Bootstrapping a Perl Development Environment

https://github.com/hesco/perl_toolchain

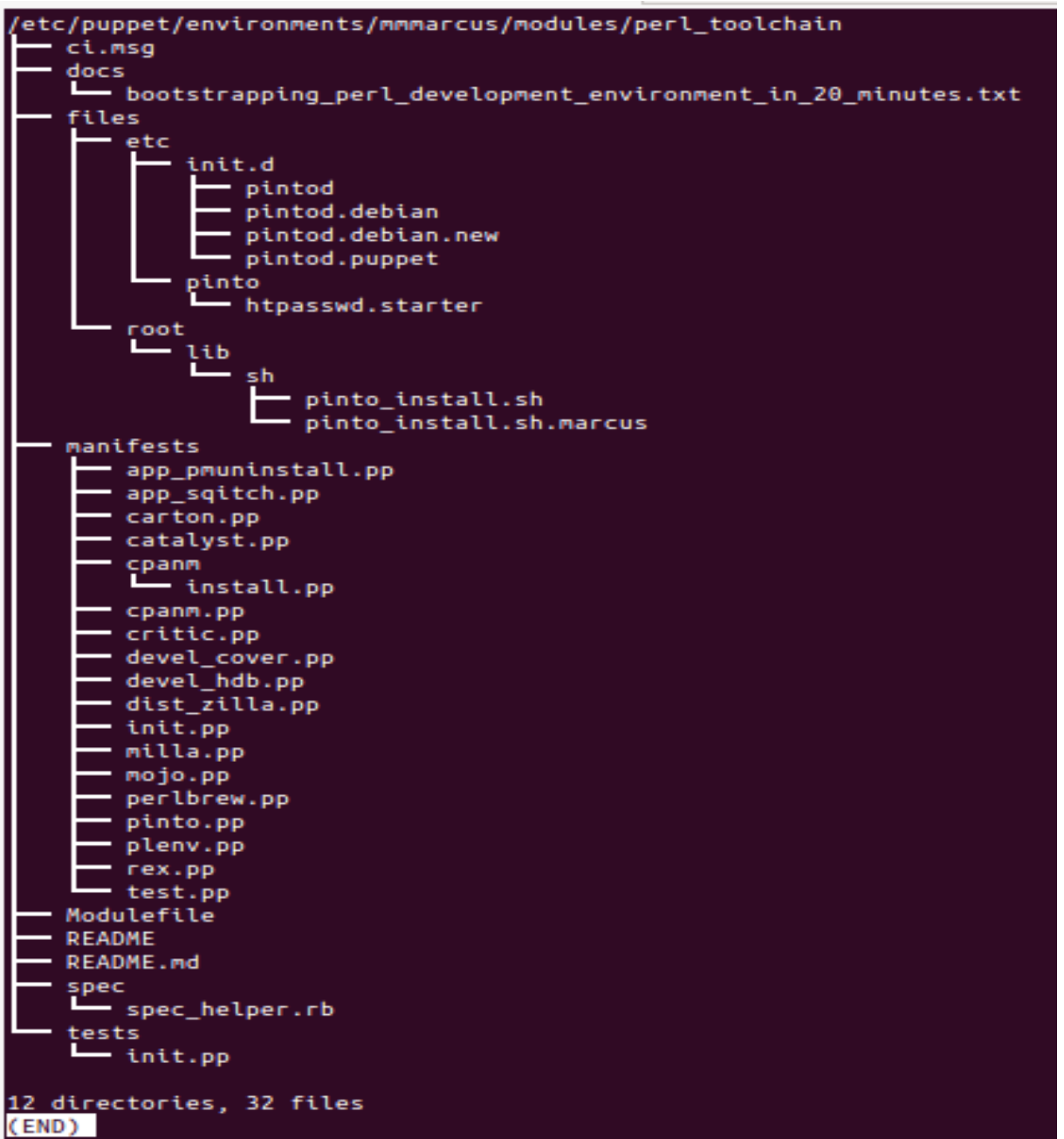
by: Hugh Esco

YAPC::NA::2014
Orlando Florida
Wednesday, June 25th, 2014

Configuration Management

* manage and document a node's configuration
* common options
** puppet
** chef
** ansible

perl_toolchain puppet manifest



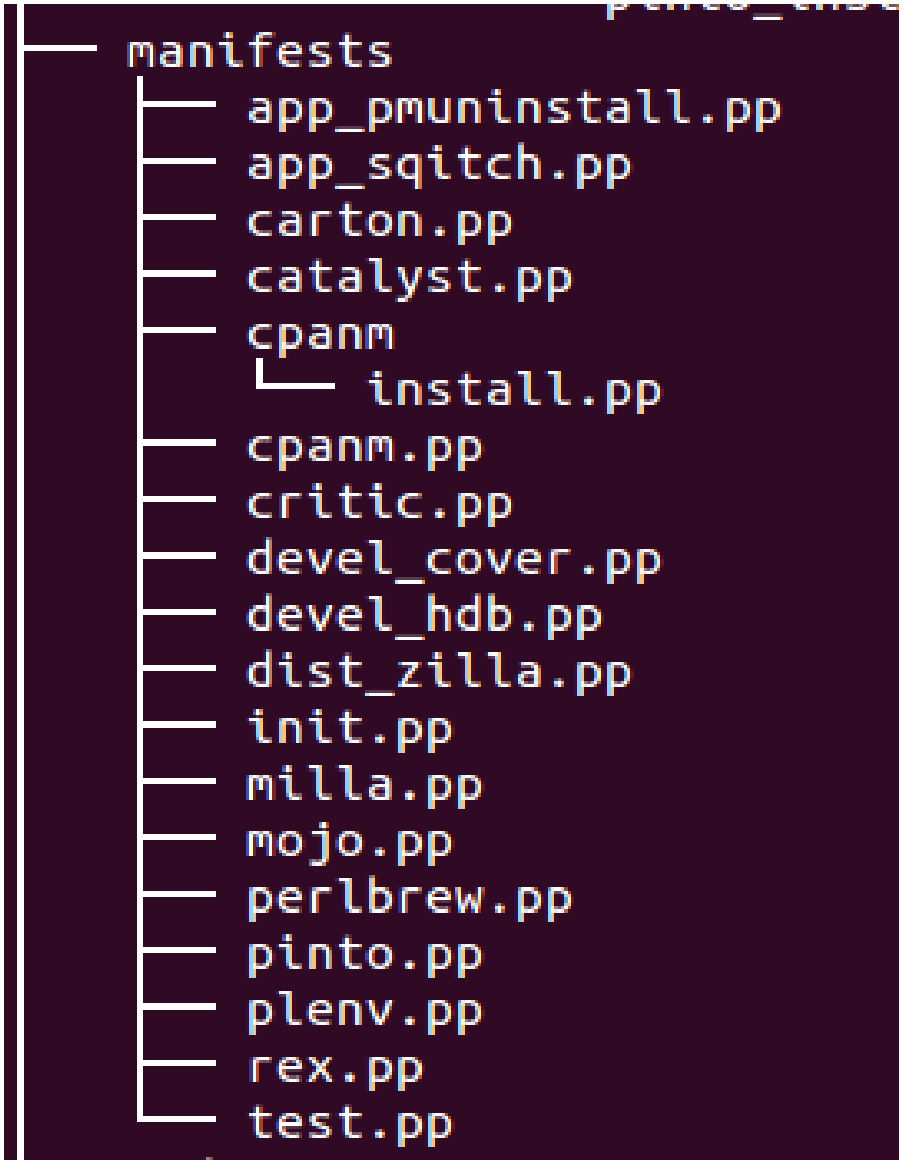
other development tools I use perhaps they belong here, not sure.

- compass_sass
- drush
- fpm
- libcloud
- vagrant

Thanks to alnewkirk

```
# Our cpanm is actually a symlink to the pinto
# installation.  See: perl_toolchain::pinto

# Thanks to alnewkirk on #perl-help channel
# http://pastie.org/8443394
# usage:
#   profile::utility::cpan { 'Dancer':
#         using => 'cpanm',
#        module => 'Dancer'
#   }

class perl_toolchain::cpanm::install {

  # cpan dependency installation
  define perl_toolchain::cpanm::install (
    $using  = '/usr/bin/cpanm',
    $perl   = '/usr/bin/perl',
    $local  = '/opt/local',
    $source = undef,
    $module = undef
  )
  {
    exec { "cpanm:${module}":
      environment => "PERL5LIB=${local}/lib/perl5:$PERL5LIB",
          command => "${using} -L ${local} ${module}",
          onlyif  => "${perl} -e \"!eval q{require ${module}}?exit(0):exit(1)\""
    }
  }

}
```

My Perl Development Environment



Puppet Development Workflow

- I develop my puppet manifests here:
  - /home/hesco/sandbox/marcus/puppet
- a shell script provides static lint, syntax checks and pushes the clean results here:
- /etc/puppet/modules/marcus/
- Although this tree structure is now deprecated and the latest version of puppet suggests:
- /etc/puppet/environments/marcus/modules/

Poor Man's External Node Classifier

```
[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter

server = mmmarcus
certname = mmmarcus
module_repository = http://forge.puppetlabs.com
modulepath = /etc/puppet/modules/$environment:/usr/share/puppet/modules

[master]
ssl_client_header = SSL_CLIENT_S_DN
ssl_client_verify_header = SSL_CLIENT_VERIFY
environmentpath = $confdir/environments
basemodulepath = $confdir/modules:/usr/share/puppet/modules
reports = store, http
bindaddress = 0.0.0.0
vardir = /var/lib/puppet {owner = root, mode = 644}
logdir = /var/log/puppet

[agent]
server = mmmarcus
environment = mmmarcus
```

Puppet configuration

- /etc/puppet/manifests/site.pp --
  - import 'nodes.pp'
- /etc/puppet/manifests/nodes.pp --
  - node 'marcus10.yourmessagedelivered.com' { include marcus_init }
- /etc/puppet/modules/marcus/marcus_init/manifests/init.pp

Poor Mans ENC Class List

Focusing on just the development tools now managed with puppet:

/etc/puppet/modules/marcus/marcus_init/manifests/init.pp --

```
class marcus_init {

  include puppet
  include utilities::dev_utils
  include utilities::vm_tools
  include utilities::removed_packages
  include git
  include ymd_dev_env
  include ymd_dev_env::bats
  include ymd_dev_env::jshint
  include ymd_dev_env::phantomjs
  include ymd_dev_env::bluefish
  include vagrant
  include perl_toolchain // <-----
  include libcloud
  include drush
  include fpm
  include db_clients
  include client1_dev_env
  include client2_dev_env
  include client3_dev_env
  include compass_sass
  // plus the office, entertainment and other
  // modules used to manage a primary home desktop.

}
```

perl_toolchain/manifests/init.pp

```
class perl_toolchain {
    include perl_toolchain::pinto
    include perl_toolchain::cpanm
    include perl_toolchain::plenv
    # include perl_toolchain::perlbrew
    include perl_toolchain::carton
    include perl_toolchain::mojo
    include perl_toolchain::milla
    include perl_toolchain::app_sqitch
    include perl_toolchain::devel_cover
    include perl_toolchain::devel_hdb
    include perl_toolchain::rex
    include perl_toolchain::test
    include perl_toolchain::app_pmuninstall
    include perl_toolchain::critic
    include perl_toolchain::dist_zilla

    package { 'perl-doc':
        ensure => latest,
    }

    file { '/usr/bin/cpanm':
        ensure => 'link',
        target => '/opt/local/pinto/sbin/cpanm',
    }

}
```

Sqitch – database change management application

```
app_sqitch.pp

Its author, David Wheeler, describes sqitch
as 'a database change management application'.

I mangle the config a bit from this,
but here is what init does:

$ sqitch init project_name

Created sqitch.conf
Created sqitch.plan
Created deploy/
Created revert/
Created verify/
```

Sqitch command vocabulary

```
sqitch understands these commands:

        add
        bundle
        checkout
        config
        deploy
        help
        init
        log
        plan
        rebase
        checkout
        revert
        rework
        status
        show
        tag
```

The sqitch directory

This is the sql/ directory on my current project:

```
$ tree -L 2 ../sql/
../sql/
├── mysql
│   └── sqitch.plan
├── pg
│   ├── deploy
│   ├── revert
│   ├── sqitch.plan
│   └── verify
├── sqitch_
├── sqitch.conf
├── sqlite3
│   ├── deploy
│   ├── revert
│   ├── sqitch.plan
│   └── verify
└── product_leads_dev
```

* sqitch_ and product_leads_dev are sqlite3 databases.

sqitch deploy

```
This is how I deploy schema changes --

script/deploy_app.sh --

function deploy_version_of_db () {

  cd "$DEPLOY_DIR/sql"
  /usr/bin/sudo -u hesco /usr/bin/perl -CAS -I/opt/local/sqitch/lib/perl5 /usr/bin/sqitch deploy app-$MODE
  cd -

  export DB_TAG='db_tag not yet implemented'
  return
}
```

sqitch deploy output

```
It produces output like this:

Adding registry tables to db:sqlite:sqitch_
Deploying changes to sqlite-dev
  + prospects ........................ ok
  + list_types ....................... ok
  + students ......................... ok
  + states ........................... ok
  + campus_types ..................... ok
  + campus ........................... ok
  + populate_states .................. ok
  + populate_campus .................. ok
  + populate_campus_types ............ ok
  + campus_unsupported @v0.08 ........ ok

 --- slide ---

The $db_engine/{deploy|revert|verify}/ directories are filled
with .sql files written in the native SQL dialect for each engine.

$ cat pg/deploy/states.sql
-- Deploy states
```

sqitch uses native SQL

```
The $db_engine/{deploy|revert|verify}/ directories are filled
with .sql files written in the native SQL dialect for each engine.

$ cat pg/deploy/states.sql
-- Deploy states

BEGIN;

  CREATE TABLE states (
    id serial UNIQUE,
    cdh_country_code integer,
    cdh_state_code integer UNIQUE,
    country_code varchar(3),
    state_code char(2),
    state_name varchar(35),
    iso3166_level_name varchar(25)
  );

COMMIT;
```

sqitch reverts and verifies too

```
$ cat pg/revert/states.sql
-- Revert states

BEGIN;

    DROP TABLE states;

COMMIT;

$ cat pg/verify/states.sql
-- Verify states

BEGIN;

    SELECT id, cdh_country_code, cdh_state_code, country_code,
        state_code, state_name, iso3166_level_name
    FROM states
    WHERE false;

ROLLBACK;
```

sqitch verify output

David Wheeler seemed to like the idea of converting the verify output into TAP, but for the moment, here is what we get:

```
$ perl -CAS -I/opt/local/sqitch/lib/perl5 /usr/bin/sqitch verify
Verifying tfc-dev
  * list_types ................ ok
  * prospects ................. ok
  * states .................... ok
  * populate_states ........... ok
  * campus_types .............. ok
  * populate_campus_types ..... ok
  * campus .................... ok
  * populate_campus ........... ok
  * campus_unsupported ........ ok
  * students @v0.08 ........... ok
Verify successful
```

Tatsuhiko Miyagawa's carton manages perl dependencies

```
This is how my current projecct uses carton, . . .

script/deploy_app.sh --

function deploy_version_of_code () {

  GIT=/usr/bin/git
  # /bin/echo 'used to pull in the latest version, now let vcsrepo handle that'
  /usr/bin/sudo -u hesco $GIT pull --tags origin master

  MOST_RECENT_TAG=`/usr/bin/sudo -u hesco $GIT tag | /usr/bin/tail -n1`
  MOST_RECENT_VERSION=`/usr/bin/sudo -u hesco $GIT log --oneline | sed "s,\ .*$,," | /usr/bin/head -n1`

  if [[ $DEPLOY_VERSION == '' ]]
  then
        DEPLOY_VERSION=$MOST_RECENT_VERSION
  elif [[ $DEPLOY_VERSION == 'stable' ]]
  then
        DEPLOY_VERSION=$MOST_RECENT_TAG
  fi

  # /bin/echo "vcsrepo should have deployed the version specified in the manifest "
  # /bin/echo "but otherwise we are running git reset $DEPLOY_VERSION "
  /usr/bin/sudo -u hesco $GIT reset $DEPLOY_VERSION
  # /bin/echo "Deploying missing dependencies for app, version: $DEPLOY_VERSION "
  /bin/chown hesco: -R local/
  /usr/bin/sudo -u hesco /usr/bin/perl -I/opt/local/carton/lib/perl5 /usr/bin/carton install

  export DEPLOY_VERSION="$DEPLOY_VERSION"
  return
}
```

Jeffrey Ryan Thalhammer's Perl Critic – static code analysis

```
Our CI server gives us reports which look like this --

POLICY THEME: t/static_code_analysis/complexity.sh

ok 1         lib/App.pm source OK
ok 2         lib/App/DB.pm source OK
ok 3         lib/App/Email.pm source OK
ok 4         lib/App/Example.pm source OK
ok 5         lib/App/LandingPages.pm source OK
ok 6         lib/App/Metrics.pm source OK
```

Adding static code analysis to your test suite

The test suite includes a directory of files which look like this:

```
$ cat t/static_code_analysis/complexity.sh

/opt/local/critic/bin/perlcritic \
    --theme='complexity' \
    --verbose '%p: %m at line %l\n%f\n%r\n' \
    lib/ script/ t/

$ ls t/static_code_analysis/
bugs.sh         maintenance.sh  performance.sh      roles.sh     unicode.sh
complexity.sh   moose.sh        portability.sh     security.sh
core.sh.off     pbp.sh.off      readability.sh.off  tests.sh
```

Deploy Perl Critic without immediately fixing your code

```
t/static_code_analysis/maintenance.sh --

/opt/local/critic/bin/perlcritic \
    --theme='maintenance' \
    --verbose '%p: %m at line %l\n%f\n%r\n' \
    --exclude=RequireUseStrict \
    --exclude=RequireExplicitPackage \
    --exclude=RequirePodSections \
    --exclude=RequirePodLinksIncludeText \
    --exclude=RequireExtendedFormatting \
    --exclude=ProhibitUselessNoCritic \
    lib/ script/

/opt/local/critic/bin/perlcritic \
    --theme='maintenance' \
    --verbose '%p: %m at line %l\n%f\n%r\n' \
    --exclude=RequireUseStrict \
    --exclude=RequireExplicitPackage \
    --exclude=RequirePodSections \
    --exclude=RequirePodLinksIncludeText \
    --exclude=RequireExtendedFormatting \
    --exclude=ProhibitMagicNumbers \
    t/
```

Pinto – curate
a repository of Perl modules

Devel::Cover
measuring test coverage

Devel::hdb – Perl debugger as a web page and REST service

Plenv – perl binary manager

Test.pp

- Test::Most
  - Test::More
  - Test::Warn
  - Test::Deep
  - Test::Difference
  - Test::Exception
- Test::Perl::Critic
- Test::Perl::Critic::Progressive
- DB::Skip

Mojolicious –
Real-time web framework

Catalyst – Elegant MVC Web Application Framework

- Catalyst::Runtime
- Catalyst::Devel