Managing Complexity With Perlbrew and Carton

David Larochelle
About Myself

• Long time Perl developer
• Berkman Center for Internet & Society at Harvard
• Lead Developer of the Media Cloud project
CPAN

- Comprehensive Perl Archive Network
- Over 100,000 Perl modules
- A module may already exist for your problem
- “Perl's killer app”
A simple example:
Which block is more readable?

my $r;
return [ grep { !$r->{ $_ }++ } @{ $urls } ];

Use List::MoreUtils qw (distinct )

(urls = [ distinct @ { $urls } ];
return $urls;
The Darkside of CPAN: Software Deployment

- Module Configuration
- Installation failures
  - Compile errors
  - Test failures
  - Unspecified dependencies
- Problem may be in a dependency not the module you're installing
CPAN: Deploying to multiple systems

- cpan install – gives the latest module version which may be different than what was tested
- Installation differences between versions
- Subtle module version incompatibilities
- Worst case: software works on development machines but malfunctions in production in non-obvious ways
Our Experience Installing Media Cloud to multiple systems

- Deploying to a new system required 1+ days of resolving CPAN install issues
- Bugs due to CPAN version differences
- Installing new modules required sys admin access
Deploying to multiple operating systems

- Different distribution versions contain different Perl versions
- Distributed Perl tends to lag the latest release
- Perl is deeply integrated into Debian & Ubuntu:
  - Vendor patches usually only for security issues
  - Manually upgrading system Perl is difficult and risky
Write for Oldest Perl on Deployed Systems

- Can't use new features:
  - Lower productivity
  - Less readable and reliable code
- Perl: excellent backwards compatibility but still version differences
“But I'm using an LTS distribution”

- Ubuntu LTS versions still in use after their Perl is EOLed by the community
- May encounter program crashes due to internal bugs in non-updatable System Perl
- Don't count on vendors patches for obscure Perl bugs
Dealing with bugs in older Perls

• OS upgrades so systems have newer Perl
  – Great if you can do it
  – May not be feasible do to system downtime, sys admin time, risk, etc.
  – Requires CPAN module reinstall

• Kludges to prevent crashes
  – Makes code ugly and hard to read
  – Hard to know if problem is fixed
  – E.g. Regex substitution crashes with long strings so split into multiple substrings, substitute, and concatenate
Possible CPAN Module Solutions

• Avoid / Minimize CPAN
• Package Modules as deb or RPM packages
• cpanm
• local::lib
• carton
Avoiding & Minimizing CPAN

- Purge modules you don't need
- Write your own code instead of using CPAN
- Works for small modules with fewer dependencies
- You can't write your own catalyst, moose, or DBIx
- Lots of work to reinvent the wheel
Our Solution Perlbrew & Carton

perlbrew

- Allows easy installation of many Perl versions
- Perl separate from system Perl
- Root access not required to install
- Root access not required to add CPAN modules
- Easy to switch between multiple Perl versions
Carton

- Tatsuhiko Miyagawa – cpanm, plack, etc
- Perl module dependency manager
- aka Bundler for Perl
- Allows installation of specific
- Module versions separate from the System modules
Using Perlbrew and Carton Together

- Insulates your application from the system
- Allows easy replication of the application environment
- Removes external constraints on which Perl version you use
Carton basics

• carton.lock file
  – Stored in source control
  – Tracks module versions
• Carton install – deployment
• Carton install Module::Name
• Module are stored in ./local
• Installed version information in ./carton
• Carton sets env vars and execs Perl
Downsides

- Carton is technically still alpha
- Hard to remove modules (but why bother)
- Occasional noise diffs for carton.lock
- In the worst case:
  - `rm local` and `.carton`
  - Rerun `carton install`
Our approach

• Place Perlbrew and Carton tarballs in source control
• Install script sets up installs Perlbrew and uses it to run carton to install modules
Wrapper Scripts

- Run carton within the Perlbrew environment
- Use carton to run scripts
- Carton wrappers for debugging and profiling
Portability

- Easy deployment to Ubuntu 12.04
- Also tested on Ubuntu 08.04
- Student got it to work on OSX
DEMO
Conclusions

- Thanks
  - Hal Roberts, Ethan Zuckerman, Nathan Matias, Linas Valiukas
- Questions?
- Slides online soon