Beyond reproducible builds

We are not there yet and why
"just" achieving reproducible builds
won't be enough

Chris Lamb (lamby)
lamby@debian.org

Holger 'h01ger' Levsen
holger@debian.org

MiniDebConf 2015
Cambridge, UK

Debian reproducible builds team

akira
Andrew Ayer
Asheesh Laroia
Chris Lamb
Chris West
Christoph Berg
Daniel Kahn Gillmor
David Suarez
Dhole
Drew Fisher
Esa Peuha
Guillem Jover
Hans-Christoph Steiner
Helmut Grohne
Holger Levsen
Jelmer Vernooij
josch
Juan Picca
Lunar
Mathieu Bridon
Mattia Rizzolo
Nicolas Boulenguez
Niels Thykier
Niko Tyni
Paul Wise
Peter De Wachter
Philip Rinn
Reiner Herrmann
Stefano Rivera
Stéphane Glondu
Steven Chamberlain
Tom Fitzhenry
Valentin Lorentz
Wookey
Ximin Luo

Debian reproducible builds team

akira
Andrew Ayer
Asheesh Laroia
Chris Lamb
Chris West
Christoph Berg
Daniel Kahn Gillmor
David Suarez
Dhole
Drew Fisher
Esa Peuha
Guillem Jover
Hans-Christoph Steiner
Helmut Grohne
Holger Levsen
Jelmer Vernooij
josch
Juan Picca
Lunar
Mathieu Bridon
Mattia Rizzolo
Nicolas Boulenguez
Niels Thykier
Niko Tyni
Paul Wise
Peter De Wachter
Philip Rinn
Reiner Herrmann
Stefano Rivera
Stéphane Glondu
Steven Chamberlain
Tom Fitzhenry
Valentin Lorentz
Wookey
Ximin Luo

Who are you?

- Seen a talk about reproducible builds this year?

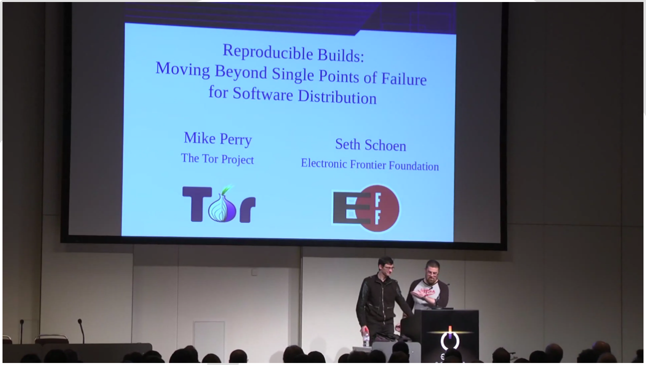
Who are you?

- Seen a talk about reproducible builds this year?
- Contributed to this effort?

Who are you?

- Seen a talk about reproducible builds this year?
- Contributed to this effort?
- Thinks ”packages **should** produce reproducible binaries” should be added to Policy?

The problem


Reproducible Builds:
Moving Beyond Single Points of Failure
for Software Distribution
Mike Perry
The Tor Project
Seth Schoen
Electronic Frontier Foundation

Available on `media.ccc.de`, 31c3

The solution

Promise that anyone can always generate identical binary packages from a given source

The solution

We call this:

“Reproducible builds”

Demo

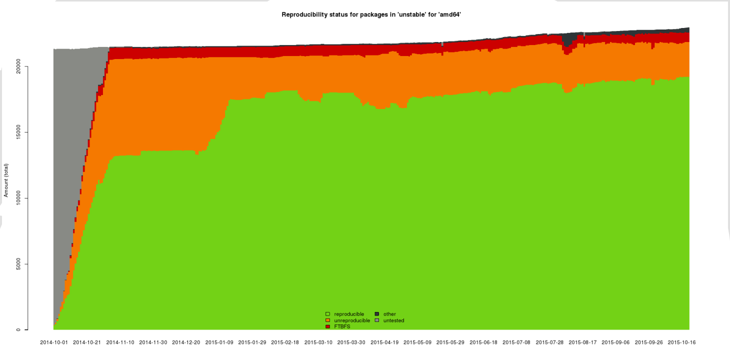
This should become the **norm**.

This should become the **norm**.

We want to change the meaning of ”free software”:

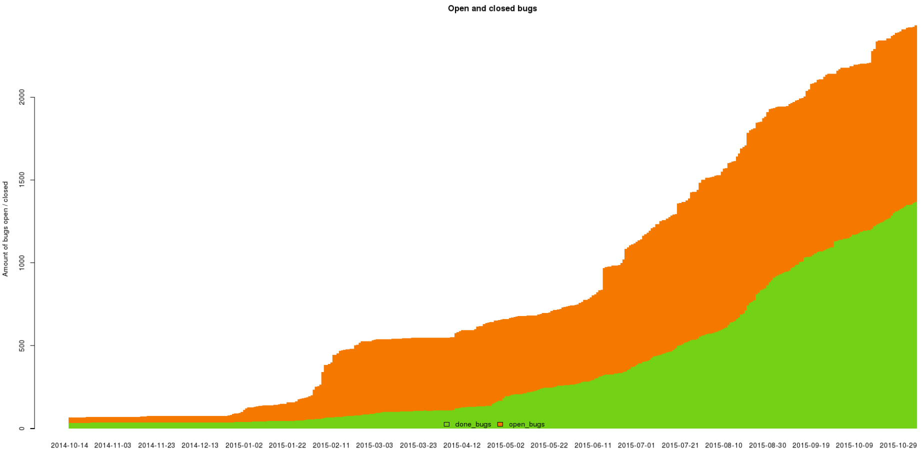
it’s only free software if it’s reproducible!

Progress in Debian `unstable`



19,500 out of 23,079 source packages are reproducible in our test framework

Progress in the Debian BTS



What we did since Summer 2014

- Agreed on a fixed build path: `/build`
- Recording the build environment: `.buildinfo`
- `strip-nondeterminism`
- `reproducible.debian.net`
- `diffoscope` (formerly `debbindiff`)
- `SOURCE_DATE_EPOCH`
- `disorderfs`
- 700+ patches: `dpkg`, `debhelper`, `sbuild`, ...

What we did since Summer 2014

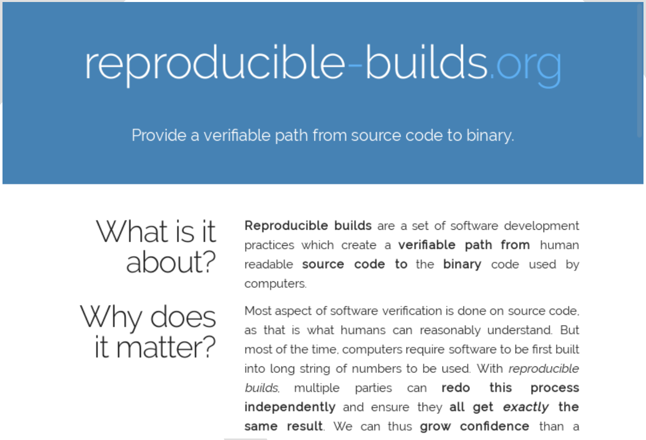
- Agreed on a fixed build path: `/build`
- Recording the build environment: `.buildinfo`
- `strip-nondeterminism`
- `reproducible.debian.net`
- `diffoscope` (formerly `debbindiff`)
- `SOURCE_DATE_EPOCH`
- `disorderfs`
- 700+ patches: `dpkg`, `debhelper`, `sbuild`, ...
- Tell the world & collaborate

Tell the world & collaborate

- Recent talks available with subtitles:
  - 2015-08-13: Chaos Communication Camp 2015
  - 2015-08-20: DebConf15
- Weekly reports since May 2015
- Summit in December 2015 (Athens)
  - 40 people from 16 projects

Tell the world & collaborate, cont.

`https://reproducible-builds.org`


reproducible-builds.org

Provide a verifiable path from source code to binary.

What is it about?

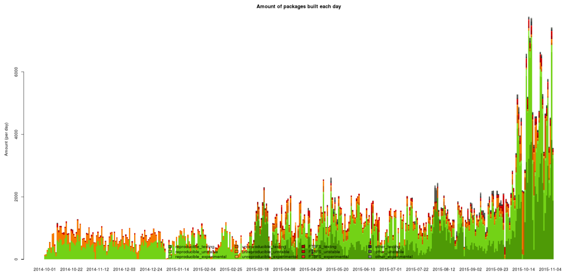
Reproducible builds are a set of software development practices which create a **verifiable path from** human readable **source code to** the **binary** code used by computers.

Why does it matter?

Most aspect of software verification is done on source code, as that is what humans can reasonably understand. But most of the time, computers require software to be first built into long string of numbers to be used. With *reproducible builds*, multiple parties can **redo this process independently** and ensure they **all get *exactly* the same result**. We can thus **grow confidence** than a

Stats about reproducible.debian.net

- Continuously testing Debian testing, unstable and experimental
  - `main` only
  - can we build contrib without legal troubles?
- Also testing coreboot, OpenWrt, NetBSD, FreeBSD, Archlinux and soon Fedora
  - those currently only weekly though...



More stats on reproducible.debian.net

- 111 jenkins jobs running on 10 hosts
- 27 contributors for `jenkins.debian.net.git`
- 4k lines of Python and 5k lines Bash code
- `amd64`: 109 cores and 194 GB RAM split on 8 VMs, provided by https://profitbricks.co.uk
- `armhf`: 12 cores and 6 GB RAM on 4 systems, provided by vagrant@d.o.

PROFITBRICKS
The IaaS-Company.

Good to know about reproducible.debian.net

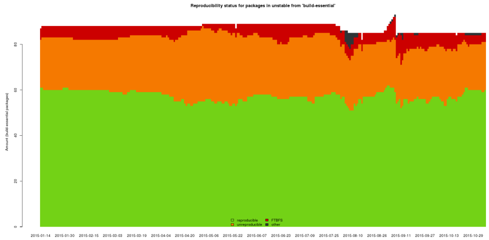
- https://reproducible.debian.net/$src

Good to know about reproducible.debian.net

- https://reproducible.debian.net/$src
- 165 categorised distinct issues
- 3,270 packages to be fixed, but only 249 without annotated issues

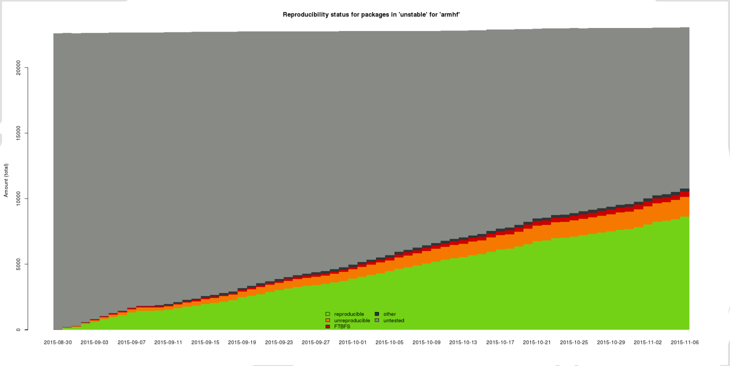
Good to know about reproducible.debian.net

- https://reproducible.debian.net/$src
- 165 categorised distinct issues
- 3,270 packages to be fixed, but only 249 without annotated issues
- 29 different "package sets", eg. build-essential is only <70% reproducible



Future of reproducible.debian.net

- We want more more more arm(64) cores!



Variations on reproducible.debian.net

variation	**first build**	**second build**
hostname	jenkins	i-capture-the-hostname
domainname	debian.net	i-capture-the-domainname
env TZ	GMT+12	GMT-14
env LANG	en_GB.UTF-8	fr_CH.UTF-8
env LC_ALL	not set	fr_CH.UTF-8
env USER	pbuilder1	pbuilder2
uid	1111	2222
gid	1111	2222
UTS namespace	shared with the host	*modified using /usr/bin/unshare --uts*
kernel version	Linux 3.16.0-4-amd64	Linux 2.6.56-4-amd64
umask	0022	0002
CPU type	same for both builds *(work in progress)*	
filesystem	same for both builds *(work in progress - disorderfs)*	
year, month, date	same for both builds *(work in progress)*	
hour, minute	hour is usually the same... usually, the minute differs... *(work in progress)*	
everything else	*is likely the same...*	

Debian .buildinfo

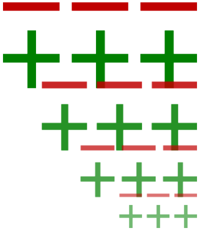
- Aggregates in the same file:
  - Sources (checksums)
  - Generated binaries (checksums)
  - Packages used to build (with specific version, checksums coming soon)
- Can be later used to exactly recreate environment
- For Debian, all versions are available from `snapshot.debian.org`

Example .buildinfo

```
Format: 1.9
Build-Architecture: amd64
Source: txtorcon
Binary: python-txtorcon
Architecture: all
Version: 0.11.0-1
Build-Path: /buildd/debian/txtorcon-0.11.0-1
Checksums-Sha256:
 a26549d9…7b 125910 python-txtorcon_0.11.0-1_all.deb
 28f6bcbe…69 2039 txtorcon_0.11.0-1.dsc
Build-Environment:
 base-files (= 8),
 base-passwd (= 3.5.37),
 bash (= 4.3-11+b1),
 …
```

Debugging problems: diffoscope

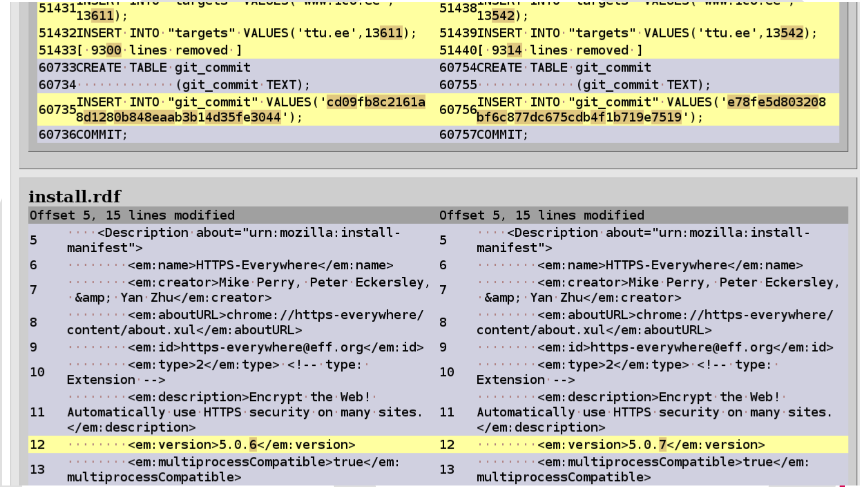
- Examines differences **in depth**
- Outputs HTML or plain text showing differences
- Recursively unpacks archives
- Seeks human readability:
  - uncompresses PDF
  - disassembles binaries
  - unpacks Gettext files
  - ... *easy to extend to new file formats*
- Falls back to binary comparison
- Available in Debian sid and stretch
- Maintainers in other distros wanted



`http://diffoscope.org/`

(formely known as `debbindiff`)

diffoscope example (HTML output)



```
51431                                            51438
     13611);                                          13542);
51432INSERT·INTO·"targets"·VALUES('ttu.ee',13611); 51439INSERT·INTO·"targets"·VALUES('ttu.ee',13542);
51433[·9300·lines·removed·]                      51440[·9314·lines·removed·]
60733CREATE·TABLE·git_commit                     60754CREATE·TABLE·git_commit
60734·············(git_commit·TEXT);             60755·············(git_commit·TEXT);
60735INSERT·INTO·"git_commit"·VALUES('cd09fb8c2161a  60756INSERT·INTO·"git_commit"·VALUES('e78fe5d803208
     8d1280b848eaab3b14d35fe3044');                    bf6c877dc675cdb4f1b719e7519');
60736COMMIT;                                      60757COMMIT;
```

install.rdf

```
Offset 5, 15 lines modified                      Offset 5, 15 lines modified
5   ····<Description·about="urn:mozilla:install-   5   ····<Description·about="urn:mozilla:install-
    manifest">                                         manifest">
6   ········<em:name>HTTPS-Everywhere</em:name>    6   ········<em:name>HTTPS-Everywhere</em:name>
7   ········<em:creator>Mike·Perry,·Peter·Eckersley,  7   ········<em:creator>Mike·Perry,·Peter·Eckersley,
    ·&amp;·Yan·Zhu</em:creator>                        ·&amp;·Yan·Zhu</em:creator>
8   ········<em:aboutURL>chrome://https-everywhere/  8   ········<em:aboutURL>chrome://https-everywhere/
    content/about.xul</em:aboutURL>                    content/about.xul</em:aboutURL>
9   ········<em:id>https-everywhere@eff.org</em:id>  9   ········<em:id>https-everywhere@eff.org</em:id>
10  ········<em:type>2</em:type>·<!--·type:·       10  ········<em:type>2</em:type>·<!--·type:·
    Extension·-->                                      Extension·-->
    ········<em:description>Encrypt·the·Web!·          ········<em:description>Encrypt·the·Web!·
11  Automatically·use·HTTPS·security·on·many·sites. 11  Automatically·use·HTTPS·security·on·many·sites.
    </em:description>                                  </em:description>
12  ········<em:version>5.0.6</em:version>         12  ········<em:version>5.0.7</em:version>
13  ········<em:multiprocessCompatible>true</em:   13  ········<em:multiprocessCompatible>true</em:
    multiprocessCompatible>                            multiprocessCompatible>
```

`diffoscope` is "just" for debugging

- Reminder: `diffoscope` is for **debugging**

`diffoscope` is ”just” for debugging

- Reminder: `diffoscope` is for **debugging**
- ”reproducible” according to our definition means: **bit by bit identical**. So the tools for testing whether something is reproducible are either `diff` or `sha256sum`!

SOURCE_DATE_EPOCH

- Build date usually not useful for the user
- Value of `SOURCE_DATE_EPOCH` instead of current date & for other seeds
- In Debian, set from the latest `debian/changelog` entry
- General solution for other projects & distributions

SOURCE_DATE_EPOCH (closed bugs)

- ~~#791823~~: debhelper
- ~~#787444~~: help2man
- ~~#790899~~: epydoc
- ~~#794004~~: ghostscript
- ~~#783475~~: texi2html
- ~~#794586~~: ocamldoc
- sphinx
  https://github.com/sphinx-doc/sphinx/pull/1954

SOURCE_DATE_EPOCH (open bugs)

- gcc (__DATE__ and __TIME__ macros)
  https://gcc.gnu.org/ml/gcc-patches/2015-06/msg02210.html
- #792687: gettext (xgettext)
- #792201: doxygen
- #800797: docbook-utils
- #790801: txt2man
- #791815: libxslt
- #794681: qt4-x11 (qthelpgenerator)
- #792202: texlive-bin

Missing bits

- NB. This is just a proof-of-concept, Debian is not 80% reproducible
- Changes still need to be merged

dpkg

- ~~#719844: make compression of {data,control}.tar.gz deterministic~~
- #759999: set reproducible timestamps in `.deb` ar file headers
- #787980: normalize file permissions when creating control.tar
- #719845: make file order within data,control.tar.gz deterministic
- `dpkg-genbuldinfo`: *patch already written, but waiting on agreement about spec*

debhelper

- #759886: make mtimes of packaged files deterministic
- ~~#759895: add a call to dh_strip_nondeterminism in dh~~
- ~~#791823: set SOURCE_DATE_EPOCH env var for reproducible builds~~

sbuild

- ~~#790868: allow sbuild to use a deterministic build path to build packages~~
- #778571: predictible build location for reproducible builds
- Finish the `srebuild` script

ftp.debian.org

- #763822: please include .buildinfo file in the archive

debian-policy

- Section 4.15: “Sources **must** build reproducible binaries.”

debian-policy

- Section 4.15: “Sources **must** build reproducible binaries.”
- We hope this will happen after stretch

debian-policy

- Section 4.15: “Sources **must** build reproducible binaries.”
- We hope this will happen after stretch
- (In 2016: “Sources **shall** build reproducible binaries.”)

Reproducible builds demand a defined build environment

- Re-creating an identical build environment is mandatory too.
- Without an identical build environment, reproducible builds will only happen by sheer luck.

Reproducible builds demand a defined build environment

- Re-creating an identical build environment is mandatory too.
- Without an identical build environment, reproducible builds will only happen by sheer luck.
- Only solved for Debian right now and currently proof of concept only...

Debian release process

- In our current design and practices, rebuilding stretch will require package versions which are not part of stretch.
- This design might put a high load on snapshot.debian.org.

Debian release process

- In our current design and practices, rebuilding stretch will require package versions which are not part of stretch.
- This design might put a high load on snapshot.debian.org.
- Rebuilding all of Debian a month prio the release? The release team probably won't like this.

Debian release process

- In our current design and practices, rebuilding stretch will require package versions which are not part of stretch.
- This design might put a high load on snapshot.debian.org.
- Rebuilding all of Debian a month prio the release? The release team probably won't like this.
- So? (Self contained reproducibility should be the goal...)

Distributing .buildinfo files

- Probably 100,000 new files per suite; 50% increase per suite
- Mirrors would not be happy, so should not go there
- We'll need more files when we have detached signatures

Distributing .buildinfo files

- Probably 100,000 new files per suite; 50% increase per suite
- Mirrors would not be happy, so should not go there
- We'll need more files when we have detached signatures
- Revoking signatures?

Distributing .buildinfo files

- Probably 100,000 new files per suite; 50% increase per suite
- Mirrors would not be happy, so should not go there
- We'll need more files when we have detached signatures
- Revoking signatures?
- ...

Rebuilders and sharing signed checksums

- Almost no work has been done here yet.

Rebuilders and sharing signed checksums

- Almost no work has been done here yet.
- Continuous rebuilds should happen in a systematic way and resulting checksums properly published.

Rebuilders and sharing signed checksums

- Almost no work has been done here yet.
- Continuous rebuilds should happen in a systematic way and resulting checksums properly published.
- And then we need a system to sign those checksums and share them.

Rebuilders and sharing signed checksums, cont.

- Individuelly signed checksums (think web of trust) could work in the Debian case (we have a gpg web of trust), but won't scale.

Rebuilders and sharing signed checksums, cont.

- Individuelly signed checksums (think web of trust) could work in the Debian case (we have a gpg web of trust), but won't scale.
- We'll probably need systematic rebuilders, run by large organisations (ACLU, NASA, NSA, Deutsche Bank, EDF, Greenpeace, XYZ).

Rebuilders and sharing signed checksums, cont.

- Individuelly signed checksums (think web of trust) could work in the Debian case (we have a gpg web of trust), but won't scale.
- We'll probably need systematic rebuilders, run by large organisations (ACLU, NASA, NSA, Deutsche Bank, EDF, Greenpeace, XYZ).
- ...and automated installers for those...

Rebuilders and sharing signed checksums, cont.

- Individuelly signed checksums (think web of trust) could work in the Debian case (we have a gpg web of trust), but won't scale.
- We'll probably need systematic rebuilders, run by large organisations (ACLU, NASA, NSA, Deutsche Bank, EDF, Greenpeace, XYZ).
- ...and automated installers for those...
- ...and howtos (`gpg --gen-key`)...

No more source only uploads?

- Should people be forced again to always do binary uploads, which only will be accepted when the checksum matches the one done by the buildds?

No more source only uploads?

- Should people be forced again to always do binary uploads, which only will be accepted when the checksum matches the one done by the buildds?
- Probably not.

No more source only uploads?

- Should people be forced again to always do binary uploads, which only will be accepted when the checksum matches the one done by the buildds?
- Probably not.
- Instead: keep checksums of uploaded binaries and rebuild anyway, and keep those checksums too.

Integration in user tools

- "Do you really want to install this unreproducible software (y/N)"

Integration in user tools

- ”Do you really want to install this unreproducible software (y/N)”
- ”Do you want to build those packages which unconfirmed checksums, before installing? (Y/n)”

Integration in user tools

- "Do you really want to install this unreproducible software (y/N)"
- "Do you want to build those packages which unconfirmed checksums, before installing? (Y/n)"
- "How many signed checksums do you require to call a package 'reproducible'?"

Integration in user tools

- "Do you really want to install this unreproducible software (y/N)"
- "Do you want to build those packages which unconfirmed checksums, before installing? (Y/n)"
- "How many signed checksums do you require to call a package 'reproducible'?"
- "Which rebuilders do you want to trust?"

Integration in user tools - conclusion

- ”Rebuilders and sharing signed checksums” needs to be designed (and probably at least partly implemented) before thinking more about end user tools. It’s just clear we need them.

As a developer

- Stop using build dates
- Use `SOURCE_DATE_EPOCH` instead
- See `https://reproducible-builds.org/specs/`

Get involved - learning by doing

- Test for yourself:
  - Build something twice, run diffoscope on the results
    - For better results use our “reproducible” repository, `pbuilder` and a custom config
- Docs on the wiki:
  `https://wiki.debian.org/ReproducibleBuilds/Howto`
  `https://wiki.debian.org/ReproducibleBuilds/ExperimentalToolchain`
- Ask for help on `#debian-reproducible` or on mailing list

Join the team!

- Why?
  - ♡♡♡ Lovely group of people ♡♡♡
  - Learn something new everyday
  - Change the (software) world!
- What do we do?
  - Review packages
  - Identify issues and document solutions
  - `reproducible.d.n`, diffoscope, strip-nondeterminism
  - Propose changes for toolchain
  - Submit patches for individual packages
  - Write more general documentation and talk to the world

Help migrating to `.debian.org` infrastructure

- `sudo pbuilder` doesn't make DSA happy
- Maintenance script really makes DSA unhappy. (`sudo kill -9 *`..)
- DSA would give us more build nodes of other architectures
- `jenkins.debian.org` migration

Questions, comments, ideas?

- `https://reproducible-builds.org`
- `https://reproducible.debian.net`
- `#debian-reproducible` on `irc.OFTC.net`

Thanks!

- Debian "Reproducible Builds" team
  (you are just **so** awesome!)
- Linux Foundation and the Core Infrastructure Initiative
- MiniDebConf Cambridge 2015

LINUX FOUNDATION

CORE INFRASTRUCTURE INITIATIVE

```
holger@debian.org  B8BF 5413 7B09 D35C F026
                   FE9D 091A B856 069A AA1C
 lamby@debian.org  C2FE 4BD2 71C1 39B8 6C53
                   3E46 1E95 3E27 D431 1E58
```