



Shipwright:

Application Distribution Simplified

Kevin Falcone
falcone@bestpractical.com
jibsheet

- Best Practical Solutions
 - Request Tracker
 - Hiveminder
 - many other things
 - <http://github.com/bestpractical>
 - (and some other things in a public svn, sorry, we're busy)

In the beginning

Installing software is hard

Friday, July 29, 11

admit it, you've all had some trouble at some point getting some package to build on solaris x86 or had to google up the old version of a library to install an older app

In the beginning

Installing Request Tracker is hard

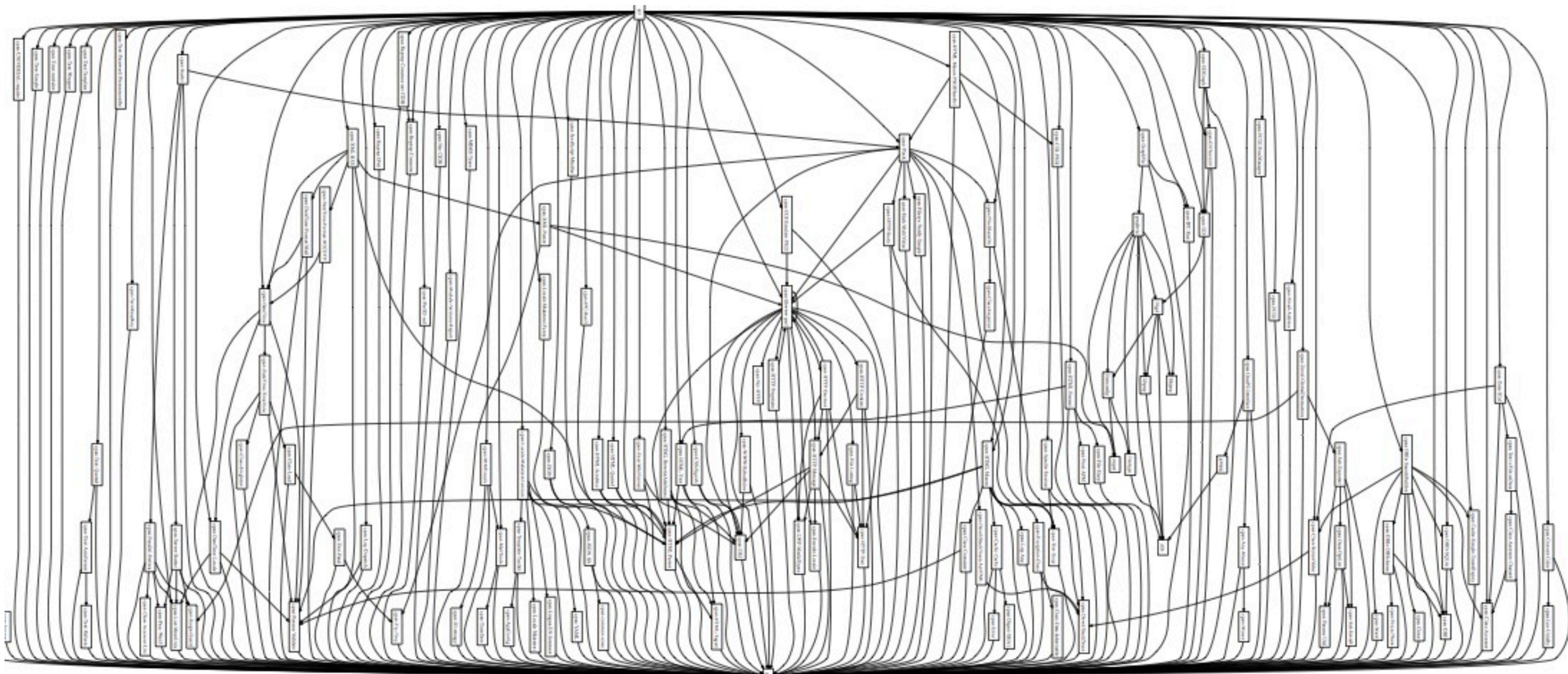
Friday, July 29, 11

I spend a lot of time on `rt-user` and `#rt` and in paid gigs helping people make it work in their systems.
I really just want this to be easy so we can move to the interesting part

In the beginning

At least 95 CPAN Dependencies

All of their dependencies



Friday, July 29, 11

Good luck reading it. But shipwright generates that as dot so you can graphviz a pdf and actually stare at it and see where all your deps are coming from.

Even with the larger resolution provided by oscon....

In the beginning

Your webserver

Apache/lighttpd/nginx/Starman

Your database

MySQL/Pg/Oracle

In the beginning

I haven't even talked about a mailserver

In the beginning

- libpng
- libjpeg
- readline
- zlib
- gnupg
- graphviz
- libgd
- expat
- fontconf

In the beginning

Installing RT should be easy

In the beginning

You just want to use your linux package repos

In the beginning

Most distributions

Package most of the dependencies

Most of the time

In the beginning

Some distributions

Package some of the dependencies

Some of the time

In the beginning

We should be able to hand you a magic box
all of the deps built for your system
all of the deps to build for your system

In the beginning

Should be relocatable
all scripts are wrapped

In the beginning

Should be installable on multiple servers

In the beginning

Should be installable on multiple OSes

In the beginning

Should be versioned

the deps should be versioned

reproducible builds

Friday, July 29, 11

I should be able to yank the RT 4.0.1 vessel out of the system and hand it off to a user who really needs it
Why do you care about versioning deps? There was a bug reported against a recent RT that turned out to be one of our MIME parsing modules changing what a mime word was and causing russian subjects that wrap in the SMTP line length to decode incorrectly. Easiest solution was downgrading a cpan module, but that sucks for end users.

In the beginning

Should be verifiable

Friday, July 29, 11

run the same vessel on dev and production
since libraries/deps are all in the vessel, you shouldn't run into those hilarious bugs where production is .01 versions behind in some obscure dep
if you're super paranoid, we could shove perl in there for you

In the beginning

Should store as many or as few deps as desired
including libc is discouraged

In the beginning

should compute a full dependency chain
no more CPAN install loops

In the beginning

automated builds of all packages

In the beginning

My boss wrote a script
it was pretty horrifying to look at
but it worked

In the beginning

My boss wrote a script
it was pretty horrifying to look at
but it "worked"

In the beginning

My colleague sunnavy took it away from him

Now we have Shipwright

In the beginning

We're pretty bad at marketing

blogged about this back in February 2008

Quick Real World Example

Prophet

SD (Simple Defects)

120 perl module dependencies

many of which are to support trac/redmine/etc

Quick Real World Example

```
curl fsck.com/sd | perl
```

\$ curl fsck.com/sd | perl
building cpan-Scalar-List-Utills
building cpan-ExtUtills-MakeMaker
building cpan-String-BufferStack
building cpan-Class-Accessor
building cpan-Class-Data-Inheritable
building cpan-Template-Declare
building cpan-HTTP-Server-Simple
building cpan-Class-Inspector
building cpan-File-ShareDir
building cpan-Test-Simple
building cpan-DBI
building cpan-DBD-SQLite
building cpan-Term-ReadLine-Perl
building cpan-common-sense
building cpan-JSON-XS
building cpan-Digest-SHA1
building cpan-Digest-HMAC
building cpan-Digest-SHA
building cpan-Net-IP
building cpan-Net-DNS
building cpan-Net-Bonjour
building cpan-TermReadKey
building cpan-XSLoader
building cpan-Sub-Uplevel
building cpan-Test-Exception
building cpan-ExtUtills-CBuilder
building cpan-ExtUtills-ParseXS
building cpan-Devel-PPPport
building cpan-Mouse
building cpan-Any-Moose
building cpan-Config-GitLike
building cpan-YAML-Syck
building cpan-UUID-Tiny
building cpan-XML-Atom-SimpleFeed
building cpan-Compress-Raw-Zlib
building cpan-Compress-Raw-Bzip2

building cpan-IO-Compress
building cpan-URI
building cpan-HTML-Tagset
building cpan-HTML-Parser
building cpan-libwww-perl
building cpan-IPC-Run3
building cpan-Exporter-Lite
building cpan-Attribute-Handlers
building cpan-Params-Validate
building cpan-Path-Dispatcher
building cpan-Module-Pluggable
building cpan-Module-Refresh
building cpan-Time-Progress
building cpan-Carp-Assert
building cpan-Proc-InvokeEditor
building cpan-MIME-Base64-URLSafe
building cpan-JSON
building cpan-Params-Util
building cpan-Sub-Install
building cpan-Data-OptList
building cpan-Sub-Exporter
building cpan-Path-Dispatcher-Declarative
building prophet.git
building cpan-Devel-StackTrace
building cpan-Exception-Class
building cpan-Error
building cpan-RT-Client-REST
building cpan-Email-Address
building cpan-HTML-Tree
building cpan-Crypt-SSLeay
building cpan-JSON-Any
building cpan-File-Slurp
building cpan-WWW-Mechanize
building cpan-WWW-Mechanize-GZip
building cpan-Net-GitHub
building cpan-YAML
building cpan-Class-Singleton

building cpan-DateTime-TimeZone
building cpan-List-MoreUtills
building cpan-DateTime-Locale
building cpan-DateTime
building cpan-Path-Class
building cpan-Clone
building cpan-Hash-Merge
building cpan-Net-Jifty
building cpan-version
building cpan-Lingua-EN-Inflect
building cpan-Text-CSV
building cpan-Net-Trac
building cpan-XML-TreePP
building cpan-XML-FeedPP
building cpan-MIME-Types
building cpan-File-MMagic
building cpan-Test-Mock-LWP
building cpan-Net-Google-Code
building cpan-Test-Script-Run
building cpan-boolean
building cpan-DateTime-Format-Natural
building cpan-Net-Lighthouse
building cpan-pQuery
building cpan-Text-CSV-Slurp
building cpan-IO-String
building cpan-IO-All
building cpan-Parse-RecDescent
building cpan-CSS
building cpan-HTML-WikiConverter
building cpan-Set-Infinite
building cpan-DateTime-Set
building cpan-DateTime-Event-Recurrence
building cpan-DateTime-Event-ICal
building cpan-DateTime-Format-ICal
building cpan-Task-Weaken

building cpan-DateTime-Format-Strptime
building cpan-Class-Factory-Util
building cpan-DateTime-Format-Builder
building cpan-DateTime-Format-Flexible
building cpan-Test-Inter
building cpan-Date-Manip
building cpan-DateTime-Format-DateManip
building cpan-TimeDate
building cpan-DateTimeX-Easy
building cpan-DateTime-Format-DateParse
building cpan-HTML-WikiConverter-Markdown
building cpan-Net-Redmine
building sd.git
install finished, the dists are at /Users/falcone/sd

Quick Real World Example

install finished, the dists are at /Users/falcone/sd

```
$ cd sd
```

```
$ ./bin/sd help
```

```
sd 0.74 - Help Index
```

```
-----
```

```
sd help intro      - Getting started with SD
sd help search     - Searching for and displaying tickets
sd help tickets    - Working with tickets
sd help comments   - Working with ticket comments
sd help attachments - Working with ticket attachments
sd help sync       - Publishing and importing ticket databases
sd help history    - Viewing repository history
sd help environment - Environment variables which affect sd
sd help config     - Local configuration variables
sd help ticket.summary_format - Details of this config variable
sd help aliases    - Command aliases
sd help settings   - Database configuration variables
```

Running 'sd help' on a specific command should also redirect you to the proper help file.

Distributing Complex Software

Everything is contained within ~/sd

mv sd simpledefects 'just works'

copy it to another mac

contains .bundle files

Using Shipwright

Friday, July 29, 11
Let's ship something

Using Shipwright

Create a shipyard

fs, svn, git

Using Shipwright

Import a project

file, http/ftp, svn, git, cpan

we'll try to detect dependencies

Using Shipwright

add dependencies

C libraries / programs

expat / libxml2

python libraries

Using Shipwright

build your vessel

a "compiled" relocatable directory with all of your code and dependencies inside

Using Shipwright

or build a source vessel

allows it to be built on the end system

Using Shipwright

Update your source project

Update one of your dependencies

to a specific CPAN version

Using Shipwright

build a new vessel

tag your shipyard so you know what you released and when



Actually Using
Shipwright

```
$ export SHIPWRIGHT_SHIPYARD=\
  git:file:///Users/falcone/shipwright/tinydancer.git
$ shipwright create
successfully created
$ ls tinydancer.git/
HEAD      description  info      refs     config   hooks    objects
```

Friday, July 29, 11

I'm using git, but you can just store things into a filesystem backend if you like and then manually put that filesystem backend into git
This is actually what my coworker is using with the experimental RT shipwright vessel

```
$ ls tiny-dancer-example/  
MANIFEST  META.yml  Makefile  Makefile.PL  bin  inc
```

```
$ cat tiny-dancer-example/bin/server  
#!/usr/bin/perl
```

```
use Dancer;
```

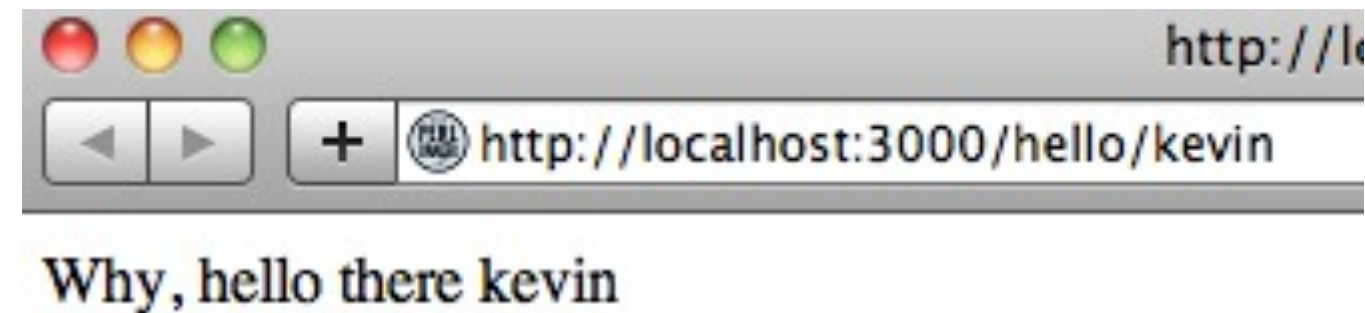
```
get '/hello/:name' => sub {  
    return "Why, hello there " . params->{name};  
};
```

```
dance;
```

Friday, July 29, 11

tiny tiny distribution with just enough code to create a bin/server in Dancer
If you've used sinatra or flask, dancer is a similar perl framework
If you wanted to learn Dancer, there was a talk on Wednesday.
make sure to explain the code

```
$ perl tiny-dancer-example/bin/server  
>> Dancer 1.3060 server 67597 listening on http://0.0.0.0:3000  
== Entering the development dance floor ...
```



```
$ shipwright import git:file:///Users/falcone/gitprojects/tiny-dancer-example
CPAN related output will be at /Users/falcone/tmp/shipwright_cpan.log
import cpan-Dancer
import cpan-MIME-Types
import cpan-HTTP-Body
import cpan-Test-Deep
import cpan-Test-Tester
import cpan-Test-NoWarnings
import cpan-HTTP-Message
import cpan-URI
import cpan-HTML-Parser
import cpan-HTML-Tagset
import cpan-HTTP-Date
import cpan-Encode-Locale
import cpan-LWP-MediaTypes
import cpan-libwww-perl
import cpan-File-Listing
import cpan-HTTP-Cookies
import cpan-HTTP-Daemon
import cpan-WWW-RobotRules
import cpan-HTTP-Negotiate
import cpan-Net-HTTP
import cpan-HTTP-Server-Simple-PSGI
import cpan-HTTP-Server-Simple
importing tiny-dancer-example
successfully imported
```

```
$ git clone tinydancer.git/  
Cloning into tinydancer...  
done.
```

```
$ ls tinydancer  
__default_builder_options inc          sources  
bin          scripts          t  etc          shipwright
```

```
$ ls tinydancer/bin  
shipwright-builder shipwright-filter shipwright-utility
```

```
$ cd tinydancer
$ ./bin/shipwright-builder --skip-test --install-base ./dancer
building cpan-MIME-Types
building cpan-Test-Tester
building cpan-Test-NoWarnings
building cpan-Test-Deep
building cpan-URI
building cpan-HTML-Tagset
building cpan-HTML-Parser
building cpan-HTTP-Date
building cpan-Encode-Locale
building cpan-LWP-MediaTypes
building cpan-HTTP-Message
building cpan-HTTP-Body
building cpan-File-Listing
building cpan-HTTP-Cookies
building cpan-HTTP-Daemon
building cpan-WWW-RobotRules
building cpan-HTTP-Negotiate
building cpan-Net-HTTP
building cpan-libwww-perl
building cpan-HTTP-Server-Simple
building cpan-HTTP-Server-Simple-PSGI
building cpan-Dancer
building tiny-dancer-example
install finished, the dists are at /Users/falcone/shipwright/tinydancer/dancer
```

Friday, July 29, 11

Chases down and builds all the cpan modules.
Puts the built vessel into the dancer dir
next slide – what's in the dir

```
$ ls dancer/
```

```
__as as bin  etc  lib man  tools
```

```
$ ls dancer/tools/
```

```
shipwright-source-bash  shipwright-source-tcsh  shipwright-utility
```

```
$ ls dancer/bin/
```

```
dancer  server
```

```
$ ls dancer/as/Darwin/
```

```
bin  lib  usr
```

Friday, July 29, 11

just a directory, tar it up or rsync it out to your linodes
things to update your shell
things to symlink into bin/lib
it installs modules into your OS, we'll touch on that later

```
$ ./dancer/bin/server
```

```
>> Dancer 1.3070 server 90109 listening on http://0.0.0.0:3000
```

```
== Entering the development dance floor ...
```

```
$ head ./dancer/bin/server
```

```
#!/bin/sh
```

```
if [ -z `which readlink` ] || [ "`which readlink | grep 'no readlink in'" ]; then
```

```
    # if we don't have readlink, we're on some pitiful platform like solaris
```

```
    test -h $0 && LINK=`ls -l $0 | awk -F\> '{print $NF}' | sed -e 's/^ //'`
```

```
else
```

```
    LINK=`readlink $0`
```

```
fi
```

```
if [ "$LINK" = "" ] || [ $LINK = '../etc/shipwright-script-wrapper' ] || [ $LINK =  
'../../etc/shipwright-script-wrapper' ]; then
```

```
    BASE=$0
```

```
    BASE_DIR=`dirname "$BASE"`
```

Another Deployment Option

```
$ ./bin/shipwright-utility --generate-tar-file dancer.pl  
successfully generated
```

```
$ head dancer.pl  
#!/usr/bin/env perl  
use strict;  
use warnings;  
use File::Spec::Functions;  
use Config;  
use Cwd;  
use File::Temp;  
my $bin_quote = $^O =~ /MSWin/ ? q{"} : q{'};  
  
...  
__DATA__  
(gzipped tar data)
```

```
$ perl dancer.pl --skip-test --install-base /tmp/deploy-tiny-dancer  
building cpan-MIME-Types
```

```
....
```

```
building cpan-HTTP-Server-Simple
```

```
building cpan-HTTP-Server-Simple-PSGI
```

```
building cpan-Dancer
```

```
building tiny-dancer-example
```

```
install finished, the dists are at /tmp/deploy-tiny-dancer
```

```
$ /tmp/deploy-tiny-dancer/bin/server
```

```
>> Dancer 1.3070 server 41501 listening on http://0.0.0.0:3000
```

```
== Entering the development dance floor ...
```

```
$ mv /tmp/deploy-tiny-dancer/ /tmp/newlocation
$ /tmp/newlocation/bin/server
>> Dancer 1.3070 server 41543 listening on http://0.0.0.0:3000
== Entering the development dance floor ...
```

Updating Your App

```
$ cd gitprojects/tiny-dancer-example/  
(coding interlude)  
$ git di HEAD^  
diff --git a/bin/server b/bin/server  
index ac5881c..84d2811 100644  
--- a/bin/server  
+++ b/bin/server  
@@ -6,4 +6,8 @@ get '/hello/:name' => sub {  
    return "Why, hello there " . params->{name};  
};  
  
+get '/goodbye/:name' => sub {  
+  return "Goodbye " . params->{name};  
+};  
+  
dance;
```

```
$ shipwright list tiny-dancer-example
```

```
tiny-dancer-example:
```

```
version:
```

```
vendor: 2a2ac9dd402d67a1d15485a65eda4212982051c4
```

```
from:
```

```
vendor: git:file:///Users/falcone/gitprojects/tiny-dancer-example
```

```
$ shipwright update tiny-dancer-example --only-sources
```

```
successfully updated
```

```
$ shipwright list tiny-dancer-example
```

```
tiny-dancer-example:
```

```
version:
```

```
vendor: f22a5ec92971e8f883e55fc4a4dff5349cc962dc
```

```
from:
```

```
vendor: git:file:///Users/falcone/gitprojects/tiny-dancer-example
```

```
$ cd ~/shipwright/tinydancer; git pull
$ git show
commit dfa1c529c6ca7d3c1b7b845e6c63c06c096f8f7 |
--- a/shipwright/version.yml
+++ b/shipwright/version.yml
@@ -44,4 +44,4 @@ cpan-WWW-RobotRules:
 cpan-libwww-perl:
   vendor: 6.02
 tiny-dancer-example:
- vendor: 2a2ac9dd402d67a1d15485a65eda4212982051c4
+ vendor: f22a5ec92971e8f883e55fc4a4dff5349cc962dc
diff --git a/sources/tiny-dancer-example/vendor/bin/server b/sources/tiny-dancer-example/vendor/bin/server
index ac5881c..84d2811 100644
--- a/sources/tiny-dancer-example/vendor/bin/server
+++ b/sources/tiny-dancer-example/vendor/bin/server
@@ -6,4 +6,8 @@ get '/hello/:name' => sub {
   return "Why, hello there " . params->{name};
 };

+get '/goodbye/:name' => sub {
+  return "Goodbye " . params->{name};
+};
+
dance;
```

```
$ ./bin/shipwright-builder --skip-test --install-base ./dancer
```

```
building cpan-MIME-Types
```

```
...
```

```
building cpan-Dancer
```

```
building tiny-dancer-example
```

```
install finished, the dists are at /Users/falcone/shipwright/tinydancer/dancer
```

```
$ ./dancer/bin/server
```

```
>> Dancer 1.3070 server 91790 listening on http://0.0.0.0:3000
```

```
== Entering the development dance floor ...
```



Other Commands

```
$ source dancer/tools/shipwright-source-bash ~/shipwright/tinydancer/dancer
$ cd
$ which server
/Users/falcone/shipwright/tinydancer/dancer/bin/server
$ server
>> Dancer 1.3070 server 91838 listening on http://0.0.0.0:3000
== Entering the development dance floor ...
```

Friday, July 29, 11

in case you want to stop typing full paths, this puts bin in your path

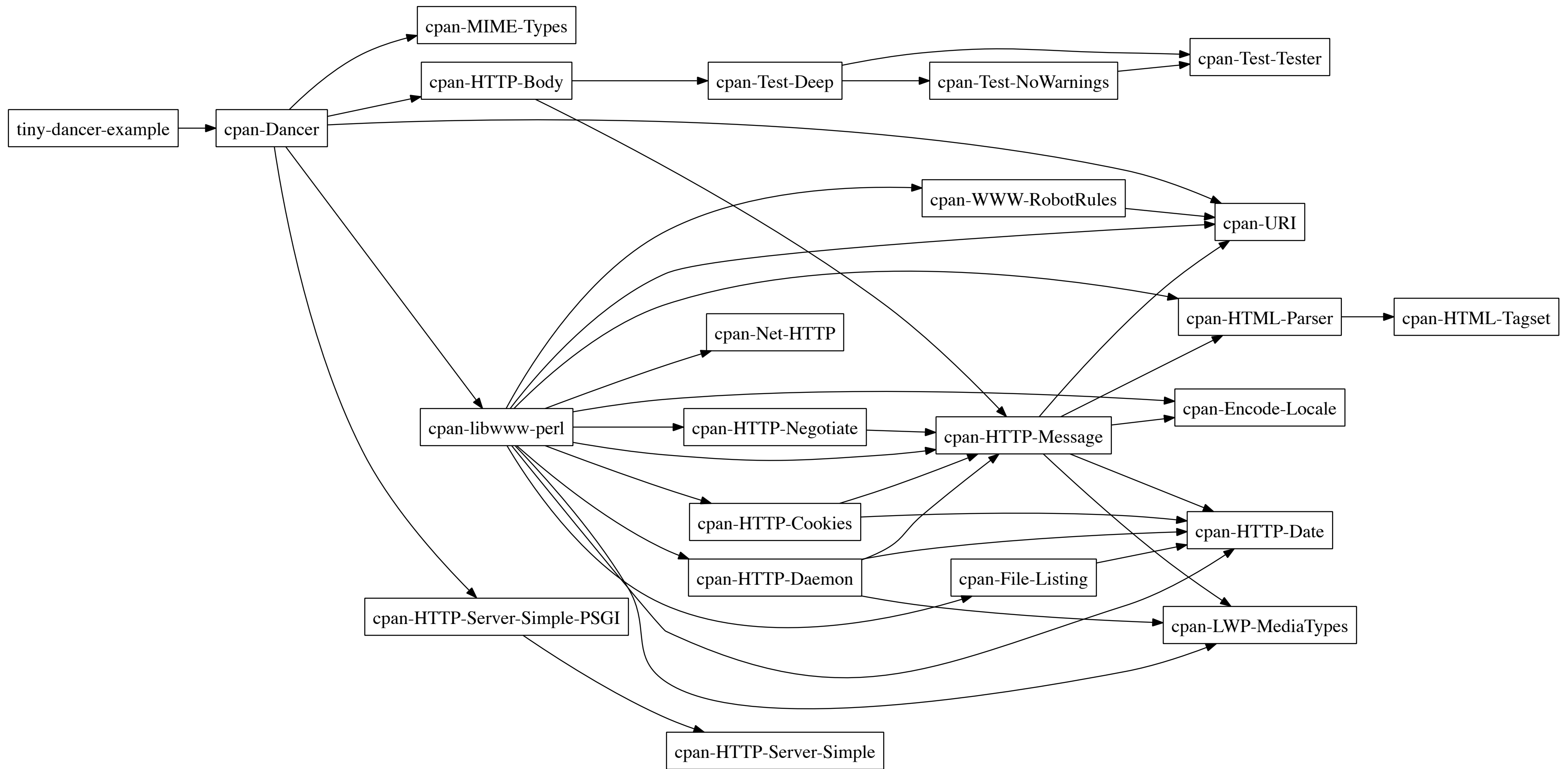
We have tcsh if you like that sort of thing

Sorry, no ksh support

I assume zsh is sentient enough to find it for you without us doing anything

There's another tool in there if you just want symlinks to the right place, not env frobbing

\$ shipwright maintain --graph-deps | dot -Tpdf > deps.pdf



Customizing Builds

```
import perl-5.14.1.tar.gz
```

```
shipwright import http://www.cpan.org/src/5.0/perl-5.14.1.tar.gz
```

```
update --add-dep perl (each of your cpan deps)
```

```
ls -d cpan-* | xargs -n1 shipwright update --add-deps perl
```

```
shipwright maintain --update-refs
```

```
shipwright maintain --update-order
```

Friday, July 29, 11

I keep hoping to find the magic add a base perl dep flag, I'll make a branch

cpan-* is in the dists dir in your shipyard

Let's see what those last two commands actually do

Customizing Builds

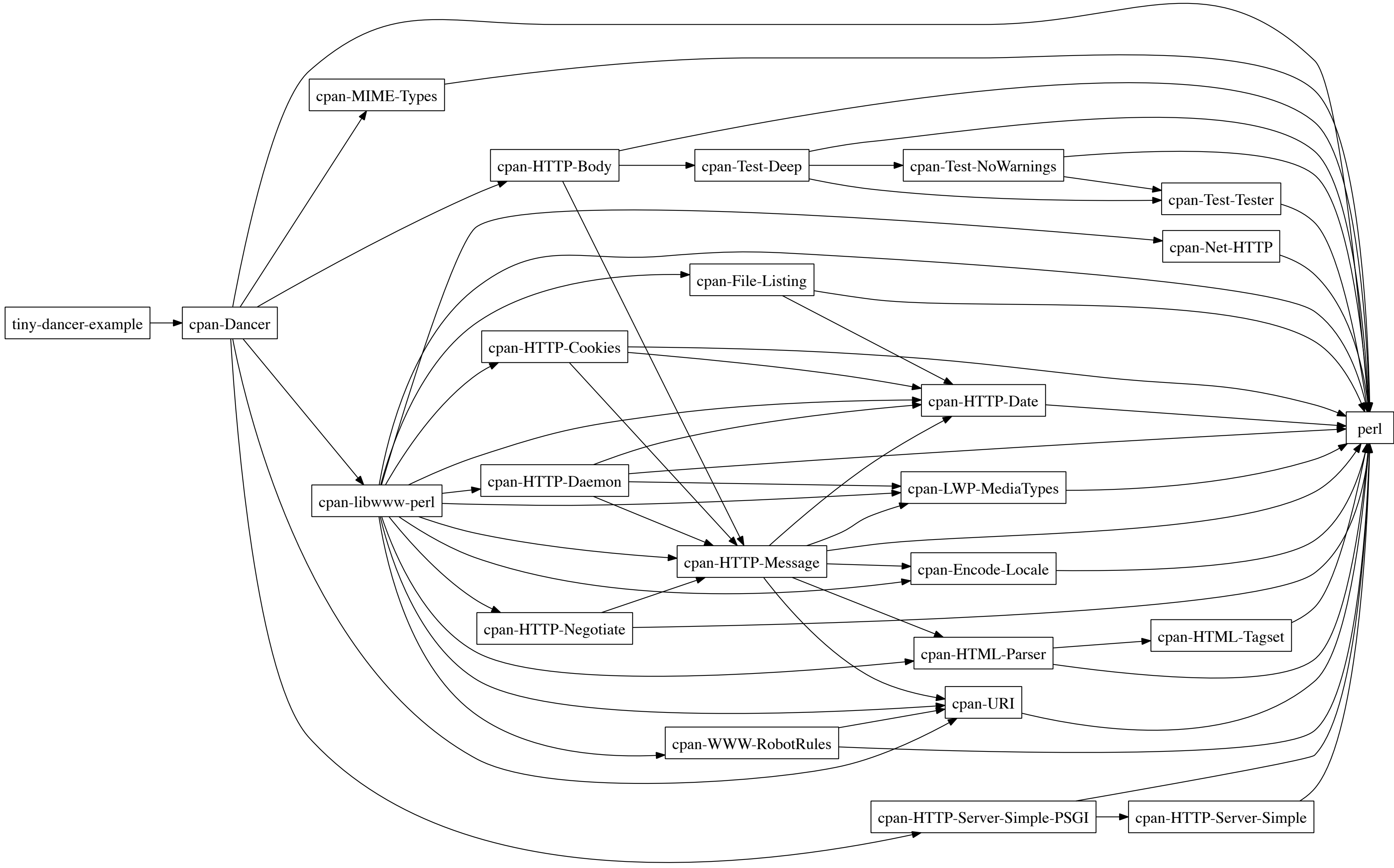
```
$ head shipwright/order.yml
```

```
---
```

- perl
- cpan-Encode-Locale
- cpan-HTML-Tagset
- cpan-HTTP-Date
- cpan-HTTP-Server-Simple
- cpan-HTTP-Server-Simple-PSGI
- cpan-LWP-MediaTypes
- cpan-MIME-Types
- cpan-Net-HTTP

```
$ tail shipwright/refs.yml
```

```
cpan-MIME-Types: 1  
cpan-Net-HTTP: 1  
cpan-Test-Deep: 1  
cpan-Test-NoWarnings: 1  
cpan-Test-Tester: 2  
cpan-URI: 4  
cpan-WWW-RobotRules: 1  
cpan-libwww-perl: 1  
perl: 22  
tiny-dancer-example: 0
```



Customizing Builds

`scripts/DISTRO/build`

`scripts/DISTRO/build.pl`

Customizing Builds

control

configure

make

install

clean

Customizing Builds

`update --only-sources`

avoids rewriting build on you

```
$ git di HEAD^
diff --git a/scripts/perl/build b/scripts/perl/build
index 75652fd..9c84dc9 |00644
--- a/scripts/perl/build
+++ b/scripts/perl/build
@@ -1,4 +1,4 @@
-configure: ./configure --prefix=%%INSTALL_BASE%%
+configure: sh Configure -de -Dprefix=%%INSTALL_BASE%%
make: %%MAKE%%
install: %%MAKE%% install
clean: %%MAKE%% clean
```

```
$ ./bin/shipwright-utility --generate-tar-file dancer-with-perl.pl  
successfully generated
```

```
$ perl dancer-with-perl.pl --skip-test --install-base ~/standalone.dancer  
building perl  
building cpan-Encode-Locale  
...  
building cpan-Dancer  
building tiny-dancer-example  
install finished, the dists are at ~/standalone.dancer
```

```
$ ./standalone.dancer/bin/perl -v
```

This is perl 5, version 14, subversion 1 (v5.14.1) built for i686-linux

```
falcone@am:~/shipwright$ ./standalone.dancer/bin/server  
>> Dancer 1.3070 server 9416 listening on http://0.0.0.0:3000  
== Entering the development dance floor ...
```

Friday, July 29, 11

You'll see that it contains a perl, and that it runs the server.
Just like everything else in bin/, perl is wrapped

Other dep frobbing

`cpan-XML-Parser`

`cpan-XML-LibXML`



Other Customizations

Other Customizations

multiple arches

flags

Flags

```
$ shipwright flags cpan-DBD-mysql --set mysql  
successfully set flags  
flags of cpan-DBD-mysql are mysql
```

```
$ shipwright flags cpan-DBD-pg --set postgresql  
successfully set flags  
flags of cpan-DBD-pg are postgresql
```

```
$ shipwright flags --mandatory --set mysql,postgresql database  
successfully set mandatory flags  
mandatory flags of database are mysql, postgresql
```

Flags

```
scripts/RT/build.pl
```

```
...
```

```
if ( $args{flags}{mysql} ) {  
    $db_type = 'mysql';
```

```
...
```

```
./configure.ac --with-db-type=$db_type
```

Multiple Arch Builds

```
./bin/shipwright-builder --skip-test --install-base /tmp/standalone.dancer --as Ubuntu  
building perl
```

....

```
building cpan-Dancer
```

```
building tiny-dancer-example
```

```
install finished, the dists are at /tmp/standalone.dancer
```

copy shipyard and standalone.dancer to another machine and rerun

```
$ ls standalone.dancer/as/
```

```
Debian  Ubuntu
```

```
$ cd standalone-dancer
```

```
$ ./tools/shipwright-utility --switch Debian
```

```
successfully switched to Debian.
```

```
$ ./bin/perl -V | grep osname
```

```
osname=linux, osvers=2.6.28-1-686-bigmem, archname=i686-linux
```

```
$ ./tools/shipwright-utility --switch Ubuntu
```

```
successfully switched to Ubuntu.
```

```
$ ./bin/perl -V | grep osname
```

```
osname=linux, osvers=2.6.31-23-server, archname=x86_64-linux
```

```
$ ./bin/server
```

```
>> Dancer 1.3070 server 11744 listening on http://0.0.0.0:3000
```

```
== Entering the development dance floor ...
```

```
$ du -sh standalone-dancer  
115M  standalone-dancer
```

```
$ ./bin/shipwright-filter --squeeze  
squeeze finished.
```

```
$ ./bin/shipwright-filter --remove-pod  
removing pod finished.
```

```
$ du -sh standalone-dancer  
82M   standalone-dancer
```

The Stern

<http://search.cpan.org/dist/Shipwright>

<https://github.com/bestpractical/shipwright>

<http://jibsheet.com/talks/oscon2011.pdf>

Wrapping non-perl

- C
 - we already set much of the relevant environment
- Python
 - more interesting

```
$ shipwright import git:file:///.../gitprojects/flask-tiny
importing flask-tiny
$ shipwright import file:///.../gitprojects/flask-tiny/Flask-0.7.2.tar.gz
importing Flask
$ shipwright import file:///.../gitprojects/flask-tiny/Jinja2-2.5.5.tar.gz
importing Jinja2
$ shipwright import file:///.../gitprojects/flask-tiny/Werkzeug-0.7.tar.gz
importing Werkzeug
```

```
$ shipwright update --add-deps Werkzeug Flask
```

```
successfully updated
```

```
$ shipwright update --add-deps Jinja2 Flask
```

```
successfully updated
```

```
$ shipwright update --add-deps Flask flask-tiny
```

```
successfully updated
```

```
$ shipwright maintain --update-refs
```

```
successfully updated refs
```

```
$ shipwright maintain --update-order
```

```
successfully updated order
```

A python app

hack around easy_install in build

update the script wrapper for which version of python you're using so that PYTHONPATH is set

Friday, July 29, 11

this involves not just setting --prefix, but using the --no-really-just-do-what-I-said flags otherwise it won't install into a path that doesn't exist
I thought ExtUtils::MakeMaker was hairy

```
$ ./bin/shipwright-builder --skip-test --install-base ~/shipwright/flask/flask
building Jinja2
building Werkzeug
building Flask
building flask-tiny
install finished, the dists are at /Users/falcone/shipwright/flask/flask
$ ./flask/bin/server
* Running on http://127.0.0.1:5000/
```

Off the Stern

<http://search.cpan.org/dist/Shipwright>

<https://github.com/bestpractical/shipwright>

<http://jibsheets.com/talks/oscon2011.pdf>