What's new in sbuild/trixie

Johannes Schauer Marin Rodrigues <josch@debian.org>
Jochen Sprickerhof <jspricke@debian.org>
Chris Hofstaedtler <zeha@debian.org>

DebConf Brest 2025

https://deb.li/buTG
`git clone https://salsa.d.o/jspricke/sbuild-talk`

Agenda

- Overview of `sbuild` and whats new
- How you can use `sbuild`
- Helping you to use `sbuild`

About us

josch (Johannes Schauer Marin Rodrigues) <josch@debian.org>

- maintaining sbuild since 2015
- added the **unshare** backend in 2018 for Debian 10 “Buster”
- not here with us in Brest today

Jochen (Sprickerhof) <jspricke@debian.org>

- maintaining sbuild since 2022
- pushed **unshare** backend adoption on the buildd

Chris (Hofstaedtler) <zeha@debian.org>

What is sbuild?

sbuild(1) is Debian's build manager

- Builds packages in a clean and minimal environment using `dpkg-buildpackage`
- Creates and manages chroot environments with only Essential, `build-essential` & `apt`

What's new?

- sbuild works out of the box*

  the **unshare** backend

- **unshare** backend used on the buildd network since 2024
- Try it on Trixie or bookworm-backports:

```
$ sudo apt install sbuild mmdebstrap uidmap
$ sbuild --chroot-mode=unshare --dist=unstable hello
```

- Don't worry, `schroot` setups will continue to work in Trixie

*with `--chroot-mode=unshare`

Why do we want the **unshare** backend?

- No need for root*
- Zero configuration needed
- Automatic chroot management using `mmdebstrap`
- No network for `dpkg-buildpackage`
- No cruft left behind, including mountpoints
- Stricter definition of the build environment, example: GCC #1089007

Validated by:

- debrebuild & reproduce.debian.net
- archive rebuilds
- buildd network

*except suid uidmap

How to use the **unshare** backend?

- `sbuild --chroot-mode=unshare`

Our Recommendation:

- remove old `~/.sbuildrc`

```
$ apt install sbuild mmdebstrap uidmap
$ rm .sbuildrc
$ echo "\$chroot_mode = 'unshare';" > ~/.config/sbuild/config.pl
```

- In case your user does not have subids yet:

```
$ sudo usermod --add-subuids 100000-165535 \
               --add-subgids 100000-165535 <user>
```

How did we get here?

- in 2015 josch turns Helmut's Python scripts into Perl and calls it user-unshare
- in June 2018 josch embeds that script into a new sbuild backend – runs in a new perl interpreter, surrounded by an embedded POSIX shell script
- in September 2018 josch turns the script into another tool: `mmdebstrap`
- in April 2022 jspricke starts hacking on the unshare backend
- in April 2024 the wanna-build team switches some buildds to unshare
- in April 2024 Santiago Vila and jspricke start regular trixie and unstable rebuilds with `sbuild --chroot-mode=unshare` and file bugs
- in August 2024 josch refactors the sbuild **unshare** backend, making it pretty
- in October 2024 the wanna-build team switches all Debian buildds to **unshare**

Where are we going?

- Empty `apt` cache – Bugs: user: debian-qa@lists.debian.org tag: apt-cache
- Support on porterboxes – RT#9664
- Support on salsa-ci – salsa-ci!569
- Switch to `autopkgtest-virt-auto` – autopkgtest!515
- No longer install `apt` by default
- Wrap unshare chroot in QEMU
- Reduce configuration boilerplate, better manual pages
- `unschroot` – see Helmut Grohne's talk! - next slot

How to debug failed builds?

Normal invocation:

```
$ sbuild --chroot-mode=unshare --dist=unstable \
  --run-piuparts \
  --run-autopkgtest \
  hello
```

Add a debug shell:

```
$ sbuild --chroot-mode=unshare --dist=unstable \
  --run-piuparts \
  --run-autopkgtest \
  --anything-failed-commands=%s \                    <-- add this!
  hello
```

Not enough space

unshare backend builds in `TMPDIR` – default: `/tmp`.

`/tmp` defaults to `tmpfs`, 50% of RAM size.
Size can be increase beyond RAM – by using swap!

Pick different build location – set `$unshare_tmpdir_template`.
Suggestion: use `/var/tmp`.

Or use any other world writable directory.
All path components leading to it must be world `+x`.

Cleaning the source

```
dpkg-checkbuilddeps: error: unmet build dependencies: libfoo-dev
dpkg-buildpackage: error: build dependencies/conflicts unsatisfied; aborting
E: Failed to clean source directory
```

The *input* to `sbuild` is a source package: `dsc` & friends.
If called *inside* an unpacked source tree, builds `dsc` using `dpkg-source -b .` – to do that, the unpacked source must be **clean**. To run `debian/rules clean`, `Build-Depends` must be installed – required by policy.

- Skip the `clean` target: use `--no-clean-source` or `$clean_source=0;`
  - `gbp buildpackage` enforces a clean git tree - could automatically set this
- Move build dependencies from `Build-Depends` to `Build-Depends-Indep` and `-Arch`.
  - Many packages only need `debhelper` in their `Build-Depends`
  - `Build-Depends-Arch`/`-Indep` not required by `clean` target
- Create the `dsc` yourself and pass it to `sbuild`

Discussion in bug #1088269 - currently *wontfix*

How sbuild users configure sbuild: zeha

```
$external_commands = {"build-failed-commands" => [ [ '%SBUILD_SHELL' ] ]};
$run_autopkgtest = 1;

$chroot_mode = 'unshare';
$unshare_tmpdir_template = '/var/tmp/tmp.sbuild.XXXXXXXXXX';
$unshare_mmdebstrap_keep_tarball = 1;
$unshare_mmdebstrap_max_age = 680400;
push @{$unshare_mmdebstrap_distro_mangle}, qr/^grml-.*$/ => 'unstable';
push @{$unshare_mmdebstrap_extra_args}, "*",
  ['--aptopt=Acquire::http { Proxy "http://127.0.0.1:3142"; }'];
```

How core sbuild developers configure sbuild: jspricke

```
$clean_source = 0;

$environment_filter = [];

$external_commands = {"build-failed-commands" => [ [ '%SBUILD_SHELL' ] ]};

$lintian_opts = ['-EvIL', '+pedantic', '-i'];

$run_piuparts = 1;
$run_autopkgtest = 1;
```

How to add apt-cacher-ng

```
my @common_mmdebstrap_args = (
  '--aptopt=Acquire::http { Proxy "http://127.0.0.1:3142"; }',
);
push @{$unshare_mmdebstrap_extra_args}, "*", \@common_mmdebstrap_args;

use String::ShellQuote 'shell_quote';
my $str_common_mmdebstrap_args = shell_quote @common_mmdebstrap_args;

$piuparts_opts = ["--no-eatmydata", "--distribution=%r", '--arch', '%a', \
  '--bootstrapcmd=mmdebstrap ' . $str_common_mmdebstrap_args];
```

How to add ccache

```
$dsc_dir = "package";

my @ccache_mmdebstrap_args = (
  '--include=ccache',
  '--chrooted-customize-hook=update-ccache-symlinks',
);
push @{$unshare_mmdebstrap_extra_args}, "*", \@ccache_mmdebstrap_args;

$build_environment = {'CCACHE_DIR' => '/build/ccache', \
 'CCACHE_UMASK' => '002'};
$path = '/usr/lib/ccache:/usr/sbin:/usr/bin:/usr/games';
$unshare_bind_mounts = [{directory => '/tmp/ccache', \
 mountpoint => '/build/ccache'}];
```

How to add a ccache-ed autopkgtest

```
my $str_ccache_mmdebstrap_args = shell_quote @ccache_mmdebstrap_args;

$autopkgtest_opts = ['--shell-fail', '--env=CCACHE_DIR=/build/ccache', \
 '--env=PATH=/usr/lib/ccache:/usr/sbin:/usr/bin:/sbin:/bin', '--', \
 'unshare', '--release', '%r', '--arch', '%a', \
 '--bind', '/tmp/ccache', '/build/ccache',  \
 '--bootstrapcmd=mmdebstrap ' . $str_common_mmdebstrap_args \
 . " " . $str_ccache_mmdebstrap_args];
```

How to add extra packages?

- `--enable-network`
- `--extra-repository "deb http://..." --extra-repository-key`
- `--extra-package` - this can be a directory!

How to cross build?

- `--arch arm64` - build with qemu
- `--host arm64` - cross build

How to work on a package?

- `--profiles=nocheck`
- edit the chroot from outside in `/tmp/tmp.sbuild.*`
  - unshare –map-root-user –map-auto
- run screen inside `--commands` hooks (`tmux` does not work)

Who needs help?

We want you to switch over to **unshare**.

We want *your* special cases.

We want to help you switch over to **unshare**.

Now? - Find us at this DebConf!