The Debian Continuous Integration project

Antonio Terceiro

August 26th, 2018

Introduction

The role of distributions

- Provide a convenient user experience for managing and using a system
- Be a bridge between upstream developers and users
  - Get upstream software to users
  - Integrate different upstream components
  - Triage and forward feedback and patches back to upstream

How Debian works

- *unstable*: receives updates directly from developers
- *testing*: updates migrate from *unstable*
  - package has no release-critical bugs
  - all dependencies are also eligible for migration
  - **package has spent at least 2/5/10 days in unstable**
  - migration is automated, and automatic
- *stable*: every ~2 years *testing* becomes *stable*
- Freeze: unstable $\rightarrow$ testing migration is no longer automatic

Distribution QA

- "If it builds, it works"
  - Common attitude in large transitions (libraries, toolchains)
  - Doesn't work always
- Sometimes focused on critical paths
  - installation is a big concern
  - less common use cases are not prioritized
- Sometimes general QA is relegated to "QA people"

Automated tests

- Undeniably useful
- Test-Driven Development (TDD)
- Regression testing
- Different levels
  - Unit testing
  - Integration testing
  - Functional testing

Putting it all together

- **"package has spent at least 2/5/10 days in unstable"**
  - depends on the package having enough users in *unstable*
  - depends on users finding bugs soon enough
- we want *testing* to be safe for everyday use, and to always be in a releaseable state
- Can we find bugs ealier?
- Can we avoid having the most basic bugs from blowing up on users' faces?
- Let's apply automated testing

Debian Continuous Integration Project

History

- 2006: first prototype of `autopkgtest`
- early 2014: initial hacking on Debian CI
- mid 2014: two GSOC interns working on Debian CI
- late 2015: distributed architecture
- mid 2017: initial discussions about testing migration integration
- mid 2018: testing migration uses Debian CI test results as input

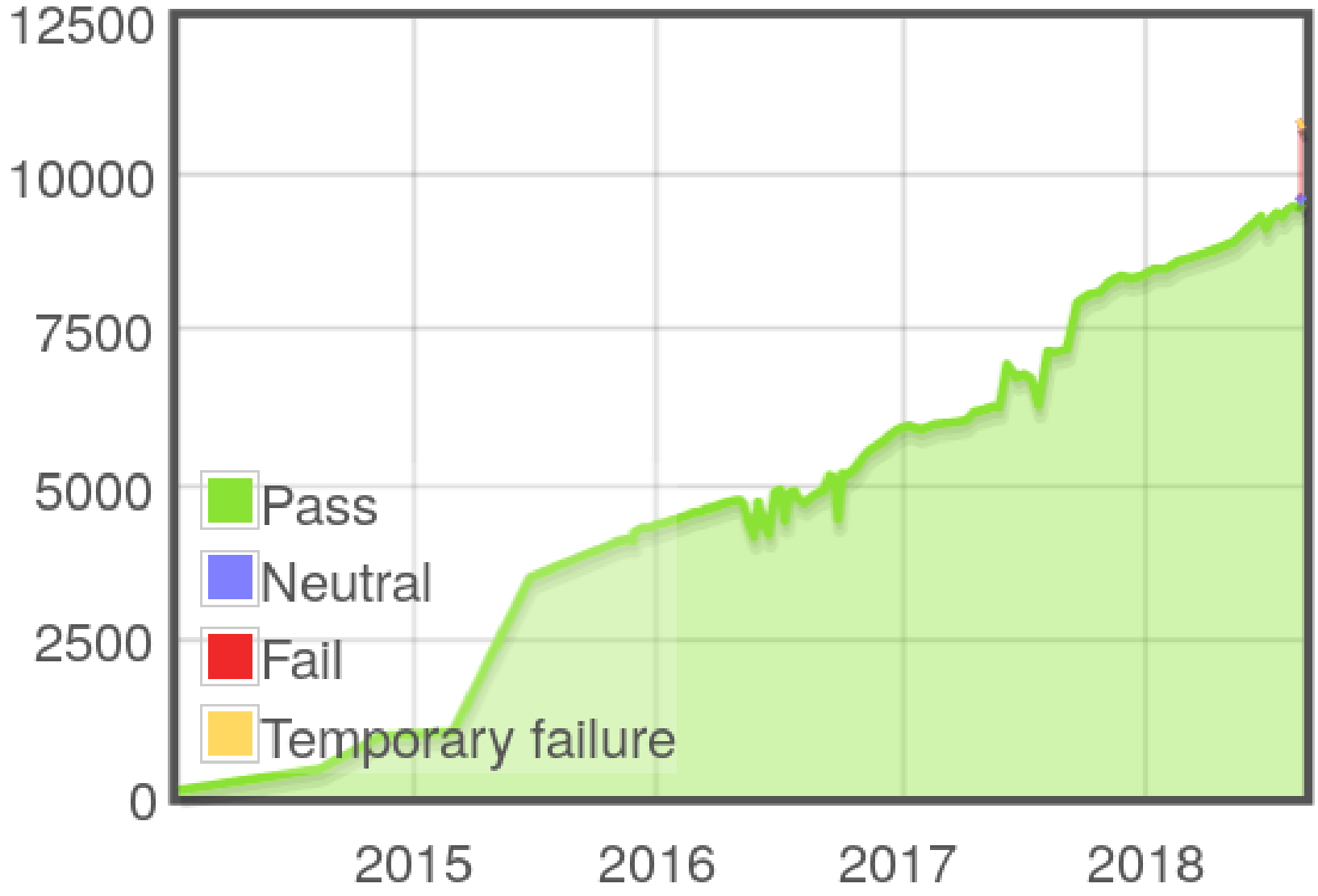

Figure 1: Number of packages tested in *unstable*

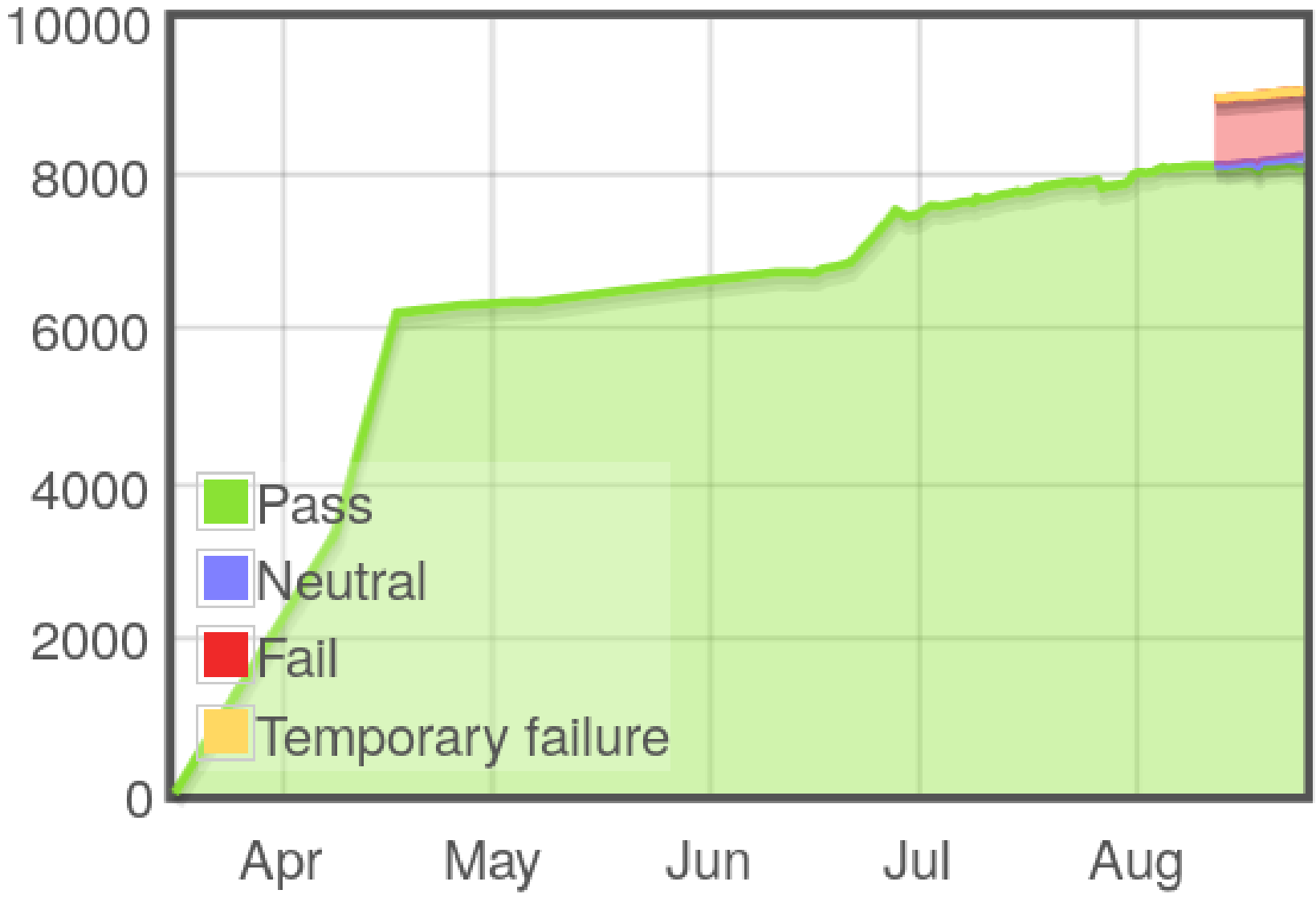

Figure 2: Number of packages tested in *testing*

Testing migration

- No regressions: package migrates *faster*
  - tests for package pass
  - tests for all packages that depend on it also pass
- Regressions: package migrates *slower*
- For the future, the plan is to completely *block* migration on regressions

The platform: tools

- `autopkgtest`: the test runner tool
- `autodep8`: tool to generate test specifications for well-known types of packages
- `debci`: the CI platform itself
  - can be installed locally by any user
  - official instance running at ci.debian.net
- `debian-ci-config`: configuration management repository for ci.debian.net
  - not a package
- all source code available at salsa.debian.org/ci-team

The platform: web UI

- mostly static HTML
  - pro: no performance concerns
  - con: not up to date all the time
- HTTP API

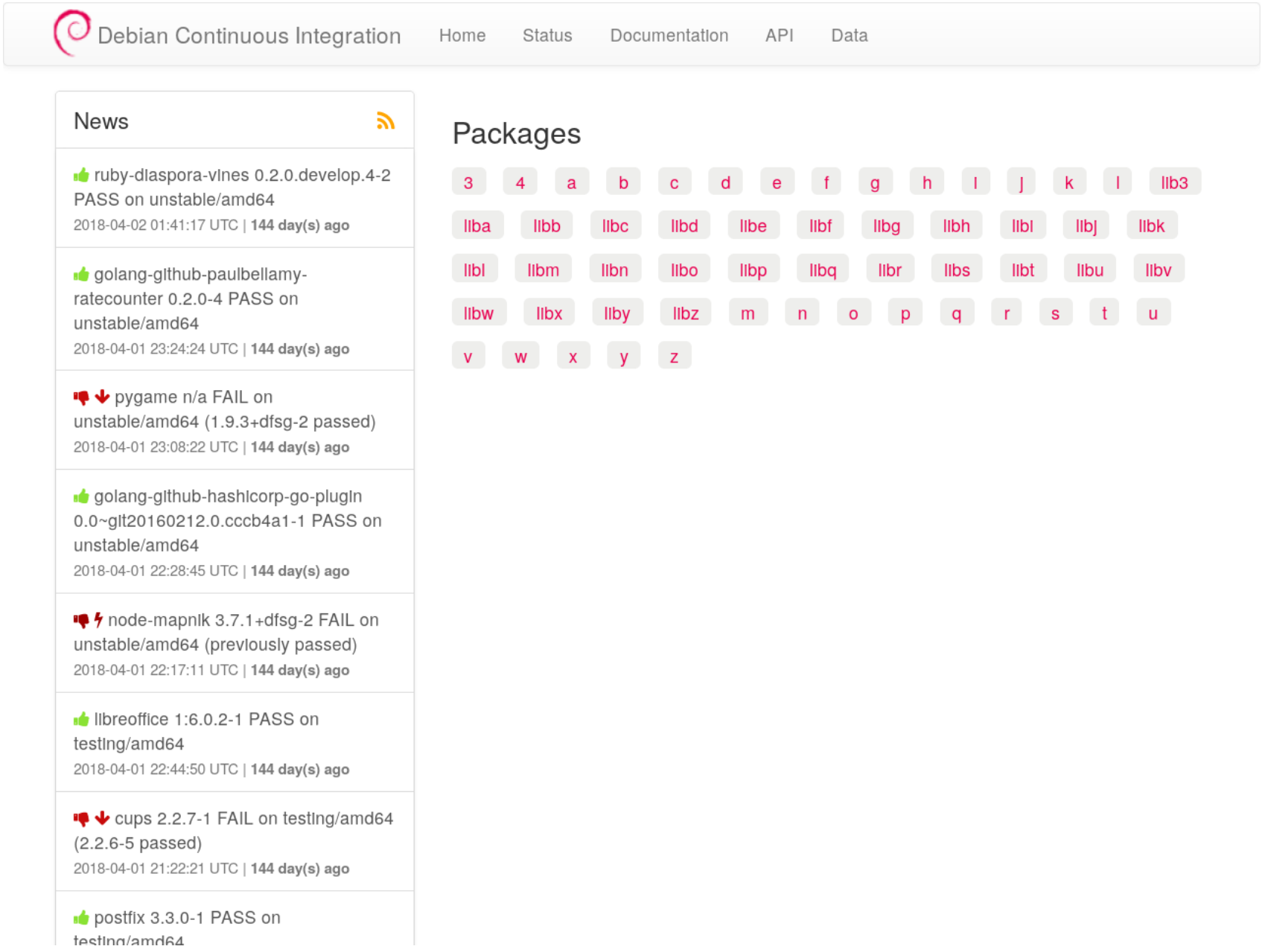

Figure 3: https://ci.debian.net/

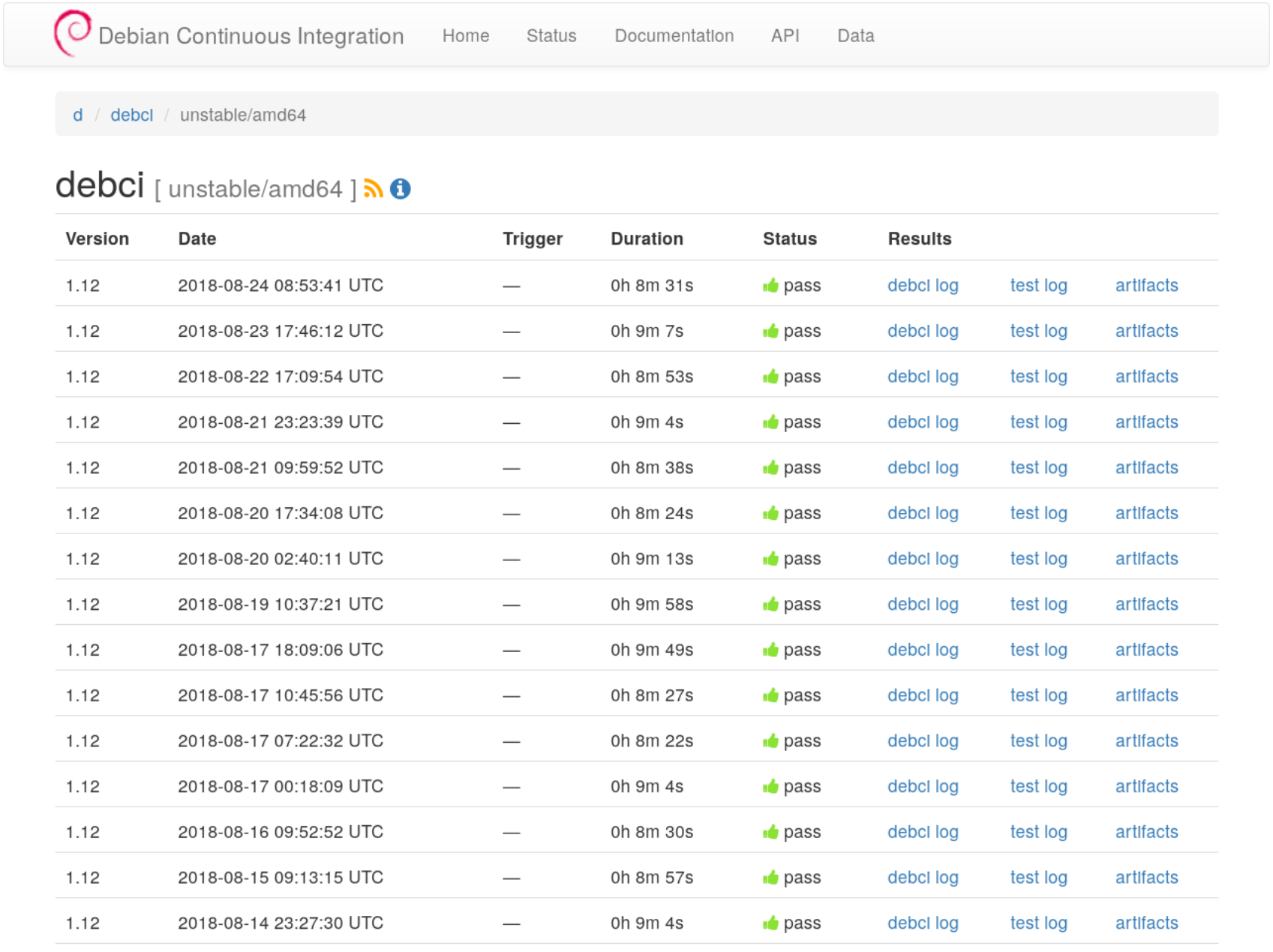

Figure 4: https://ci.debian.net/packages/d/debci/unstable/amd64/

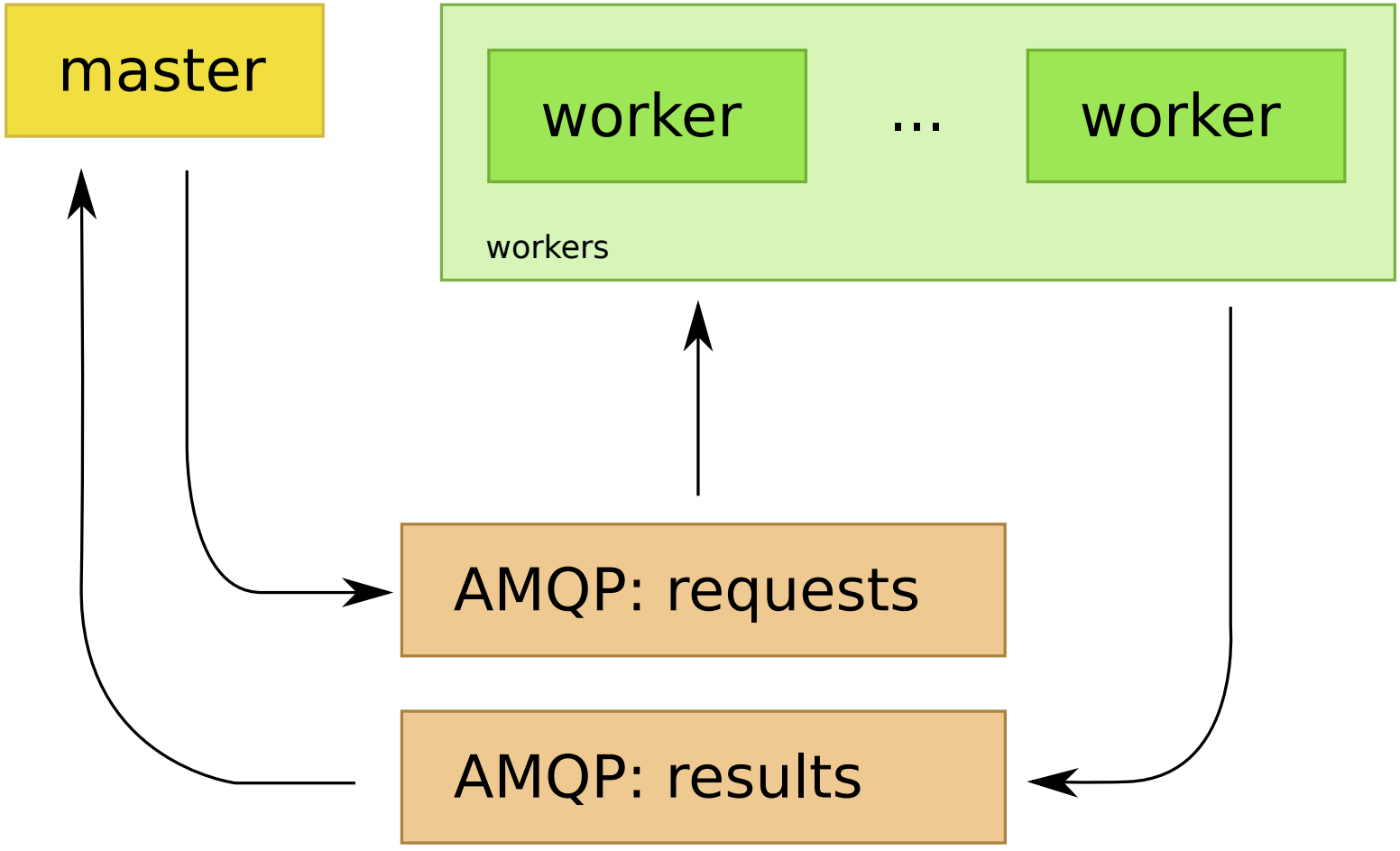

Figure 5: Architecture

The platform: hosting

- 1 master node
- 10 worker nodes, running tests for the `amd64` architecture
- ci.debian.net is kindly hosted on AWS on a Debian account with credits sponsored by Amazon.

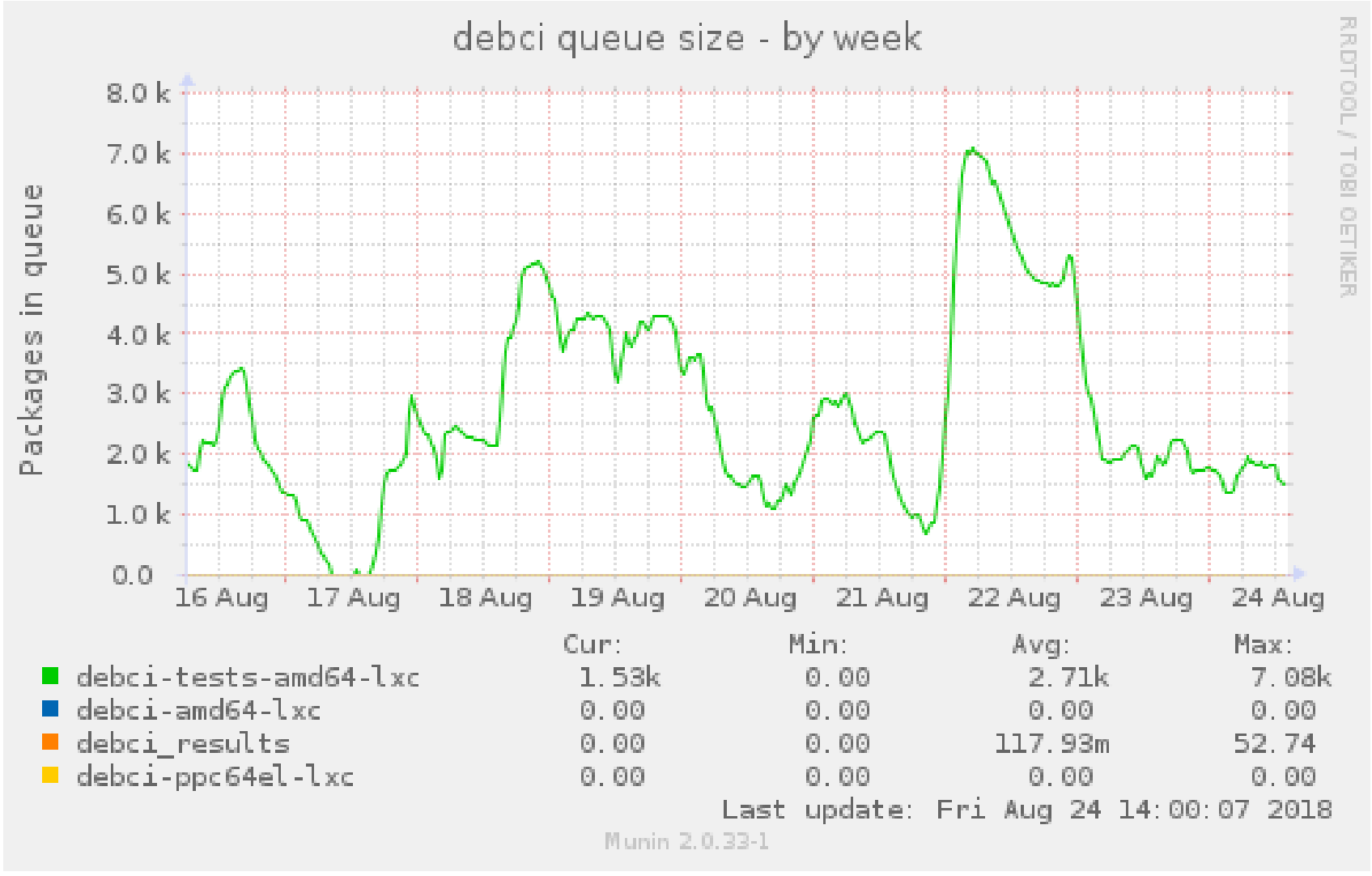

Figure 6: Test job queue

The platform: test execution environment

- standard autopkgtest operation
  - developers can easily reproduce the exact same test run on their local machine
- tests are run in LXC containers
  - full system container (*vs* application containers)
  - clean system: nothing installed besides what the test itself specifies

The Developer Point of view (writing tests)

Basic concepts

- Tests are part of the source package
  - every maintainer is responsible for his/her own testing
- Specification: DEP-8
- Tests defined in `debian/tests/control`
- Source metadata (`debian/control`) must provide a `Testsuite:` field
  - added automatically if `debian/tests/control` exists by `dpkg` >= 1.17.11
- Tests must exercise the package *installed* on a system, i.e. it should not test the *source tree*.

Example

```
$ grep Source: debian/control
Source: auto-apt-proxy
$ cat debian/tests/control
Tests: apt-integration
Depends: @, apt-cacher-ng

Test-Command: clitest debian/tests/apt-cacher-ng.txt
Depends: @, clitest, apt-cacher-ng

[...]

Tests: remove, reinstall
Restrictions: needs-root
Depends: @, clitest
$
```

autopkgtest

virt: LXC

```
$ autopkgtest --no-built-packages . \
  -- lxc --sudo autopkgtest-unstable-amd64
```

Runs the tests from the source package in the current directory (`.`), but using binary packages from the repository.

virt: null

```
$ autopkgtest --no-built-packages . -- null
```

Runs the tests from the source package in the current directory (`.`) againt the *current system* ("null" virtualization). Requires that the needed packages are already installed.

autodep8

- Solves the problem of several similar packages having identical `debian/tests/control` (e.g. “Ruby libraries”, “Kernel modules using DKMS”)
- When a package does not have an explicit `debian/tests/control`, `autopkgtest` calls `autodep8` to produce one.
- Instead of duplicating `debian/tests/control`, packages declare a special value for their `Testsuite:` field and let autodep8 produce the test specification.
- `autodep8` can also detect supported package types by looking at the contents of packages
- Currently supported: Ruby, Perl, Python, *NodeJS*, DKMS, R, Emacs Lisp, Go, Octave

autodep8: example

```
$ grep Source: debian/control
Source: ruby-json
$ cat debian/tests/control
cat: debian/tests/control: No such file or directory
$ autodep8
Test-Command: gem2deb-test-runner --autopkgtest
  --check-dependencies 2>&1
Depends: @, ruby-test-unit, gem2deb-test-runner
```

More information on writing tests

Debian CI documentation: https://ci.debian.net/doc/

Patterns for Writing As-Installed Tests for Debian Packages
https://deb.li/pattestdeb

Conclusions

Lessons

- Adoption *takes time*
- Improvements to the system are always needed
- Using the very same tools that developers can use directly makes everyone's lifes easier.
- Most of the ideas (and maybe even some of the tools) presented here could be applied to other distributions. I would love to chat about that.

Thank you

Contact: terceiro@debian.org