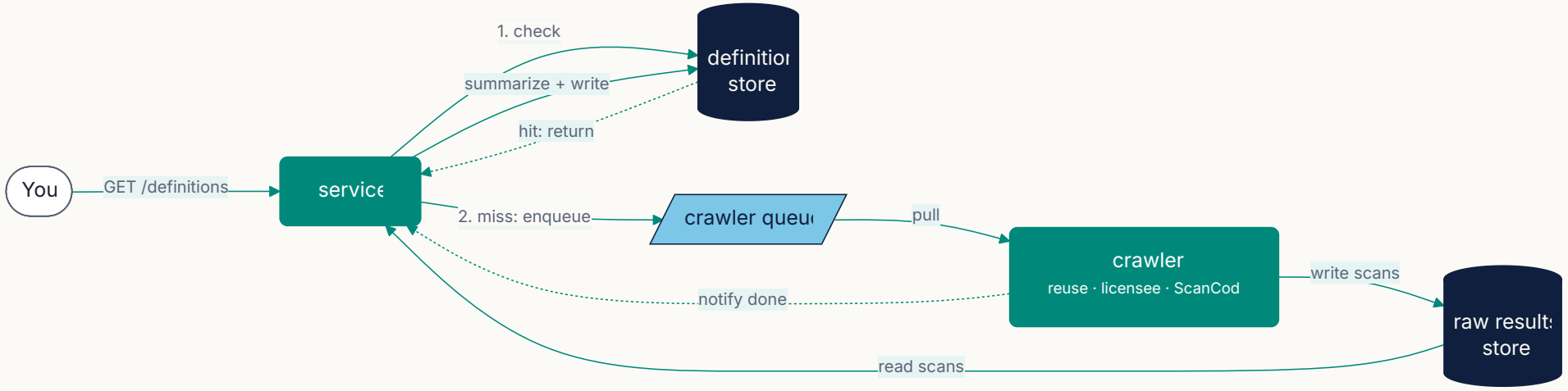
Curate

How the data gets *fixed*.

Humans propose corrections when the scanners get it wrong. Other humans review them.

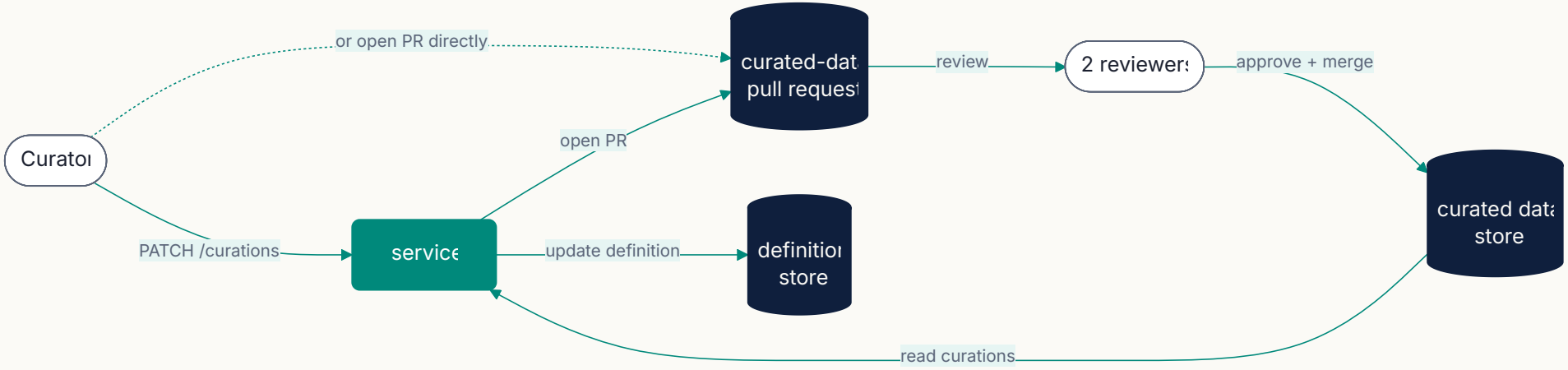
The people half.

Two halves · Harvest



Service checks the definition store first. On a miss, it enqueues a job; the crawler pulls, runs the scanners, writes raw output, and notifies the service to summarize.

Two halves · Curate



One service, the other side. Curators submit a patch; two humans review; the next definition build picks it up.

Coordinates

```
type / provider / namespace / name / revision

npm / npmjs / -       / lodash    / 4.17.21
maven / mavencentral / org.apache.commons / commons-lang3 / 3.14.0
pypi / pypi / -       / requests  / 2.31.0
git / github / clearlydefined / curated-data / HEAD
```

One identifier scheme across every ecosystem. PURL conversion exists; coordinates predate PURL and are still the primary key today.

SECTION 3

One component, end to end

Stage 1 · The SBOM entry

A real entry for `uuid@9.0.0` from a typical SBOM.

```
{
  "name": "uuid",
  "versionInfo": "9.0.0",
  "licenseDeclared": "MIT",
  "licenseConcluded": "NOASSERTION",
  "copyrightText": "NOASSERTION"
}
```

`package.json` says MIT. The generator believed it. Spoiler: three files in the package disagree.

Stage 2 · Ask ClearlyDefined

```
curl https://api.clearlydefined.io/definitions/\
npm/npmjs/-/uuid/9.0.0
```

No auth. No rate-limit headache for a single lookup. Returns a definition document.

Stage 2 · The definition


```
{
  "coordinates": { "type": "npm", "name": "uuid", "revision": "9.0.0" },
  "described": {
    "sourceLocation": {
      "type": "git", "provider": "github", "namespace": "uuidjs", "name": "uuid"
    }
  },
  "licensed": {
    "declared": "MIT",
    "facets": {
      "core": {
        "discovered": {
          "expressions": [ "MIT", "MIT AND BSD-3-Clause"]
        },
        "attribution": {
          "parties": [
            "Copyright (c) 2010-2020 Robert Kieffer and other contributors",
            "Copyright 2011, Sebastian Tschan https://blueimp.net",
            "Copyright (c) Paul Johnston 1999 - 2009 ..."
          ]
        }
      }
    }
  }
}
```

Stage 3 · The conflict

`declared`	**MIT** (from package.json)
`discovered`	**MIT, and MIT AND BSD-3-Clause in `dist/*md5*.js`**
`attribution`	**3 copyright holders, 2 not in the SBOM**
`your NOTICE file`	**missing BSD-3-Clause attribution**

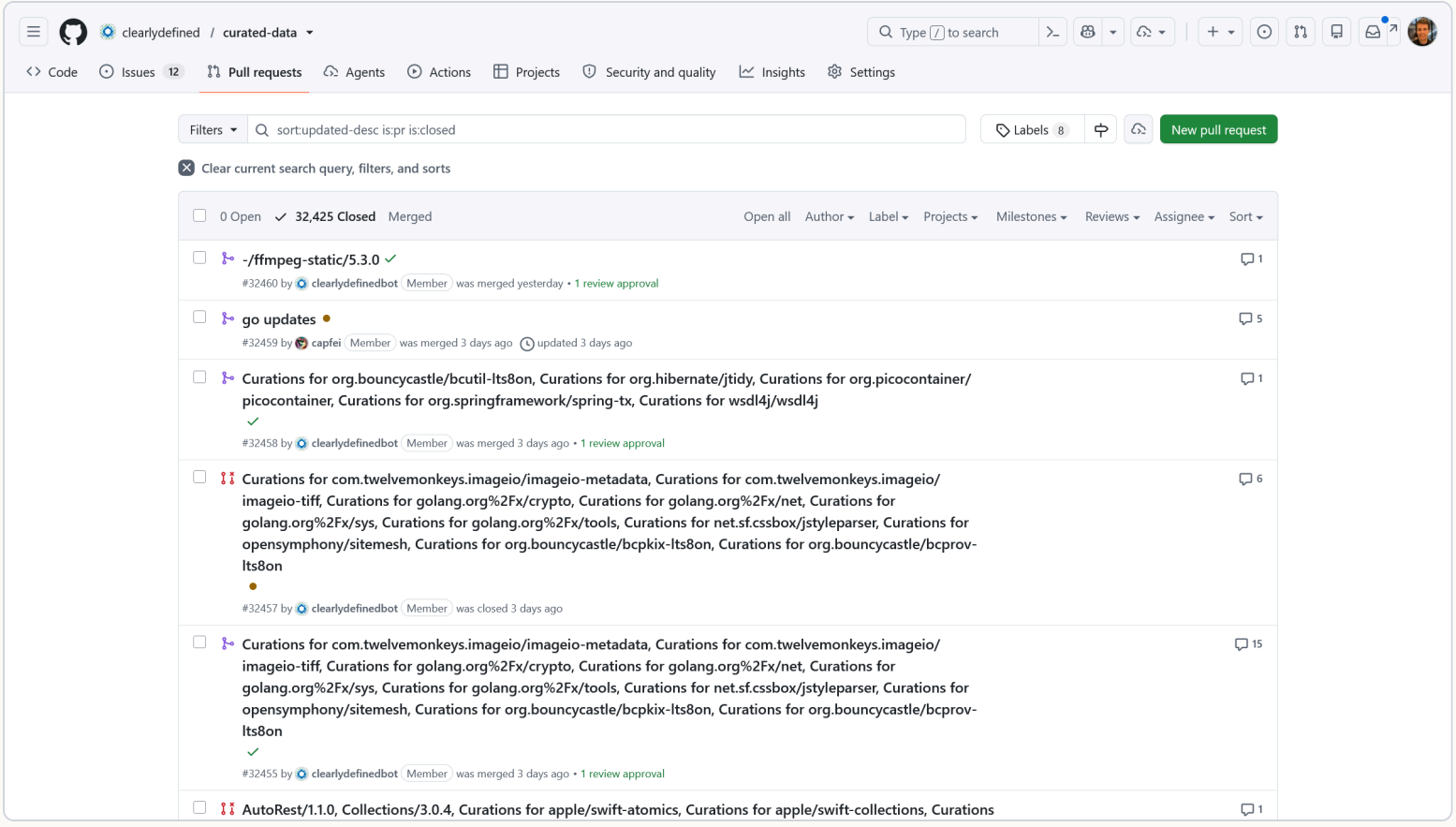
uuid bundles a public-domain-style md5 implementation by Paul Johnston. BSD-3-Clause has a real attribution clause. Your NOTICE file owes him a line.

Stage 4 · Curate

A curation is a patch on the definition. You can write one in the UI or POST one to the API.

```
{
  "patch": {
    "licensed": {
      "declared": "MIT AND BSD-3-Clause"
    }
  },
  "contributionInfo": {
    "summary": "Fix declared license",
    "details": "dist/*md5*.js bundles BSD-3-Clause code by Paul Johnston.",
    "type": "missing"
  }
}
```

Stage 4 · Pull request



Stage 5 · Fixed


```
{
  "coordinates": { "type": "npm", "name": "uuid", "revision": "9.0.0" },
  "described": {
    "sourceLocation": {
      "type": "git", "provider": "github", "namespace": "uuidjs", "name": "uuid"
    }
  },
  "licensed": {
    "declared": "MIT AND BSD-3-Clause",
    "facets": {
      "core": {
        "discovered": {
          "expressions": [ "MIT", "MIT AND BSD-3-Clause"]
        },
        "attribution": {
          "parties": [
            "Copyright (c) 2010-2020 Robert Kieffer and other contributors",
            "Copyright 2011, Sebastian Tschan https://blueimp.net",
            "Copyright (c) Paul Johnston 1999 - 2009 ..."
          ]
        }
      }
    }
  }
}
```


Same curl, after the PR has been merged. Now the NOTICE file writes itself.

SECTION 4

Using the data

1. The API, directly

```
curl https://api.clearlydefined.io/definitions/\
npm/npmjs/-/uuid/9.0.0
```

Public, no auth, JSON. Batch endpoints exist if you have a list. You can wire this into a build step, a CI check, or a Slack bot.

2. cdsbom

```
go install github.com/jeffmendoza/cdsbom@latest
cdsbom -out enhanced-sbom.json input-sbom.json
```

Reads SPDX or CycloneDX. Replaces license fields with curated data from CD. Writes the enhanced SBOM in the same format.

The Linux Foundation runs this across audits for their projects. Roughly 1,200 projects covered.

3. GUAC

GUAC is OpenSSF's supply-chain graph. As of v0.8 it ships a ClearlyDefined certifier that pulls license data into the graph alongside SBOMs, vulnerability reports, and attestations.

Same data, different shape. If you are already running GUAC, the license layer is now there.

40M+

licenses
on github.com

4. github.com

License badges, dependency graph, dependency review, repository SBOM exports.

If you have looked at a license on github.com lately, you have probably already used ClearlyDefined.

SECTION 5

Why contribute back

The alternative is rot

Without curation, every consumer rescans the same package and re-fixes the same broken metadata in private.

Compute is wasted. Fixes are not shared. The next team learns nothing.

One curation. Every downstream consumer gets it for free.

Maintainers in the room

The cheapest way to control how the world sees your project's license is to curate your own ClearlyDefined definition.

Five minutes of work. Goes through your pull-request review like any other patch. Stops a thousand downstream teams from guessing.

SECTION 6

Try it this week

Three things to do this week

1. `curl` the API for one package you ship.
2. If something is wrong, file a curation.
3. Drop in to the weekly community meeting.

Links

- clearlydefined.io · the project
- docs.clearlydefined.io · API, coordinates, curation guide
- api.clearlydefined.io · the API itself
- github.com/clearlydefined/curated-data · open curations
- github.com/jeffmendoza/cdsbom · enrich an SBOM in one command
- guac.sh · Graph for Understanding Artifact Composition

Back to that NOTICE file.

It still has hundreds of dependencies.
Now you have an answer for each one.

Thank you

Questions?

Jamie Magee · @JamieMagee.bsky.social