

Run Your Own CPAN

by brian d foy
brian.d.foy@gmail.com



The Comprehensive Perl Archive Network (CPAN) is the best part of the Perl community. I have available to me over 15,000 modules taking up over five gigabytes. Almost any task I need to do has probably already been turned into a CPAN module.

Even with the great benefit of CPAN, relying on any public repository represents risk. I don't control what's in the repository: I have to accept its composition, its upgrade schedule, and so on. As I develop my own applications, I want to control as much of it and its dependencies as possible so I can manage the complexity of work.

I want to make my own CPAN. I can take the best parts of the canonical, public Comprehensive Perl Archive Network (CPAN), take out the parts that I don't want, add my own, private module distributions, or even add locally-patched versions of public modules. Once created, I use my private CPAN just like I would any other CPAN mirror using the same tools.

My recent work indexing BackPAN makes it possible for everyone to make their own, private CPAN that they completely control. I've been calling this DPAN, since D comes after C, but also since this idea falls within the notion of what some people call DarkPAN---all the Perl stuff floating around that's not publicly available. I didn't make up these names myself, though. If you did, let me know so I can give you credit.

■ How CPAN works -----

The CPAN is at the heart of the Perl community, and stores the most popular and useful Perl modules. It makes it amazingly easy to program complex tasks very quickly by re-using the work of others. It's Perl's killer feature.

The CPAN tool chain is the circulatory system. It transports the modules from a CPAN mirror to the people using the modules, as well as taking the modules from authors and bringing them to CPAN. These tools make some assumptions that might not fit what I need though. For instance, they always want to install the latest version of a module and the latest version of all of its dependencies. The common tools don't support dependencies in a version range or a maximum version. They are also fairly dumb about versions, not knowing anything about major, minor, or point releases. As long as the number gets bigger, they don't complain.

CPAN itself is just a tree of directories containing mostly module distributions. People create accounts with the Perl Authors Upload Server (PAUSE) and that allows them to upload files to PAUSE, which then analyzes and indexes any distributions

it sees. Periodically, the master CPAN mirror syncs with PAUSE so CPAN gets all of the latest releases. As the master CPAN server syncs with PAUSE, other CPAN mirrors sync with the master CPAN server. In all, there are about 200 CPAN mirrors which eventually have the same content as PAUSE.

To allow CPAN clients to find the right distribution, PAUSE prepares some index files that map a module name to the distribution it is in. I've already explained these in "Making My Own CPAN" in *The Perl Review* 4.0 (Fall 2007), so I won't go through it again. It's enough to know that I need to generate the indices myself to run my own CPAN.

DPAN works by eliminating the PAUSE portion and limiting the mirroring, but keeping the rest of the process and toolchain. People use familiar tools and workflows so there is nothing new to learn. I create the repository I want, index it, and mirror it within my organization. The process is the same even if the access to the result isn't. People using DPAN as a source for their Perl modules do the same things with the same tools they'd use for a public CPAN. They can even use both at the same time.

■ The risks of CPAN -----

The trade-off for the usefulness of CPAN is that I have to accept some of its assumptions. These represent risks for my project. These are often the things that concern project managers, legal teams, and the hordes of other people whose job sometimes seem to be to annoy programmers with silly restrictions. They do have some valid points, though, so don't let a glib Perl fanboi mislead you into believing in magic ponies and unicorns. The DPAN is going to control all of these risks for me.

The risk of CPAN is often much lower than the benefit, which is why CPAN works, but by recognizing some of these risks, I can do a little extra work to mitigate them and to make my programming job easier, while at the same time lessening the concerns about the risk from the rest of my project team.

First, I have to depend on the work of other people who I do not control. Any author can upload to PAUSE anything they like. Virtually every Perl programmer has experienced downloading the latest version of a module only to discover that it has a bug in it. Since I can't control the work of other people, if I use CPAN as my only module source, I have to wait until the original author decides to fix the problem. In a common case, the new version of a dependency breaks and sets off a chain reaction of module installation failures as test suites fail. A failure in `Test::More`, for instance, essentially breaks all of