

The CPAN Toolchain

137 859 Perl modules, 30 268 distributions,
11 695 authors, 255 mirrors

Weborama Tech Day 2014

September 5, 2014

1 CPAN

- Tools built around CPAN

2 Vendor distributions

- Hosting vendor distributions

3 Outro

What is CPAN?

A public repository of practically all open source Perl code.

Very, **very** few open source Perl modules are not on CPAN.

The “killer app” of Perl

CPAN, the good parts:

- zero barrier to entry
- it has a module for everything
- enables high rate of code reuse

1. The Perl community is generally not big on snippet sharing (they learned their lesson with Matt’s Script Archive). Some modules are just the result of one author thinking “I keep writing this thing, I should get it right once and write a 10-loc module with it”, and thanks to him everyone gets it right now.
2. Because it’s trivial and cheap to declare a dependency.

The “killer app” of Perl

CPAN, the bad parts:

- zero barrier to entry
- it has a module for everything
- enables high rate of code reuse

1. This means you can find a lot of crappy modules on CPAN.
2. This means you can find a **lot** of crappy modules on CPAN.
3. Fortunately you can sift the good from the bad relatively easily. When comparing several options, check CPAN Testers to eliminate the ones that don't pass their tests on your platform. Check out the ratings (no ratings at all for a distribution probably means nobody really uses it). Check out the reverse dependencies (from different authors).
4. Because it's so simple to install a distribution and add it as a dependency, and because many simple things are implemented as third party modules, some distributions have very deep and wide dependency trees. `Dist::Zilla` is a particular offender with 6934 files installed.

A public mirror of Perl code

```
authors/  
  01mailrc.txt.gz  
  02authors.txt.gz  
  id/  
    A/  
    B/  
    ...  
    E/  
      ET/  
        ETHER/  
          CHECKSUMS  
          Class-Method-Modifiers-2.07.tar.gz  
          Module-Metadata-1.000019.tar.gz  
          Moose-2.1210.tar.gz  
          MooseX-Types-0.44.tar.gz  
          Safe-Isa-1.000004.tar.gz  
          strictures-1.004004.tar.gz  
      ...  
modules/  
  02packages.details.txt.gz  
  03modlist.data.gz
```

CPAN clients know how to navigate this conventional structure

1. This mirror could be exposed via HTTP (like Pinto, most mirrors), FTP (out of fashion), or just the local filesystem.
2. The important files here are all the *.tar.gz distribution tarballs, and the 02packages.details.txt.gz package index file which maps package names to tarball paths, e.g. Moose::Role version 2.1210 is available in E/ET/ETHER/Moose-2.1210.tar.gz.
3. Because it's such a basic interface, tools can make use of it very easily.

The CPAN Testing Service

- unpacks every newly-uploaded distribution
- checks 51 different indicators – is there a README file, a LICENSE file, can the tarball be extracted correctly, does it provide documentation...
- calculates an overall “kwalitee” score

Kwalitee

“Kwalitee” is something that looks like quality, sounds like quality, but is not quite quality.

1. This is very useful to module authors, not so much to users, except insofar as it turns authors better at making distributions.
2. <http://cpants.cpanauthors.org/author/WEBORAMA> – oops
3. From the DESCRIPTION of `Test::Kwalitee`: “Kwalitee is an automatically-measurable gauge of how good your software is. That’s very different from quality, which a computer really can’t measure in a general sense. (If you can, you’ve solved a hard problem in computer science.)”

CPAN Testers

Probably the largest continuous integration service in the Open Source community



- individual volunteers from the community use special CPAN clients to pull newly-uploaded distributions
- they each attempt to run the distribution's unit tests, and report the results to the cpantesters service
- the result is one big matrix of test results without any work done on the distribution author's part

CPAN Testers

- after a couple days, every dist version you've uploaded has been tested on all Perl versions from 5.8.8 to 5.21.1
- and on most platforms, from vanilla GNU/Linux to win32s to cygwin to most BSDs
- and most combinations thereof

If you don't see the reports you expect to see, please consider altering the preferences above before filing a fault report for the site.

Report Summary

Version Summary:
0.003 
0.002 

Aristotle:
"Quality is not an act, it is a habit"

Selected Reports

This page details the test reports submitted for the CPAN distribution `Test-SetupTeardown`.

To view individual reports, please click the grade test associated with each report.

Module version: `Test-SetupTeardown-0.003`

Test-SetupTeardown 0.003 (617 ALLs 1 FAIL 616 PASSes)

Grade	Perl version	OS name	OS version	Architecture
PASS	5.20.1 RC1	GNU/Linux	3.2.0-4-amd64	x86_64-linux
PASS	5.14.3	GNU/Linux	2.6.27.25-78.2.56.k3.i686	i686-linux-thread-multi-64bit-td
PASS	5.12.2	Windows (Win32)	5.1	MSWin32-x86-multi-thread
PASS	5.12.2	GNU/Linux	2.6.27.25-78.2.56.k3.i686	i686-linux-thread-multi-64bit-td
PASS	5.12.0	GNU/Linux	2.6.27.25-78.2.56.k3.i686	i686-linux-thread-multi-64bit-td
PASS	5.20.0	GNU/Linux	2.6.25-14.k3.i686	i686-linux-thread-multi-64bit-td
PASS	5.18.2	Windows (Win32)	6.2	MSWin32-x86-multi-thread
PASS	5.20.0 RC1	GNU/Linux	3.2.0-4-amd64	x86_64-linux-thread-multi
PASS	5.18.2	GNU/Linux	3.2.0-4-amd64	x86_64-linux-thread-multi

This is really useful, and definitely one of the things to look at when comparing several similar distributions. `YAML`, `YAML::Syck`, `YAML::XS` all basically do the same thing, but do they all build properly for your architecture and Perl version? If the pre-install tests are failing for you, maybe they're also failing for every Perl 5.16.3 user?

Other stuff

- the MetaCPAN ElasticSearch API
- community reviews, ratings, etc.
- <http://rt.cpan.org>
- many more...

1. The MetaCPAN API may be used to get notifications when a distribution you're interested in gets an update; list all past releases of a distribution (so that you can fetch them and do some crazy bisecting to guess when a new feature has been added)...
2. Community reviews and ratings also help in the egregious TIMTOWTDI cases.
3. RT is mostly superseded by the lightweight GitHub issue tracker.

What about non-CPAN distributions?

For any closed-source project, a good chunk of code will never be uploaded to CPAN.

- it contains business logic
- it contains proprietary information
- it's just plain not interesting for the community

So, we can forget about all those packaging conventions and distribution management tools, right?

Advantages to packaging vendor code for CPAN

Even if we never upload our stuff to CPAN we still gain many things from packaging it properly.

- installation to known (almost) standard paths
- declaration and installation of dependencies
- compatibility with existing CPAN clients
- everybody knows about t/

Install a CPAN module:

```
cpanm -v Data::UUID Install a Weborama module:
```

```
cpanm -v Weborama::Database
```

TL;DR: Packaging your vendor distributions like CPAN distributions allows you to treat both in exactly the same way and use the same tools everywhere. And it's one fewer step to publication.

How do we make them available to sites?

So everybody's convinced by now that packaging vendor stuff for CPAN is good.

We still need to host the tarballs somewhere a CPAN client can find it.



HYPNOTOAD COMMANDS YOU
TO BE CONVINCED

CPAN::Mini

- minicpan is one of the more mature solutions
- it fetches all the tarballs currently indexed in an existing CPAN mirror
- with CPAN::Mini::Inject, you can then insert your own distributions in your custom mirror
- it presents the same interface as a regular CPAN mirror so you can just point your CPAN client at it
- also useful if you want a cheap local CPAN mirror
- not so useful if you're not interested in e.g. all the joke Acme:: modules

Pinto and Stratopan

- Pinto is a more recent solution
- you can use the Pinto client to talk to a remote Pinto server
- more features: adding only your vendor distributions (and their dependencies pulled from CPAN), “pinning” distributions so they don’t get upgraded by mistake, reverting changes to the mirror
- Stratopan, developed by the same author, is basically Pinto as a service
- Weborama uses six Pinto instances: one each for the integration, pre-production and production environments, for two different software stacks

Questions?

Thanks for listening.

All the missing stuff:

- PAUSE, the Perl Author Upload SErver
- cpan, cpanp and cpanm
- BackPAN
- <http://deb.perl.it>