.Deb

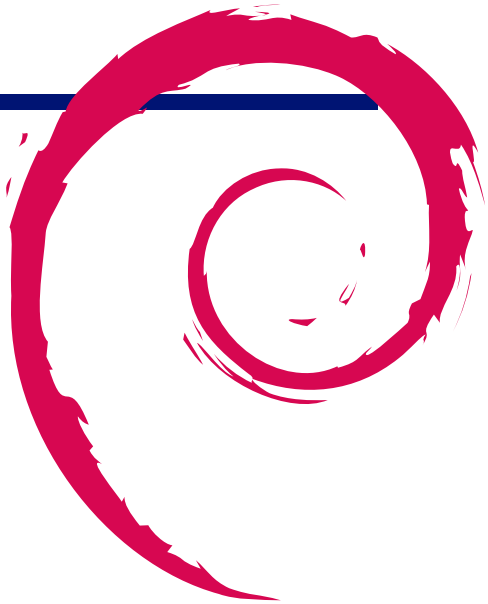
銀河系唯一のDebian専門誌

2017 年 6 月 18 日

Debian 9 リリース直後回！

デビアン勉強会

目次



1 最近の Debian 関連のミーティング報告

杉本 典充

1.1 第 151 回東京エリア Debian 勉強会

2017 年 5 月 20 日 (土) に第 151 回東京エリア Debian 勉強会を開催しました。会場は東銀座にある朝日ネットさんをお借りして行いました。参加者は 6 名でした。

発表は、kenhys さんによる「Debbugs とのつきあいかた:SOAP 編」でした。

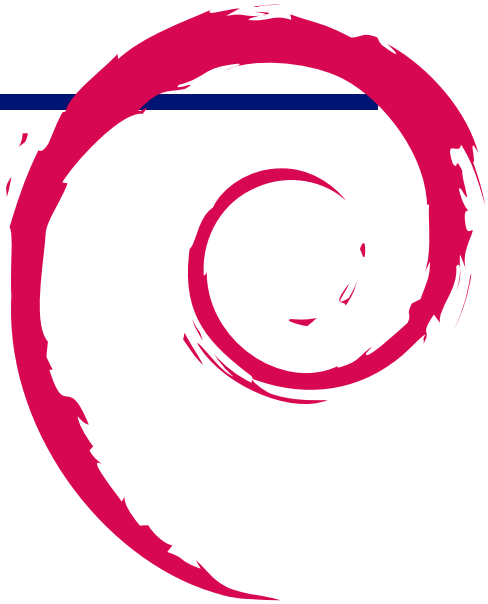
Debbugs とは Debian Bug Tracking System (以降 BTS と記述) で利用されているシステムのことです。BTS の参照や投稿メールの草稿作成は GUI フロントエンドをもつ reportbug-ng コマンドの利用者が多いです。本発表では、kenhys さんが利用しているチャットボットから Debbugs を操作するために SOAP インタフェース利用してみた報告です。[*1]。

Debbugs の SOAP インタフェースを利用するには、WSDL[*2]のインタフェース定義を入手する必要がありますが Debian Wiki に記載がなく見つけるには苦労したとのことです[*3]。そして ruby で Debubugs を操作する SOAP クライアントを実装し、Debbugs との通信に成功したとのことでした。

[*1] https://wiki.debian.org/DebbugsSoapInterface

[*2] Web Services Description Language のこと。

[*3] WSDL ファイルは `https://elpa.gnu.org/packages/debbugs.html` で発見できたとのことです。



2 事前課題

杉本 典充

今回の事前課題は以下です:

1. Debian9 “Stretch” リリースに際して一言！
2. Debian9 “Stretch” に期待することは？

この課題に対して提出いただいた内容は以下です。

2.1 yamaneco

1. （回答なし）
2. （回答なし）

2.2 Charles Plessy

1. Reproducible, continuously checked, rock solid more than ever.
2. I hope there will be good updates for hardware support, because hibernation on my laptop still does not work :(

2.3 Hiroshi Miura

1. （回答なし）
2. （回答なし）

2.4 yy_y_ja_jp

1. （回答なし）
2. （回答なし）

2.5 Marc Dequènes (Duck)

1. （回答なし）
2. （回答なし）

2.6 Yusaku

1. おめでとうございます！
2. （回答なし）

2.7 Haruki TSURUMOTO(tSU_RooT)

1. 次のリリースではもっと多くの (有用な) パッケージをいれてみたいです
2. reproducible-builds のサポート

2.8 MichelDaenzer

1. （回答なし）
2. （回答なし）

2.9 akedon

1. おめでとうございます。
2. （回答なし）

2.10 karosuwindam

1. （回答なし）
2. （回答なし）

2.11 大神祐真

1. DM、DD の皆様はお疲れ様です。おめでとうございます！
2. Linux-4.9 は最も大きな変更と言われていたので、カーネル回りの新機能も試してみたいです

2.12 koedoyoshida

1. めでたい。リリースお疲れ様です。
2. もう使ってます。

2.13 NOKUBI Takatsugu

1. はやく wheezy マシン上げなきゃ…
2. 安定していればそれで満足です

2.14 Roger Shimizu

1. GNU/screen が対応された Debian Installer を使って見てください
2. （回答なし）

2.15 bringer1092

1. おめでとうございます
2. （回答なし）

2.16 dictoss

1. 新しい debian、いやめでたい！
2. 多くの人に使ってもらえるような OS であってほしいです。

2.17 kenhys

1. （回答なし）
2. （回答なし）

2.18 Betsuyaku

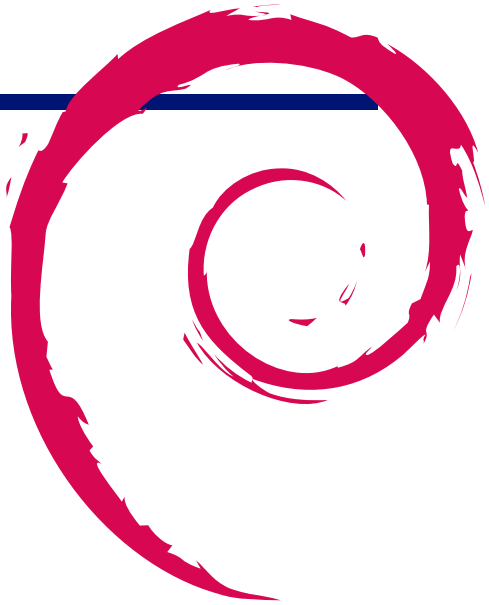
1. ibus さんの変貌に戸惑い
2. KVM まわりの進化

2.19 ysaito

1. （回答なし）
2. （回答なし）

2.20 henrich

1. （回答なし）
2. （回答なし）



3 Debian Trivia Quiz

杉本 典充

Debian の昨今の話題についての Quiz です。

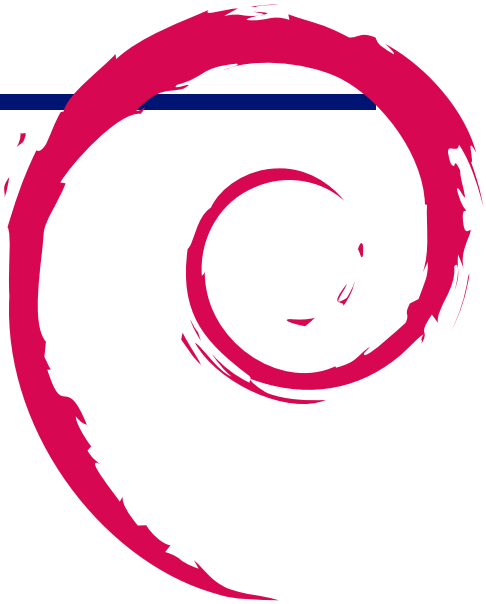
今回の出題範囲は `debian-devel-announce@lists.debian.org` や `debian-news@lists.debian.org` などに投稿された内容からです。

問題 1. ついにリリースされた Debian 9 (Stretch)。リリースノートに書かれているリリースされたパッケージ数はおよそどのくらいでしょうか。

☐ A 40,000 パッケージくらい
☐ B 50,000 パッケージくらい
☐ C 60,000 パッケージくらい

問題 2. まだ話は早いですが、これからは次の Debian 10 に向かって作業を進めることになります。Debian 10 のコードネームはなんでしょうか。

☐ A Potato
☐ B Bullseye
☐ C Buster



4 Debian Continuous Integration

Charles Plessy

4.1 Introduction

Debian Continuous Integration (debci) is both a running service (`https://ci.debian.net/`) and a set of packages powering the service, (`https://packages.debian.org/sid/debci`). The website is mobile-friendly, and cross-links with the Debian Package Tracker.

Debian runs debci on `unstable`. I want to run Debci on a local version of `Stretch` because a) a late update of `r-base` unexpectedly broke API and b) a further upload of `r-base` in *unstable* means that `ci.debian.net` does not report a situation reflecting the state of Squeeze.

(Very embarrassingly, I have to confess during this release party... some packages might still be broken at the moment...)

First, let's see what is Debci and how it works.

4.2 Regression tests at build time

Traditionally, in Debian regression tests have been running at build time.

Advantage: they run on all architectures.

Limitations:

- They run only once in the lifetime of the package.
- Their output is deep in the build logs.
- For a long time, they did not run for architecture-independent packages, because they were not autobuild. (This changed recently with the possibility of source-only uploads).
- Obviously, they require to build the package first, which may take time and require extra packages (for instance to build the documentation, etc...).

What if a binary package gets broken by the update of on of its dependencies ?

4.3 Continuous Integration

Advantages:

- Ran again and again each time a package's environment changes.
- No need to build from source.
- Opportunity to test features not available at build time (network, reboot, ...)

Limitations:

- At the moment, Debian only runs them on `amd64` and `arm64`.

Debian Continuous Integration uses the Autopkgtest framework.

4.4 Autopkgtest

Also known as *DEP 8* (`http://dep.debian.net/deps/dep8/`). In brief:

The presence of a test suite is announced by the line `Testsuite: autopkgtest` in the Debian Source Control file.

Regression tests are described in `debian/tests/control` in the *paragraph* ("pseudo-RFC-822") format well known to Debian packagers. The description is mostly about where to find the tests, and what they need to run.

When then are already run-time regression tests available upstream, populating `debian/tests` is easy. Example of the package `r-bioc-s4vectors`:

```
[r-bioc-s4vectors-0.12.1]$ cat debian/tests/control
Tests: run-unit-test
Depends: @, r-cran-runit, r-bioc-iranges (>= 2.0.0), r-bioc-genomicranges
Restrictions: allow-stderr

[r-bioc-s4vectors-0.12.1]$ cat debian/tests/run-unit-test
#!/bin/sh -e

LC_ALL=C R --no-save <<EOT
require("S4Vectors")
S4Vectors:::.test()
EOT
```

There are at least two ways to run Autopkgtest suites.

- For occasional and simple use `sadt`, from the `devscripts` package.
- For heavier or more complex use, `autopkgtest`, from the `autopkgtest` package. (Formerly, the command was called `adt-run`). One of the main advantages over `sadt` is the isolation of the test environment using containers or virtualisation.

4.5 Containerization of the test bed

`autopkgtest` supports multiple virtualisation systems:

- schroot
- LXC
- QEMU
- ssh (for instance to connect to a disposable cloud instance).

There is also the `null` system for running tests directly on the machine, like with `sadt`.

I tried `schroot`, because I am most familiar with it as it is used in the Debian build farm, and I use `sbuild` to build my packages. I created a schroot with the command `sbuild-createchroot`, and modified the

configuration according to `https://ci.debian.net/doc/file.MAINTAINERS.html`

```
$ cat /etc/schroot/chroot.d/debci-stretch-amd64
[debci-stretch-amd64]
description=Debian stretch/amd64 CI testbed
users=debci,charles
root-users=debci,charles
groups=root,sbuild
root-groups=root,sbuild
type=directory
directory=/srv/chroot/stretch-debci
union-type=overlay
```

In retrospect, I should have created it with `debci update-worker` (see below).

The test bed can then used to run tests with `autopkgtest`. For example:

```
autopkgtest --user debci --output-dir /tmp/output-dir \
  r-bioc-biostrings -- schroot debci-stretch-amd64
```

A comprehensive output is kept after the tests are run.

```
ls /tmp/output-dir/
log  run-unit-test-packages  run-unit-test-stdout  summary  testbed-packages  testinfo.json  testpkg-ve

$ head /tmp/output-dir/*
==> /tmp/output-dir/log <==
adt-run [06:58:08]: version 4.4
adt-run [06:58:08]: host bubu; command line: /usr/bin/adt-run --user debci --output-dir /tmp/output-dir
adt-run [06:58:08]: testbed dpkg architecture: amd64
adt-run [06:58:08]: testbed running kernel: Linux 4.9.0-2-amd64 #1 SMP Debian 4.9.18-1 (2017-03-30)
adt-run [06:58:08]: @@@@@@@@@@@@@@@@@@@@ apt-source r-bioc-biostrings
Get:1 http://deb.debian.org/debian stretch/main r-bioc-biostrings 2.42.1-1 (dsc) [2228 B]
Get:2 http://deb.debian.org/debian stretch/main r-bioc-biostrings 2.42.1-1 (tar) [12.7 MB]
Get:3 http://deb.debian.org/debian stretch/main r-bioc-biostrings 2.42.1-1 (diff) [4008 B]
adt-run [06:58:12]: testing package r-bioc-biostrings version 2.42.1-1
adt-run [06:58:12]: build not needed

==> /tmp/output-dir/run-unit-test-packages <==
ca-certificates 20161130+nmu1
fontconfig  2.11.0-6.7+b1
fontconfig-config   2.11.0-6.7
fonts-dejavu-core   2.37-1
libblas-common  3.7.0-2
libblas3    3.7.0-2
libcairo2   1.14.8-1
libcurl3    7.52.1-5
libdatrie1  0.2.10-4+b1
```

```
libexpat1   2.2.0-2

==> /tmp/output-dir/run-unit-test-stdout <==

R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.

==> /tmp/output-dir/summary <==
run-unit-test        PASS

==> /tmp/output-dir/testbed-packages <==
adduser 3.115
apt 1.4.4
apt-utils   1.4.4
base-files  9.9
base-passwd 3.5.43
bash    4.4-5
binutils    2.28-5
bsdmainutils    9.0.12+nmu1
bsdutils    1:2.29.2-1
bzip2   1.0.6-8.1

==> /tmp/output-dir/testinfo.json <==
{
"virt_server": "autopkgtest-virt-schroot debci-stretch-amd64",
"cpu_model": "Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz",
"kernel_version": "Linux 4.9.0-2-amd64 #1 SMP Debian 4.9.18-1 (2017-03-30)",
"nproc": "4",
"cpu_flags": "fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
}
==> /tmp/output-dir/testpkg-version <==
r-bioc-biostrings 2.42.1-1
```

The *debci* system is there to manage a queue of tests dispatched to workers, and collect the results.

4.6 Running debci locally

Debci is made of four components: an injector, a message queue, workers and a collector.

Installation instructions: `https://ci.debian.net/doc/file.INSTALL.html`

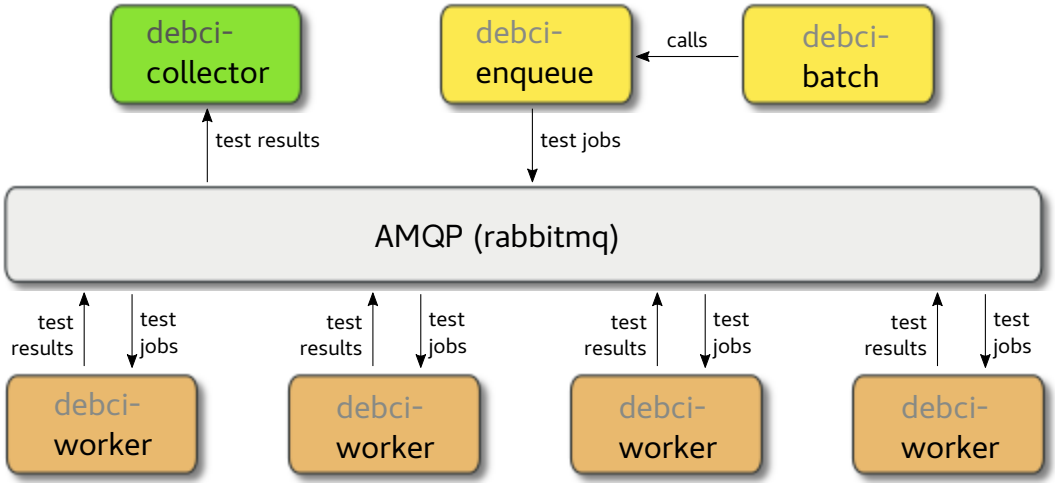


図 1　debci's architecture

sudo apt install debci debci-collector debci-worker

It is also recommended to use `apt-cacher-ng` (not sure if just installing is enough...).

4.6.1 Issues with the installation.

Disclaimer: maybe everything works well and I did not forget the instructions correctly? I did not have time to triple-check.

Debci's documentation says: "*As usual, you will prompted for the address of the AMQP server. If the rabbitmq-server is on the same host, just leave it blank*". In my case, I did not see a prompt. But since I use a local server, there was no problem.

If there are problems with rabbitmq, it is useful to check at the web interface. But it is not installed by default.

```
rabbitmq-plugins enable rabbitmq_management
```

Then, visit `http://localhost:15672/` (login:guest, password:guest0).

There is also a command-line tool to query the queues: `rabbitmqadmin`.

4.6.2 Other issues.

`https://bugs.debian.org/809652`: *debci: setup sets stamps even if setup failed*

This bugs prevents debci commands to try again when they should. For example, this command fails for a good reason (lxc not installed).

```
$ sudo debci update-worker
Starting testbed setup: Tue May 30 21:17:47 JST 2017
E: lxc is not installed
Finished testbed setup: Tue May 30 21:17:47 JST 2017
```

What happens if one configure lxc as needed and restarts the command ?

```
$ sudo debci update-worker
Starting testbed setup: Tue May 30 21:17:50 JST 2017
I: testbed already updated in the last 12h, no need to update
```

It refuses to run for the next 12 hours !

Fortunately the solution is simple: there is a `--force` option.

The debci index generator was crashing and I could not debug the problem, but I happened to have to reboot my laptop this week, and the problem disappeared... (Unfortunately, `systemd` did not keep the logs of the previous boots... how did I configure it so wrong ?)

4.6.3 Configuration

`lxc` is default, but I wanted to use `schroot`. Either I had to pass `--backend schroot` to all the commands, or I had to add it to the configuration (same for `--suite stretch`).i

The configuration files are in `/etc/debci/` and `/debci/conf.d/`. By default, these directories are empty, with no example files. But study of the source code of debci indicates that the configuration files are sourced by the shell to define environment variables. Further study of the debci code suggested that the names of the variables can easily be guessed. So here is my configuration.

```
$ cat /etc/debci/conf.d/local.conf
debci_suite=stretch
debci_backend=schroot
```

TODO: submit an example configuration file via the BTS...

4.6.4 Et c'est parti !

Let's test all R/Bioconductor packages in Squeeze.

```
apt-cache search r-bioc* | cut -f1 -d ' ' |  xargs debci enqueue
```

And voil'a, `debci` did the rest.

```
$ ls /var/lib/debci/data/packages/stretch/amd64/r/ | more
r-bioc-affy
r-bioc-affyio
r-bioc-altcdfenvs
[ plus many more ...]
r-bioc-metagenomeseq
r-bioc-multtest
r-bioc-phyloseq
```

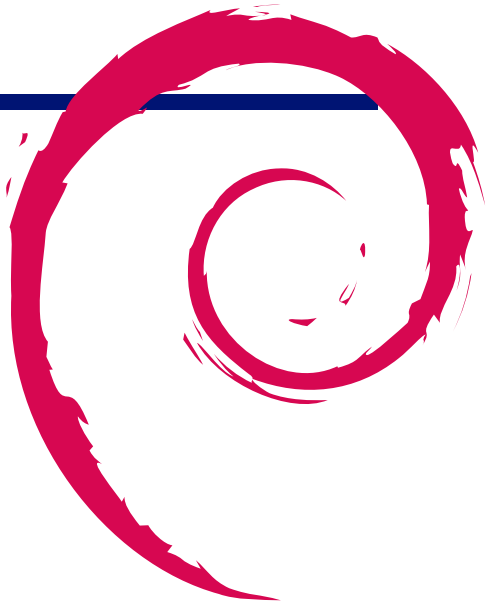
4.7 Conclusion

I still have to dig in the results.

There seems to be some autogenerated HTML pages, but I still have to figure out how to get nice diffs like on `ci.debian.net`. Example file path:

```
file:///var/lib/debci/data/.html/packages/r/r-bioc-affy/stretch/amd64/index.html
```

Final conclusion: I hope to use *debci* more extensively during the next release cycle to test potential backports.



5 メモ

Debian 勉強会資料

2017 年 6 月 18 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）