• Intel delivers *a lot* of software
• Software is a combination of own and FOSS components
• Many components have a legal source code distribution requirement
  • we also might deliver source in other cases
For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. — GPLv2
Complete Corresponding Source (CCS)

Different terms used

- GPLv2: "complete corresponding machine-readable source code" / "accompany"
- GPLv3: "Corresponding Source" / "convey"
- MPLv2: "Source Code Form" / "made available"
- EPLv2: "Source Code" / "made available"
The problem

In an ideal world

- Fool-proof processes in place
- Set it up once, always working

Practical considerations

- People change roles or leave
- Re-organizations happen
- Things get forgotten
Use cases

- Trying to build an internal service:
  - Our delivery contains our own FOSS \texttt{sw.tar.gz}
  - Our delivery contains \texttt{gcc-7.3}
  - Our delivery contains \texttt{gcc} snapshot of revision 257214
  - Our delivery contains \texttt{gcc-7.3} patched with \texttt{patches.tar.gz}
Functional requirements

We need to be able to:

- provide our own software package
- refer to a “well-known” FOSS component
  - with release version or unique revision
- combine the two
  - well-known component with own patches

Great Idea

- Can we outsource the fulfilment of these requirements?
The idea

Is it compliant?

GPL FAQ: Can I put the binaries on my Internet server and put the source on a different Internet site?

- [v3] Yes. Section 6(d) allows this. However, you must provide clear instructions people can follow to obtain the source, and you must take care to make sure that the source remains available for as long as you distribute the object code.

- [v2] The GPL says you must offer access to copy the source code “from the same place”; that is, next to the binaries. However, if you make arrangements with another site to keep the necessary source code available, and put a link or cross-reference to the source code next to the binaries, we think that qualifies as “from the same place”.

Alexios Zavras, Stefano Zacchioli
Outsourcing Source Code Distribution Requirements FOSDEM 2018
The idea

Is it compliant?

GPL FAQ: Can I put the binaries on my Internet server and put the source on a different Internet site?

- [v3] Yes. Section 6(d) allows this. However, you must provide clear instructions people can follow to obtain the source, and you must take care to make sure that the source remains available for as long as you distribute the object code.
- [v2] The GPL says you must offer access to copy the source code “from the same place”; that is, next to the binaries. However, if you make arrangements with another site to keep the necessary source code available, and put a link or cross-reference to the source code next to the binaries, we think that qualifies as “from the same place”.

Wouldn’t it be great if someone could fulfill our requirements?
Our mission

Collect, preserve and share the source code of all the software that is publicly available.

Past, present and future

Preserving the past, enhancing the present, preparing the future.
Our principles

- Open approach
- Open source
- Transparency
- In for the long haul
- Non profit
- Replication

Software Heritage
Our principles

Cultural Heritage

Industry

Research

Education

Software Heritage

Open approach
- open source
- transparency

In for the long haul
- non profit
- replication
Data flow

Distros

GitLab lister

GitHub lister

Debian lister

PyPi lister

Forge

Package repos

...
Current sources

- **live**: GitHub, Debian
- **one-off**: Gitorious, Google Code
- **WIP**: Bitbucket
Archive coverage

Current sources

- **live**: GitHub, Debian
- **one-off**: Gitorious, Google Code
- **WIP**: Bitbucket

150 TB blobs, 5 TB database (as a graph: 7 B nodes + 60 B edges)
Archive coverage

Current sources

- live: GitHub, Debian
- one-off: Gitorious, Google Code
- WIP: Bitbucket

150 TB blobs, 5 TB database (as a graph: 7 B nodes + 60 B edges)

The richest public source code archive, ... and growing daily!
Pushing source code to Software Heritage

Deposit service

- complement regular (pull) crawling of forges and distributions
- restricted access (i.e., not a warez dumpster!)
- deposit.softwareheritage.org

Tech bits

- **SWORD** 2.0 compliant server, for digital repositories interoperability
- RESTful API for deposit and monitoring, with CLI wrapper
Prepare a deposit

Prepare source code tarball

$ tar caf software.tar.gz /path/to/software/

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
      xmlns:codemeta="https://doi.org/10.5063/SCHEMA/CODEMETA-2.0">
  <title>Je suis GPL</title>
  <codemeta:author>
    <codemeta:name>Stefano Zacchiroli</codemeta:name>
    <codemeta:jobTitle>Maintainer</codemeta:jobTitle>
  </codemeta:author>
</entry>
Prepare a deposit

Prepare source code tarball

$ tar caf software.tar.gz /path/to/software/

Associate metadata

$ cat > software.tar.gz.metadata.xml
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
     xmlns:codemeta="https://doi.org/10.5063/SCHEMA/CODEMETA-2.0">
  <title>Je suis GPL</title>
  <codemeta:author>
    <codemeta:name>Stefano Zacchirole</codemeta:name>
    <codemeta:jobTitle>Maintainer</codemeta:jobTitle>
  </codemeta:author>
</entry>
^D
$ swh-deposit --username 'name' --password 'pass' --archive software.tar.gz
Send a deposit

```
$ swh-deposit --username 'name' --password 'pass' \
    --archive software.tar.gz

{
    'deposit_id': '11',
    'deposit_status': 'deposited',
    'deposit_date': 'Jan. 30, 2018, 9:37 a.m.'
}
```
Ingestion status

- Partial
- Deposited
- Verified
- Done
- Rejected
- Failed

$swh-deposit --username 'name' --pass 'secret' --deposit-id '11' --status

{
  'deposit_id': 11,
  'deposit_status': 'done',
  'deposit_status_detail': 'The deposit has been successfully loaded into the Software Heritage archive',
  'deposit_swh_id': 'swh:1:rev:a86747d201ab8f8657d145df4376676d5e47cf9f'
}
Ingestion status

$ swh-deposit --username 'name' --pass 'secret' \  
   --deposit-id '11' --status
Ingestion status

$ swh-deposit --username 'name' --pass 'secret' \ 
   --deposit-id '11' --status

{  
    'deposit_id': 11,
    'deposit_status': 'done',
    'deposit_status_detail': 'The deposit has been successfully loaded into the Software Heritage archive',
    'deposit_swh_id': 'swh:1:rev:a86747d201ab8f8657d145df4376676d5e47cf9f'
}
After ingestion a deposit becomes an integral, permanent part of the Software Heritage archive.

- it has a **persistent identifier**
  - e.g., swh:1:rev:a86747d201ab8f8657d145df4376676d5e47cf9f
- it can be **browsed online at** [archive.softwareheritage.org](http://archive.softwareheritage.org)
  - e.g., [https://archive.softwareheritage.org/browse/swh:1:rev:a86747d201ab8f8657d145df4376676d5e47cf9f](https://archive.softwareheritage.org/browse/swh:1:rev:a86747d201ab8f8657d145df4376676d5e47cf9f)
- it can be bulk **downloaded** using the Software Heritage Vault
Bulk download

- source code is thoroughly deduplicated within the Software Heritage archive
- bulk download of large artefacts (e.g., a Linux kernel release) requires collecting millions of objects
- the Software Heritage Vault cooks and caches source code bundles for bulk download needs
Bulk download

- source code is thoroughly deduplicated within the Software Heritage archive
- bulk download of large artefacts (e.g., a Linux kernel release) requires collecting millions of objects
- the Software Heritage Vault cooks and caches source code bundles for bulk download needs

```
$ curl -X POST /api/1/vault/revision/a86747d2.../gitfast
{
  'fetch_url': '/api/1/vault/revision/a86747d2.../gitfast/raw/',
  'progress_message': None,
  'status': 'new',
  'id': 4,
  'obj_id': 'a86747d201ab8f8657d145df4376676d5e47cf9f',
  'obj_type': 'revision_gitfast'
}
```

$ curl -O dump.gz /api/1/vault/revision/a86747d2.../gitfast/raw/
$ git init
$ zcat dump.gz | git fast-import
$ git checkout HEAD
Bulk download

- source code is thoroughly deduplicated within the Software Heritage archive
- bulk download of large artefacts (e.g., a Linux kernel release) requires collecting millions of objects
- the **Software Heritage Vault** cooks and caches source code bundles for bulk download needs

```
$ curl -X POST /api/1/vault/revision/a86747d2.../gitfast
{
    'fetch_url': '/api/1/vault/revision/a86747d2.../gitfast/raw/',
    'progress_message': None,
    'status': 'new',
    'id': 4,
    'obj_id': 'a86747d201ab8f8657d145df4376676d5e47cf9f',
    'obj_type': 'revision_gitfast'
}

$ curl -O dump.gz /api/1/vault/revision/a86747d2.../gitfast/raw/
$ git init
$ zcat dump.gz | git fast-import
$ git checkout HEAD
```
Wrapping up

- long-term hosting of CCS archives can be onerous in the real-world
- it is A-OK to outsource that responsibility to third parties
- Software Heritage crawls (pull) all FOSS and can now accept push deposits
- Intel and Software Heritage are working together on practical FOSS tooling to outsource CCS hosting to the Software Heritage archive

Come and join us!

- alexios.zavras@intel.com, zack@upsilon.cc
- https://www.softwareheritage.org
- https://deposit.softwareheritage.org
- https://archive.softwareheritage.org (FOSDEM 2018 preview!)