

Reproducible builds

of openSUSE Factory and SLE

Bernhard M. Wiedemann
Software Developer +
Sysadmin
bmwiedemann@suse.de

Introduction

The background features a large teal shape on the left and a green shape on the right, separated by a white diagonal line. The teal shape is a large, irregular polygon with a pointed top and a pointed bottom. The green shape is a large, irregular polygon with a pointed top and a pointed bottom, mirroring the teal shape's form. The white diagonal line runs from the top right to the bottom left, creating a sense of movement and division.

About me

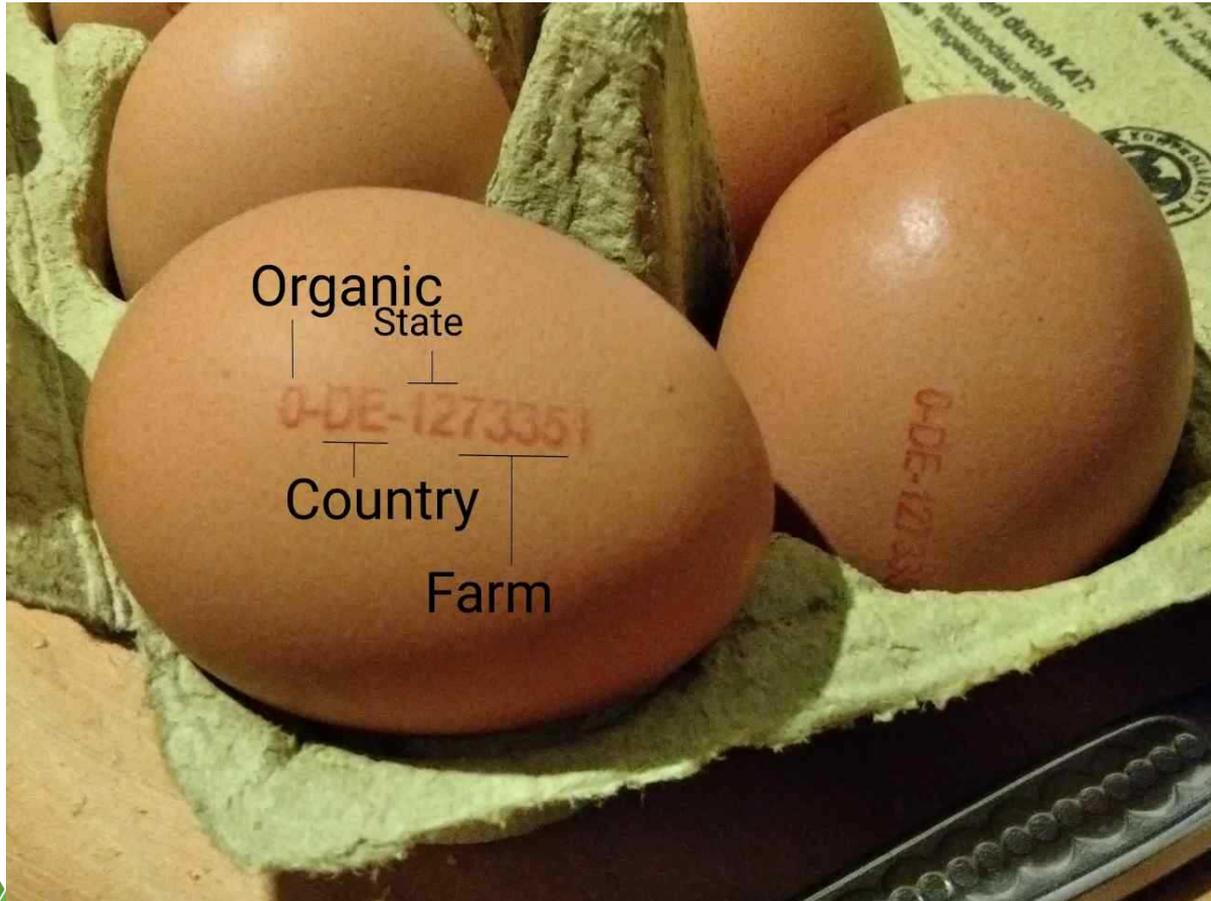
- working since 2010 for SUSE
- active since 2016 in reproducible builds
- interested in IT security for decades



The Problem

The background features a large teal shape on the left and a green shape on the right, separated by a white diagonal line. The teal shape is a large, irregular polygon with a pointed top and a pointed bottom. The green shape is a large, irregular polygon with a pointed top and a pointed bottom, mirroring the teal shape's orientation. The white diagonal line runs from the top right towards the bottom left, creating a clear division between the two colored areas.

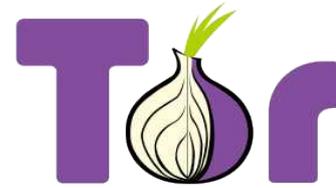
Why do we stamp out eggs?



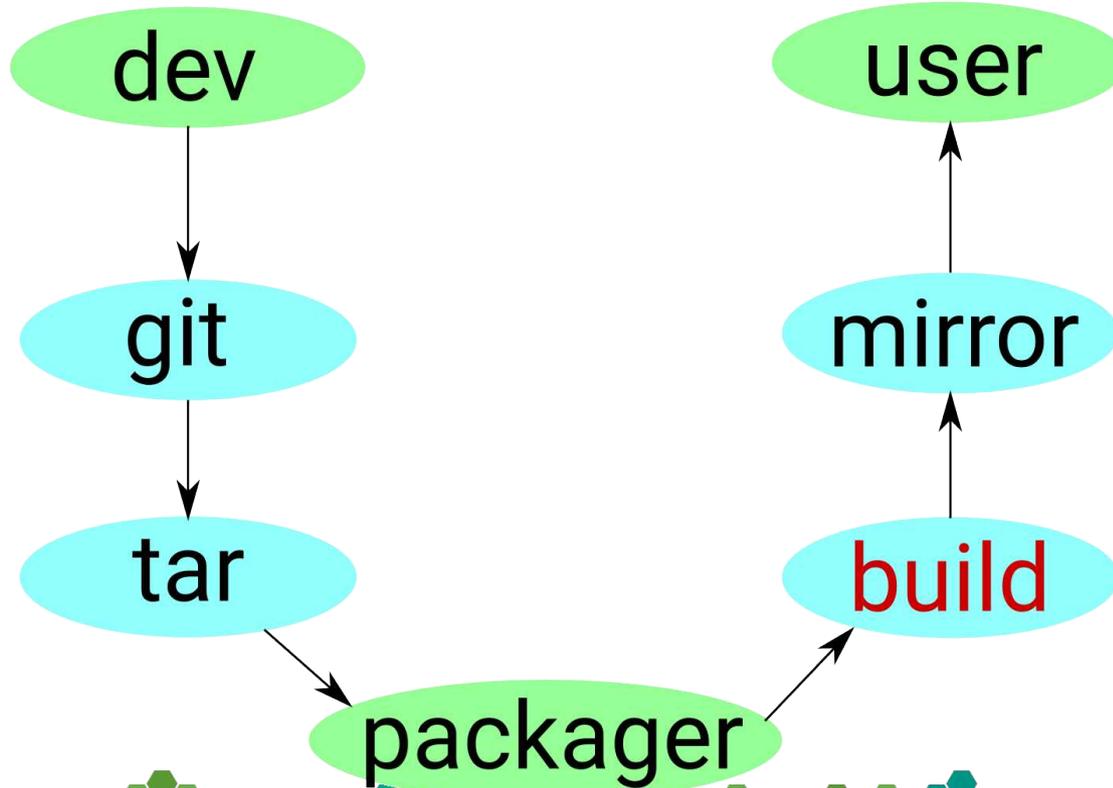
Where our eggs come from



Similar problem with software

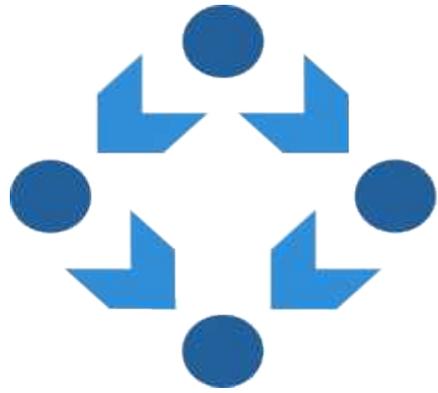


Where does our code come from



The background features a large teal shape on the left and a green shape on the right, separated by a white diagonal line. The teal shape is a large, irregular polygon with a pointed right side. The green shape is a large, irregular polygon with a pointed left side, mirroring the teal shape's form. The white line runs diagonally from the top right to the bottom left, creating a sense of movement and division.

The solution



Reproducible Builds

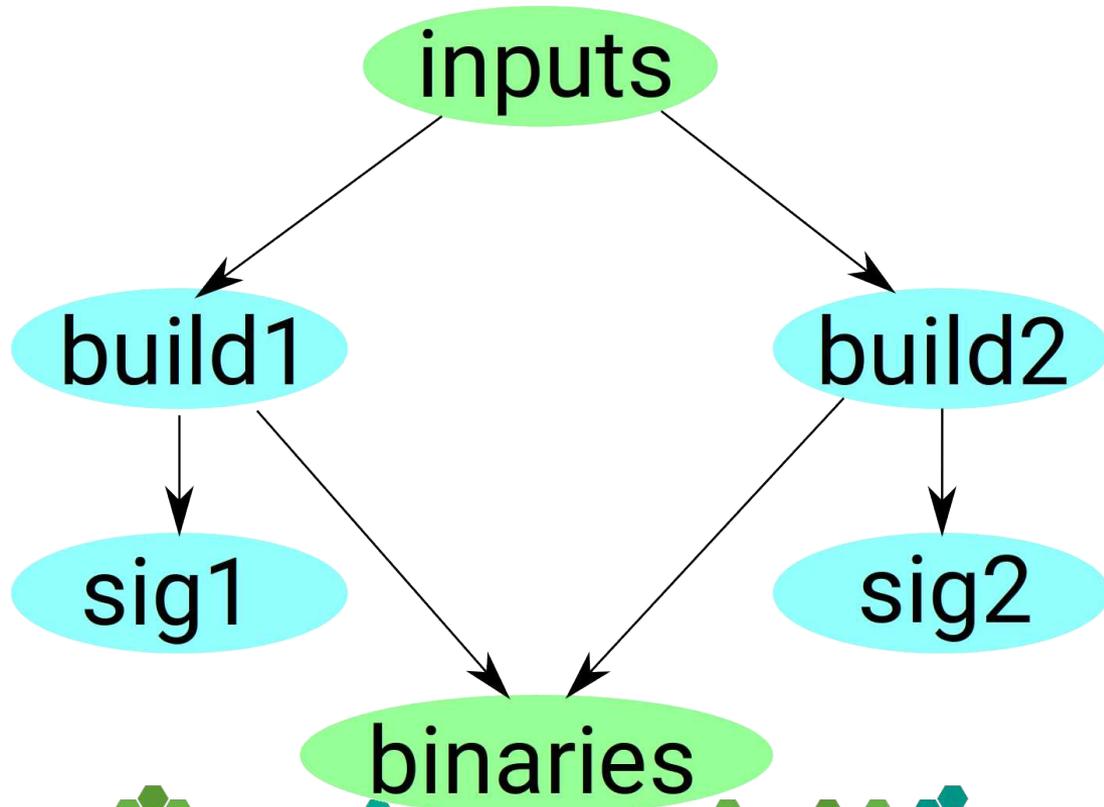


What are reproducible builds?

- Get the same results from building sources twice
- Two use-cases
 - ideally bit-by-bit identical (thus same hashes)
 - weaker: same content after applying some filters (via build-compare)



RB diagram



Why reproducible builds?

- Need less trust in the build hosts
- Reduced load on build-service from rebuilds
- Smaller delta-rpms in update repos
- Save bandwidth (by building locally)
- Find other bugs that corrupt data during build time (e.g. [boo#1021353](#), [boo#1021335](#), [bash](#), [libcbor](#))



Typical problems

- embedded timestamps, hostname
- embedded rebuild counters
- random .o file link order changes optimization
- compile-time CPU detection



new sources of randomness discovered

- gcc profile-guided optimizations
 - can be fixed by **always doing the same** in the profiling run
 - or by **removing differing .gcda files** losing some of the optimizations, but not all
- `%ghost` files have (semi-random) sizes visible in `rpm -qp --dump`
- **ASLR**



new sources of randomness discovered #2

- unsorted globs in python, bam, boost/jam
 - `glob.glob("*.*") => sorted(...)`
 - jam see <https://github.com/boostorg/container/pull/50>



The background features abstract geometric shapes in two shades of green. On the left, a large teal shape with a pointed right side contains the text. To its right, a white diagonal band separates it from a bright green shape on the far right. The overall composition is clean and modern.

Current state

Work done

- 2016: 71 submit-requests ; 6 bugs filed
- 2017: +92 submit-requests ; 4 bugs filed
- 2018: +71 submit-requests ; 13 bugs filed
- 2019: +133 submit-requests ; 28 bugs filed
- 2020: +42 submit-requests ; 24 bugs filed
- 2021: +31 submit-requests ; 23 bugs filed



Work done #2

- 2016: 4 upstream fixes merged
- 2017: +51 upstream fixes submitted - ~34 merged
- 2018: +270 upstream submissions - 162 merged
- 2019: +170 upstream submissions - 120 merged
- 2020: +130 upstream submissions - 80 merged
- 2021: +49 upstream submissions - 34 merged



rebuild-test-scripts

- available from
<https://github.com/bmwiedemann/reproducibleopensuse>
- including this presentation's source
<https://github.com/bmwiedemann/reproducibleopensuse/blob/presentation/presentation/reproducible.md>



How reproducible can we get?

- bit-identical with factory rpm and `osc build --define='%_buildhost reproducible' --define='%clamp_mtime_to_source_date_epoch Y' --define='%use_source_date_epoch_as_buildtime Y'`



Why does not everyone do reproducible builds yet

- Performance is more important
- Details about build are more important
- Generating UUIDs via random is easier
- Digital signatures contain time and randomness



No panacea

- backdoors in source
- buffer overflows and other bugs
- bad crypto
- volkswagen testing-mode



Package managers

- can fetch and verify third party rebuild certifications



Where do we want to go?

- fix all build-compare issues
- produce bit-identical rpms
- continuously verify published binaries
- report reproducibility regressions in submit-requests



Bash

From: Bernhard M. Wiedemann
Subject: [PATCH] Use memmove instead of strcpy
Date: Sat, 7 Jul 2018 07:25:52 +0200

In https://bugzilla.opensuse.org/show_bug.cgi?id=1100488
we found that depending on the build machine, bash-4.4's bash.html would
contain the string Bahh instead of Bash

strcpy can cause corruption when working on overlapping strings
so we use memmove instead that handles this case correctly

```
---  
support/man2html.c | 2 +-  
1 file changed, 1 insertion(+), 1 deletion(-)  
  
diff --git a/support/man2html.c b/support/man2html.c  
index 6ba50616..f56a8945 100644  
--- a/support/man2html.c  
+++ b/support/man2html.c  
@@ -1992,7 +1992,7 @@ unescape (char *c)  
    while (i < l && c[i]) {  
        if (c[i] == '\a') {  
            if (c[i+1])  
-                strcpy(c + i, c + i + 1);        /* should be memmove */  
+                memmove(c + i, c + i + 1, strlen(c + i));  
        }  
    }
```



foma

<https://github.com/mhulden/foma/pull/78>

Fix crash on 32-bit systems

https://bugzilla.opensuse.org/show_bug.cgi?id=1109949

On 32-bit systems like i586, the sizeof a pointer is 4 so too little memory was allocated for the following loop

```
diff --git a/foma/rewrite.c b/foma/rewrite.c
index 8ac140a..f1b2073 100644
--- a/foma/rewrite.c
+++ b/foma/rewrite.c
@@ -103,7 +103,7 @@ struct fsm *fsm_rewrite(struct rewrite_set *all_rules) {
     rb = xxcalloc(1, sizeof(struct rewrite_batch));
     rb->rewrite_set = all_rules;
     rb->num_rules = num_rules;
-    rb->namestrings = xxmalloc(sizeof rb->namestrings * num_rules);
+    rb->namestrings = xxmalloc(sizeof *rb->namestrings * num_rules);
     for (i = 0; i < rb->num_rules; i++) { sprintf(rb->namestrings[i], "@#%04i@", i+1); }
```



dpdk

<http://patches.dpdk.org/patch/29949/>

```
--- a/mk/rte.sdkdoc.mk
+++ b/mk/rte.sdkdoc.mk
@@ -93,7 +93,7 @@ $(API_EXAMPLES): api-html-clean
     $(Q)mkdir -p $(@D)
     @printf '/**\n' > $(API_EXAMPLES)
     @printf '@page examples DPDK Example Programs\n\n' >> $
(API_EXAMPLES)
-   @find examples -type f -name '*.c' -printf '@example %p\n' >> $
(API_EXAMPLES)
+   @find examples -type f -name '*.c' | LC_ALL=C sort | xargs -l
echo "@example" >> $(API_EXAMPLES)
     @printf '*/\n' >> $(API_EXAMPLES)
```



python sort

<https://github.com/skyjake/Doomsday-Engine/pull/18>

Sort input file list
so that doomsday.pk3 builds in a reproducible way
in spite of indeterministic filesystem readdir order.

```
--- a/doomsday/build/scripts/packres.py
+++ b/doomsday/build/scripts/packres.py
def process_dir(path, dest_path):
    self.msg("processing %s" % os.path.normpath(path))
-    for file in os.listdir(path):
+    for file in sorted(os.listdir(path)):
    real_file = os.path.join(path, file)
```





Join Us at www.opensuse.org



License

This slide deck is licensed under the Creative Commons Attribution-ShareAlike 4.0 International license. It can be shared and adapted for any purpose (even commercially) as long as Attribution is given and any derivative work is distributed under the same license.

Details can be found at <https://creativecommons.org/licenses/by-sa/4.0/>

General Disclaimer

This document is not to be construed as a promise by any participating organisation to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. openSUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for openSUSE products remains at the sole discretion of openSUSE. Further, openSUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All openSUSE marks referenced in this presentation are trademarks or registered trademarks of SUSE LLC, in the United States and other countries. All third-party trademarks are the property of their respective owners.

Credits

Template

Richard Brown
rbrown@opensuse.org

Design & Inspiration

openSUSE Design Team
<http://opensuse.github.io/branding-guidelines/>