Déployer des applications Perl

Vincent Pit

2012-03-05

Structure d'une distribution Perl

- code source des modules

```
$ find Mon-Module-0.01 -type f | sort
Mon-Module-0.01/bin/script
Mon-Module-0.01/Changes
Mon-Module-0.01/lib/Mon/Module.pm
Mon-Module-0.01/Makefile.PL
Mon-Module-0.01/MANIFEST
Mon-Module-0.01/META.yml
Mon-Module-0.01/README
Mon-Module-0.01/t/00-load.t
Mon-Module-0.01/t/91-pod.t
Mon-Module-0.01/t/92-pod-coverage.t
```

Structure d'une distribution Perl

- scripts

```
$ find Mon-Module-0.01 -type f | sort
Mon-Module-0.01/bin/script
Mon-Module-0.01/Changes
Mon-Module-0.01/lib/Mon/Module.pm
Mon-Module-0.01/Makefile.PL
Mon-Module-0.01/MANIFEST
Mon-Module-0.01/META.yml
Mon-Module-0.01/README
Mon-Module-0.01/t/00-load.t
Mon-Module-0.01/t/91-pod.t
Mon-Module-0.01/t/92-pod-coverage.t
```

Structure d'une distribution Perl

- tests

```
$ find Mon-Module-0.01 -type f | sort
Mon-Module-0.01/bin/script
Mon-Module-0.01/Changes
Mon-Module-0.01/lib/Mon/Module.pm
Mon-Module-0.01/Makefile.PL
Mon-Module-0.01/MANIFEST
Mon-Module-0.01/META.yml
Mon-Module-0.01/README
Mon-Module-0.01/t/00-load.t
Mon-Module-0.01/t/91-pod.t
Mon-Module-0.01/t/92-pod-coverage.t
```

Structure d'une distribution Perl

- script de configuration

```
$ find Mon-Module-0.01 -type f | sort
Mon-Module-0.01/bin/script
Mon-Module-0.01/Changes
Mon-Module-0.01/lib/Mon/Module.pm
Mon-Module-0.01/Makefile.PL
Mon-Module-0.01/MANIFEST
Mon-Module-0.01/META.yml
Mon-Module-0.01/README
Mon-Module-0.01/t/00-load.t
Mon-Module-0.01/t/91-pod.t
Mon-Module-0.01/t/92-pod-coverage.t
```

Structure d'une distribution Perl

- fichier meta (`META.{json,yml}`)

```
$ find Mon-Module-0.01 -type f | sort
Mon-Module-0.01/bin/script
Mon-Module-0.01/Changes
Mon-Module-0.01/lib/Mon/Module.pm
Mon-Module-0.01/Makefile.PL
Mon-Module-0.01/MANIFEST
Mon-Module-0.01/META.yml
Mon-Module-0.01/README
Mon-Module-0.01/t/00-load.t
Mon-Module-0.01/t/91-pod.t
Mon-Module-0.01/t/92-pod-coverage.t
```

Structure d'une distribution Perl

- MANIFEST

```
$ find Mon-Module-0.01 -type f | sort
Mon-Module-0.01/bin/script
Mon-Module-0.01/Changes
Mon-Module-0.01/lib/Mon/Module.pm
Mon-Module-0.01/Makefile.PL
Mon-Module-0.01/MANIFEST
Mon-Module-0.01/META.yml
Mon-Module-0.01/README
Mon-Module-0.01/t/00-load.t
Mon-Module-0.01/t/91-pod.t
Mon-Module-0.01/t/92-pod-coverage.t
```

Structure d'une distribution Perl

- autres (`README`, changelog, templates, code C/XS...)

```
$ find Mon-Module-0.01 -type f | sort
Mon-Module-0.01/bin/script
Mon-Module-0.01/Changes
Mon-Module-0.01/lib/Mon/Module.pm
Mon-Module-0.01/Makefile.PL
Mon-Module-0.01/MANIFEST
Mon-Module-0.01/META.yml
Mon-Module-0.01/README
Mon-Module-0.01/t/00-load.t
Mon-Module-0.01/t/91-pod.t
Mon-Module-0.01/t/92-pod-coverage.t
```

ExtUtils::MakeMaker

- Script de configuration : `Makefile.PL`
- Crée un `Makefile` pour GNU `make`
- Dans la distribution standard `perl` depuis 1994
- Invocation :

```
$ perl Makefile.PL
$ make
$ make test
$ make install
```

ExtUtils::MakeMaker

- Exemple :

```
$ cat Makefile.PL
use strict; use warnings;
use ExtUtils::MakeMaker;
WriteMakefile(
 NAME              => 'Mon::Module',
 AUTHOR            => 'Vincent Pit <perl@profvince.com>',
 LICENSE           => 'perl',
 VERSION_FROM      => 'lib/Mon/Module.pm',
 ABSTRACT_FROM     => 'lib/Mon/Module.pm',
 EXE_FILES         => [ 'bin/script' ],
 MIN_PERL_VERSION => '5.012',
 PREREQ_PM         => { 'Moose' => '1.21' },
);
```

Module::Install

- Script de configuration : `Makefile.PL`
- Surcouche de `ExtUtils::MakeMaker`
- Pas dans la distribution standard `perl`, donc à inclure dans le répertoire `inc` de votre distribution
- Invocation : idem que `ExtUtils::MakeMaker`

Module::Install

- Exemple :

```
$ cat Makefile.PL
use strict; use warnings;
use inc::Module::Install;
name            'Mon-Module';
author          'Vincent Pit <perl@profvince.com>';
license         'perl';
version_from    'lib/Mon/Module.pm';
abstract_from   'lib/Mon/Module.pm';
install_script 'bin/script';
perl_version    '5.012';
requires        'Moose' => '1.21';
WriteAll;
```

Module::Build

- Script de configuration : `Build.PL`
- Implémenté complètement en Perl
- Dans la distribution standard depuis `perl 5.10`
- Invocation :

```
$ perl Build.PL
$ ./Build
$ ./Build test
$ ./Build install
```

Module::Build

- Exemple :

```
$ cat Build.PL
use strict; use warnings;
use Module::Build;
my $build = Module::Build->new(
 module_name  => 'Mon::Module',
 dist_author  => 'Vincent Pit <perl@profvince.com>',
 license      => 'perl',
 script_files => [ 'bin/script' ];
 requires     => {
  'perl'  => '5.014',
  'Moose' => '1.21',
 },
);
$build->create_build_script;
```

Déploiement d'une distribution Perl

Comment installer cette distribution sur une machine ?

Sur N machines ?

Via le CPAN

- Uploader sa distribution sur PAUSE (`http://pause.perl.org`)
- L'installer avec un client CPAN :

```
$ cpan Mon::Module          # core depuis 1997
$ cpanp -i Mon::Module      # core depuis perl 5.10
$ wget http://cpanmin.us -O - | perl - Mon::Module
```

Via le CPAN

- Uploader sa distribution sur PAUSE (`http://pause.perl.org`)
- L'installer avec un client CPAN :

```
$ cpan Mon::Module          # core depuis 1997
$ cpanp -i Mon::Module      # core depuis perl 5.10
$ wget http://cpanmin.us -O - | perl - Mon::Module
```

Mais on n'a pas d'accès internet sur toutes les machines...

Via un miroir CPAN privé

- `CPAN::Mini` crée et synchronise une copie partielle ou totale du CPAN
- `CPAN::Mini::Inject` peut alors injecter votre distribution dans le miroir local
- Les clients CPAN n'ont plus qu'à utiliser ce miroir comme source

Via un miroir CPAN privé

- `CPAN::Mini` crée et synchronise une copie partielle ou totale du CPAN
- `CPAN::Mini::Inject` peut alors injecter votre distribution dans le miroir local
- Les clients CPAN n'ont plus qu'à utiliser ce miroir comme source

Mais mes utilisateurs ne veulent pas utiliser de client CPAN...

Via `auto_install`

- La fonction `auto_install` de `Module::Install` rajoute au `Makefile` des cibles qui installent les dépendances :

```
$ perl Makefile.PL
$ make installdeps
$ make
$ make test
$ make install
```

- Permet aussi de choisir intéractivement des fonctionnalités facultatives

Via `auto_install`

- La fonction `auto_install` de `Module::Install` rajoute au `Makefile` des cibles qui installent les dépendances :

```
$ perl Makefile.PL
$ make installdeps
$ make
$ make test
$ make install
```

- Permet aussi de choisir intéractivement des fonctionnalités facultatives

Mais je veux plutôt installer les paquets de ma distribution...

Via le gestionnaire de paquets du système

- L'utilitaire `cpan2dist` de `CPANPLUS` peut automatiser la création de paquets :

```
$ dists="Deb Fedora Mageia Slackware Arch Gentoo SUSE"
$ for dist in $dists; do                                \
    cpan2dist --format="CPANPLUS::Dist::$dist" \
              --buildprereq Mon::Module ;       \
  done
```

- Ces paquets peuvent être incorporés à un dépot local

Via le gestionnaire de paquets du système

- L'utilitaire `cpan2dist` de `CPANPLUS` peut automatiser la création de paquets :

```
$ dists="Deb Fedora Mageia Slackware Arch Gentoo SUSE"
$ for dist in $dists; do                                \
    cpan2dist --format="CPANPLUS::Dist::$dist" \
              --buildprereq Mon::Module ;       \
  done
```

- Ces paquets peuvent être incorporés à un dépot local

Mais je veux distribuer un seul fichier...

Via App::FatPacker

- App::FatPacker analyse les dépendances d'un script Perl et les rassemble en un seul fichier :

```
$ fatpack trace script.pl
$ fatpack tree $(fatpack packlists-for          \
                            `cat fatpacker.trace`)
$ (echo '#!/usr/bin/env perl'; fatpack file;    \
                  cat script.pl) > script-packed.pl
$ rm -rf fatpacker.trace fatlib
```

Via App::FatPacker

- App::FatPacker analyse les dépendances d'un script Perl et les rassemble en un seul fichier :

```
$ fatpack trace script.pl
$ fatpack tree $(fatpack packlists-for          \
                          `cat fatpacker.trace`)
$ (echo '#!/usr/bin/env perl'; fatpack file;    \
                cat script.pl) > script-packed.pl
$ rm -rf fatpacker.trace fatlib
```

Mais mes utilisateurs ne veulent pas installer perl...

Via PAR

- L'utilitaire `pp` de `PAR::Packer` peut créer un exécutable contenant `perl` et toutes les dépendances d'un script :

```
$ pp -B -o script.exe script.pl
$ ./script.exe
```

Merci!