Docker & Perl

What is docker?

It’s probably the most popular containerization technology on Linux these days

It’s somewhere between chroot jails and virtual machines.

Effectively lightweight virtual machines running on the same kernel as the host machine.

Do a ‘`ps ax`’ on the host machine and you can see all the processes from the containers too.

Building Docker containers

Our typical layout includes an empty volume.

A new version of a module?

Docker’s cache is really handy for speed. It can be tricky when you have a slight change you want built in.

You can build with --no-cache, but that’s very slow.

You can also tweak the line that needs to be executed. Either add a comment or add a specific version number for cpanm ‘`cpanm Module@0.03`’

Or you can run another command at the end of the dockerfile that upgrades that module.

Or create a new container based on the one that wants to be upgraded.

Docker-compose

This allows you to ‘compose’ multiple machines together.

Docker containers are ideally each single purpose machines.

- Database server
- Memcache server
- Application server
- Nginx frontend proxy

Docker-compose allows you to bring up a cluster of these together, linked up correctly.

Docker-compose

Docker compose hooks up name resolution automatically.

It also allows you to pass in environment variables.

Volumes can also be hooked up. These are mounts between the containers, or the host machine.

Environment variables and configuration

Your apps are a lot easier to manage with docker if you can alter config using Environment variables.

That way you can keep core passwords/connection details outside of the docker images and the same images for test/staging/live without any modification.

With Catalyst we use Catalyst::Plugin::ConfigLoader::Environment for example.

Doing development

Our typical setup mounts our code into that empty volume we setup.

It’s not perfect,

- code deleted will still appear courtesy of the original module install
- New dependencies won’t be installed

Allows us to work quickly with almost the box we will deploy in the end

We use a script to load all the modules found in the volume to PERL5LIB

Running your code - TIMTOWTDI

You can restart containers within a docker-compose cluster simply

```
docker-compose restart site
```

Or you can run a whole new container, this will spin up with a new IP

```
docker-compose run site
```

Or you can run a new process within the existing container

```
docker exec -it test_site_1 plackup /opt/app/site.psgi
```

Composing compose files

Docker-compose can pull in from multiple files and merge the configuration from them. It’s not perfect, but you can do lots of additive things like changing commands or adding settings.

You can have a docker-compose.dbic_trace.yml for example adds environment variables.

Or an docker-compose.strace.yml that starts the app up with strace.

Testing

Where I’ve needed a test container that runs Postgres on the same machine as the perl I’ve smashed containers together using a combination of volumes and dirty tricks.

Some tests need a full application stack. That’s easy as it’s probably very close to your standard docker-compose stack so you can either add it into that, or clone that.

Logging

Docker expects you to output your logs to stdout/err.

You can then ask Docker to export those logs to syslog or some other source

Buffering can be an issue with perl apps

This is generally perl level buffering

```
select( ( select(\*STDERR), $|=1 )[0] );
select( ( select(\*STDOUT), $|=1 )[0] );
```

Docker-compose gotchas

It’s handling of volumes can be confusing.

It’s designed to not catch you out.

Guess what happens.

Volumes mounted to the host machine are nice and predictable.

Mounted to another container is probably fairly predictable.

Not mounted is where it gets fun.

The ever running command

Using docker-compose run to run a single command when the container is set to restart-always can be fun.

Using --rm prevents the silliness, and tidies up afterwards.

Summary

Docker and docker-compose can be a bit of a head banger at first, but it’s well worth the investment.

It allows you to spin up complex infrastructure cheaply.

Multiple copies of the same thing are easy.