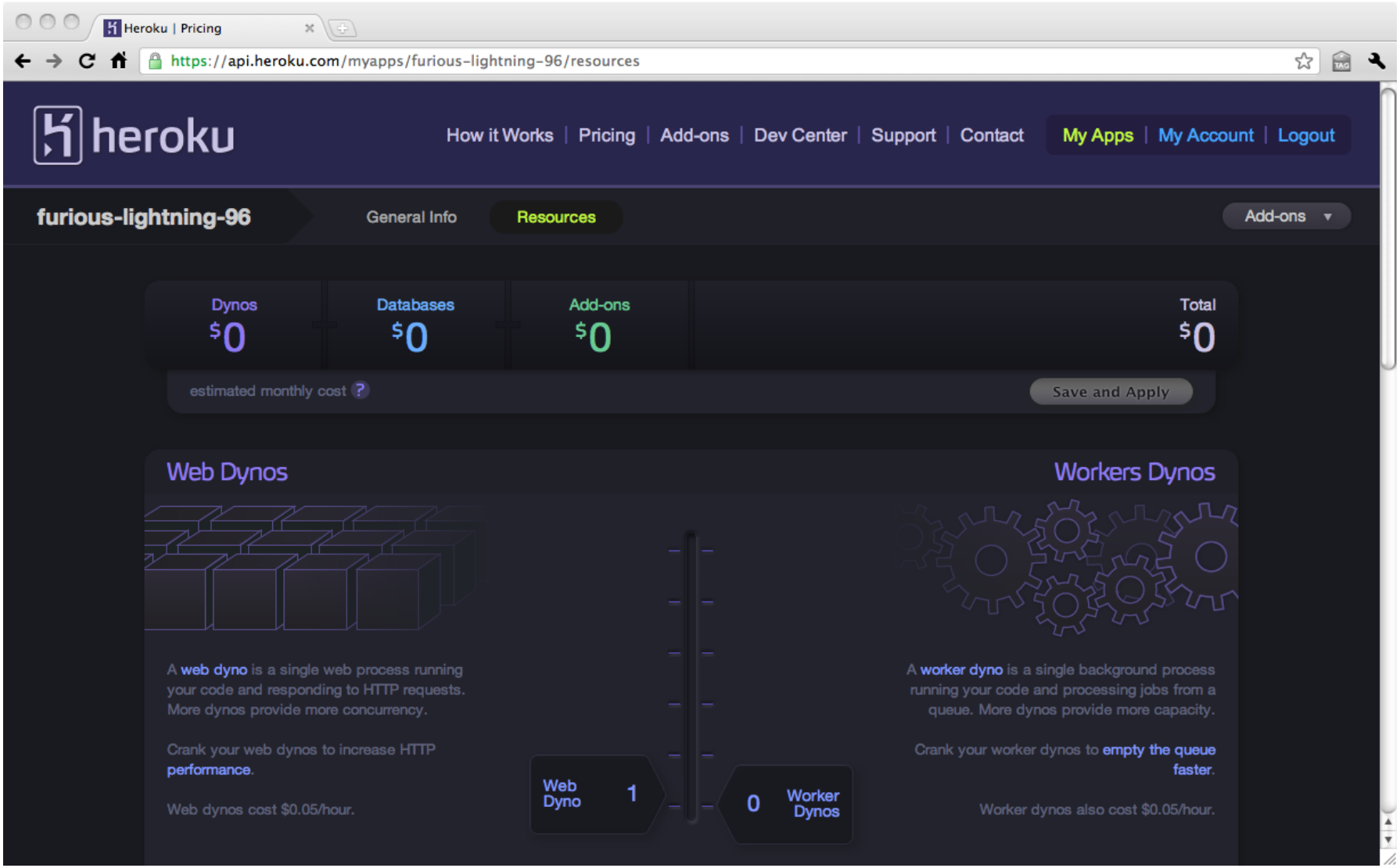
# Dancer in the Cloud

Using Perl micro web frameworks
on a PaaS

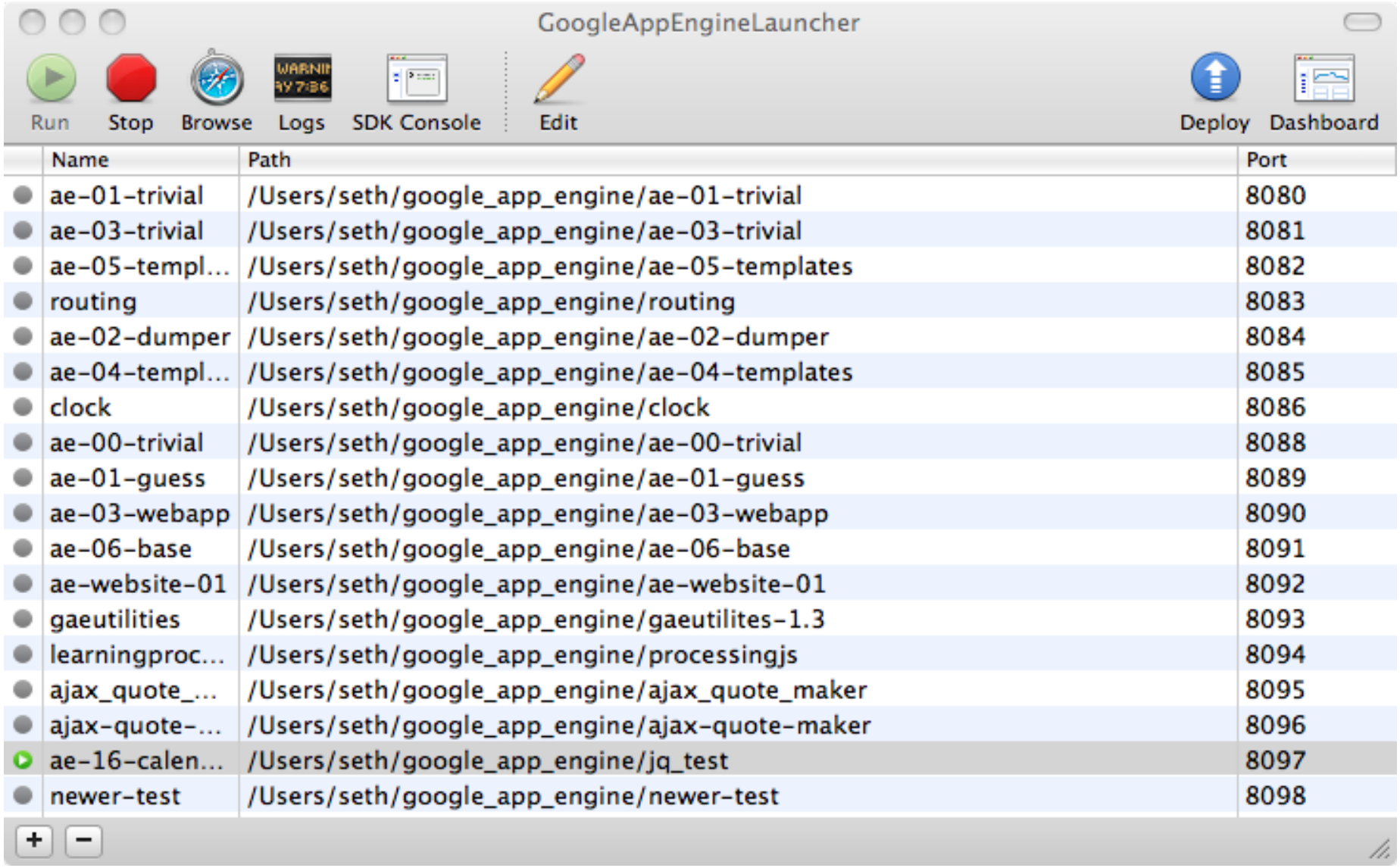
# PaaS (Platform as a Service) examples

Heroku (Ruby on Rails, nodejs) built on Amazon



# PaaS (Platform as a Service) examples

Google App Engine (Java, Python) on Google infrastructure




| Name | Path | Port |
|---|---|---|
| ae-01-trivial | /Users/seth/google_app_engine/ae-01-trivial | 8080 |
| ae-03-trivial | /Users/seth/google_app_engine/ae-03-trivial | 8081 |
| ae-05-templ... | /Users/seth/google_app_engine/ae-05-templates | 8082 |
| routing | /Users/seth/google_app_engine/routing | 8083 |
| ae-02-dumper | /Users/seth/google_app_engine/ae-02-dumper | 8084 |
| ae-04-templ... | /Users/seth/google_app_engine/ae-04-templates | 8085 |
| clock | /Users/seth/google_app_engine/clock | 8086 |
| ae-00-trivial | /Users/seth/google_app_engine/ae-00-trivial | 8088 |
| ae-01-guess | /Users/seth/google_app_engine/ae-01-guess | 8089 |
| ae-03-webapp | /Users/seth/google_app_engine/ae-03-webapp | 8090 |
| ae-06-base | /Users/seth/google_app_engine/ae-06-base | 8091 |
| ae-website-01 | /Users/seth/google_app_engine/ae-website-01 | 8092 |
| gaeutilities | /Users/seth/google_app_engine/gaeutilites-1.3 | 8093 |
| learningproc... | /Users/seth/google_app_engine/processingjs | 8094 |
| ajax_quote_... | /Users/seth/google_app_engine/ajax_quote_maker | 8095 |
| ajax-quote-... | /Users/seth/google_app_engine/ajax-quote-maker | 8096 |
| ae-16-calen... | /Users/seth/google_app_engine/jq_test | 8097 |
| newer-test | /Users/seth/google_app_engine/newer-test | 8098 |

# Perl PaaS providers (all Beta, developer preview only)

**Stackato** (ActiveState) http://www.activestate.com/cloud

* Lengthy sign up process (for me anyway), I didn't finish the questionnaire.

**Phenona** (Recently acquired by ActiveState) http://www.activestate.com/press-releases/activestate-acquires-phenona-perl-cloud-company

* I emailed them a month or so ago for an account, never heard back

**DotCloud** http://dotcloud.com

* Emailed for a beta account and got a response in a couple of days. Account and deployments are free (for now)

## Winner: DotCloud

DotCloud

https://www.dotcloud.com/manage

dotcloud

Home | Blog | Documentation | Support | Settings | Sign out

You can deploy your first application now. →
Use the cheatsheet to install the command-line dotcloud tool and get started.
Or read the detailed tutorial to learn more.
Questions? Feedback? Problems? Don't hesitate to contact us!

## Cheatsheet

1. Install dotcloud
   `$ sudo easy_install dotcloud`
2. Create your application "ramen"
   `$ dotcloud create ramen`
3. See the list of available services
   `$ dotcloud deploy -h`
4. Deploy your service named "ramen.www"
   `$ dotcloud deploy --type python ramen.www`
5. List your applications
   `$ dotcloud list`
6. Push your code on the service "ramen.www"
   `$ dotcloud push ramen.www ~/work/ramen/www`
7. Run a remote command
   `$ dotcloud run ramen.www -- ls -l`
8. Open an ssh session
   `$ dotcloud ssh ramen.www`
9. Inspect the logs (Ctrl+C to stop)
   `$ dotcloud logs ramen.www`
10. Read the docs, starting with the detailed tutorial

# Perl 'micro' Web Frameworks

## Dancer

http://perldancer.org/

## Mojolicious

.
http://www.mojolicious.org/

I found both of these frameworks to be nice to work with. Mojolicious is a little different in that it's 'self contained' (minimal dependancies), but Dancer really doesn't use CPAN all that much either. They're both good, my choice of using Dancer is mostly arbitrary.

# Goals

* Build and deploy a silly example app (Solitary Pictionary!) using dancer and dotcloud.

* Highlight potential 'gotchas' in the development/deployment process.

* Through slavishly reproducing the steps from development to deployment I'm hoping to demonstrate the ease and speed of using these tools.

*Standard Disclaimer: Unlike Dancer, my code is not smart, error handling is non-existent, insecure, etc. You've been informed.

# Installing Dancer

Installing Dancer is of course easy with CPAN. I tend to use cpan minus for installs, but it's a matter of preference.

Dancer includes much more than I'll be showing here. There is great documentation for all the features on CPAN:

http://search.cpan.org/~sukria/Dancer-1.3060/lib/Dancer.pm

```
$ cpanm Dancer
[ ... ]
# Dancer dependencies installed here, there aren't too many
$
```

# Building a Dancer App (locally)

```
$ dancer -a NewApp

+ NewApp
+ NewApp/bin
+ NewApp/bin/app.pl
+ NewApp/config.yml
+ NewApp/environments
+ NewApp/environments/development.yml
+ NewApp/environments/production.yml
[ ... ]
+ NewApp/t/002_index_route.t
+ NewApp/t/001_base.t
+ NewApp/Makefile.PL
$
```

# (Basic) Anatomy of a Dancer App

```
NewApp$ tree
.
├── Makefile.PL
├── bin
│   └── app.pl
├── config.yml
├── environments
│   ├── development.yml
│   └── production.yml
├── lib
│   └── NewApp.pm
├── public
│   ├── css
│   │   ├── error.css
│   │   └── style.css
│   ├── dispatch.cgi
│   ├── dispatch.fcgi
│   ├── favicon.ico
│   └── javascripts
│       └── jquery.js
├── t
│   ├── 001_base.t
│   └── 002_index_route.t
└── views
    ├── index.tt
    └── layouts
        └── main.tt

10 directories, 22 files
NewApp$
```

# Running a Dancer App (locally)

```
NewApp$ bin/app.pl
[38952]  core @0.000012> loading Dancer::Handler::Standalone handler in
/Library/Perl/5.10.0/Dancer/Handler.pm l. 40
[38952]  core @0.000212> loading handler 'Dancer::Handler::Standalone' in
/Library/Perl/5.10.0/Dancer.pm l. 351
>> Dancer 1.3050 server 38952 listening on http://0.0.0.0:3000
== Entering the development dance floor ...
```

NewApp

0.0.0.0:3000

# Perl is dancing

You've joined the dance floor!

## Getting started

Here's how to get dancing:

About your application's environment

1. Tune your application

   Your application is configured via a global configuration file, `config.yml` and an "environment" configuration file, `environments/development.yml`. Edit those files if you want to change the settings of your application.

2. Add your own routes

   The default route that displays this page can be removed, it's just here to help you get started. The template used to generate this content is located in `views/index.tt`. You can add some routes to `lib//NewApp.pm`.

3. Enjoy web development again

   Once you've made your changes, restart your standalone server (bin/app.pl) and you're ready to test your web application.

### Join the community

PerlDancer
Official Twitter
GitHub Community

### Browse the documentation

Introduction
Cookbook
Deployment Guide
Tutorial

### Your application's environment

Location: `/Users/seth/perl/dancers/NewApp`
Template engine: `simple`
Logger: `console`
Environment: `development`

Powered by Dancer 1.3050

# Local MySQL (create and seed db)

```
$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
[ ... ]
mysql> create database words;
Query OK, 1 row affected (0.01 sec)

mysql> exit
$ mysql -u root words < words.sql
$
```

# Local MySQL (words.sql)

```
$ vim words.sql
----------------------------
CREATE TABLE `word` (
  `id` int(16) NOT NULL AUTO_INCREMENT,
  `word` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=109548 DEFAULT CHARSET=latin1;
----------------------------
INSERT INTO `word` VALUES (1,'a'),(2,'aah'),(3,'aahed'),(4,'aahing'),(5,'aahs'),
(6,'aardvark'),(7,'aardvarks'),(8,'aardwolf')
[ ... ]
```

# Local: Dancer MySQL plugin

http://search.cpan.org/~bigpresh/Dancer-Plugin-Database-1.40/lib/Dancer/Plugin/Database.pm

```
$ cpanm Dancer::Plugin::Database
```

# Local: Dancer config files (adding MySQL)

```
NewApp$ vim environments/development.yml

plugins:
    Database:
        driver: 'mysql'
        database: 'words'
        host: 'localhost'
        username: 'root'
        password: ''
        connection_check_threshold: 10
        dbi_params:
            RaiseError: 1
            AutoCommit: 1
            on_connect_do: ["SET NAMES 'utf8'", "SET CHARACTER SET 'utf8'" ]
            log_queries: 1
```

# Local:Original Dancer Code Boilerplate

```
NewApp$ vim lib/NewApp.pm

package NewApp;
use Dancer ':syntax';

our $VERSION = '0.1';

get '/' => sub {
    template 'index';
};

true;
```

# Local: Simple 'picture' app (using Yahoo image API)

```
NewApp$ vim lib/NewApp.pm
```

```
package NewApp;
use Dancer ':syntax';
use Dancer::Plugin::Database;
use Dancer::Template::TemplateToolkit;
use LWP::UserAgent;
use XML::Simple;
use Data::Dumper;


my $yahoo_api_id = 'API_KEY_HERE';

our $VERSION = '0.1';

get '/test' => sub {
    print "ok";
};
```

# Local: Simple 'picture' app (using Yahoo image API)

```
NewApp$ vim lib/NewApp.pm (continued)
```

```
get '/word/:word' => sub {
    my $word = params->{word};
    my $urls = call_yahoo_api($word, $yahoo_api_id);
    template 'index', { word => $word, urls => $urls };
};

get '/' => sub {
    my $sth = database->prepare('select word from word order by rand() limit 1');
    $sth->execute;
    my $word = $sth->fetchrow;
    my $urls = call_yahoo_api($word, $yahoo_api_id);
    my $vars = { word => $word, urls => $urls };
    template 'index', $vars ;

};
```

# Local: Simple 'picture' app (using Yahoo image API)

```
NewApp$ vim lib/NewApp.pm (continued)
```

```
sub call_yahoo_api {

    my ($word, $yahoo_api_id) = @_;

    my $url =            "http://search.yahooapis.
com/ImageSearchService/V1/imageSearch?
appid=$yahoo_api_id&query=$word&results=20";

    my $browser = LWP::UserAgent->new;
    my $response = $browser->get($url);
    $response->is_success or die "no lwp $url: ",$response->message,"\n";
    my $links = get_image_links($response->content);
    return $links;
}
```

# Local: Simple 'picture' app (using Yahoo image API)

```
NewApp$ vim lib/NewApp.pm (continued)
```

```
sub get_image_links {
    my ($content) = @_;
    my $xml = new XML::Simple;
    my $data = $xml->XMLin( $content );

    my @links;
    for my $array ( @$data{'Result'} ) {
        for my $ref ( @$array ) {
            my $link = $$ref{'Url'};
            push(@links,$link);
        }
    }
    return \@links;
}

true;
```

# Local: Simple 'picture' app (using Yahoo image API)

```
NewApp$ vim views/index.tt
```

```
Scroll down for your word...
<br />
<% FOREACH url IN urls %>
   <a href="<% url %>"><img src="<% url %>" /></a><br />
<% END %>
Your word is: <a href="http://www.google.com/dictionary?langpair=en|en&q=<%
word %>&hl=en&aq=f"><% word %></a>
```

# Local: Simple 'picture' app (using Yahoo image API)

```
# For this simple example I didn't add any styles, however views/layouts/main.tt is
where the magic happens, and styles are located in public/css/styles.css
```

```
# I chose to use Template::Toolkit instead of the default Dancer template so added this
line (after getting Template from CPAN) to both development.yml and production.yml
in /environments

template: template_toolkit
```

# Local: Running the (silly) Dancer App

```
NewApp$ bin/app.pl
```

http://0.0.0.0:3000/test

* Makes sure dancer is loading and working properly

http://0.0.0.0:3000/

* Picks a random word, grabs images from Yahoo

http://0.0.0.0:3000/word/<your_word_here>

* Takes a word as a parameter, displays images from Yahoo

# Starting with Dotcloud

```
# install the dotcloud CLI (command line interface)

$ sudo easy_install dotcloud


# The first time you use dotcloud you'll need to paste in your API key (which you'll get
from logging into your account via the web)

$ dotcloud
[ ... ]
Enter your api key (You can find it at http://www.dotcloud.com/account/settings): ...
[ ... ]
$
```

# Dotcloud: Creating a namespace

```
# The 'namespace' in dotcloud is the place where you'll 'store' your deployments. It
doesn't really matter what you call it, but shorter is probably better because you'll be
typing it a lot (4 character minimum)

$ dotcloud create blurg
```

# DotCloud: Creating a MySQL service

```
$ dotcloud deploy -t mysql blurg.mysql
Created "blurg.mysql".
$ dotcloud info blurg.mysql
cluster: wolverine
config:
   mysql_password: CRAZY_LONG_PASSWORD
   mysql_serverid: -1
created_at: 1305327597.9725609
name: blurg.mysql
namespace: blurg
ports:
-   name: ssh
   url: ssh://dotcloud@mysql.blurg.dotcloud.com:4704
-   name: mysql
   url: mysql://root:CRAZY_LONG_PASSWORD@mysql.blurg.dotcloud.com:4705
state: running
type: mysql
$
```

# Dotcloud: creating a database

```
$ dotcloud run blurg.mysql -- mysql -uroot '-pCRAZY_LONG_PASSWORD'
# mysql -uroot '-pCRAZY_LONG_PASSWORD'
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.1.41-3ubuntu12.10 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database words
mysql> exit
$
```

# Dotcloud: Importing Data MySQL

```
# Push your data to dotcloud

$ dotcloud run blurg.mysql "cat > words.sql" < words.sql

# ssh into your mysql instance and load the data

$ dotcloud ssh blurg.mysql
# $SHELL
dotcloud@blurg-mysql:~$ mysql -p words -u root < data.sql
dotcloud@blurg-mysql:~$ exit
$

# From: http://support.dotcloud.com/entries/20078551-import-and-export-mysql-
database-dumps
```

# Dotcloud: Creating a Perl service

(as with the namespace and mysql service you can call it whatever you want)

```
$ dotcloud deploy -t perl blurg.silly
Created "blurg.silly".

# Check out your services/deployments

$ dotcloud list
blurg:
  - blurg.mysql     (mysql)
  - blurg.silly   (perl)
$
```

# Local: Getting Dancer ready for Dotcloud (Important!)

```
# dotcloud requires the file app.psgi to be in the top level directory.

NewApp$ echo "require 'bin/app.pl';" > app.psgi
```

# Local: Getting Dancer ready for Dotcloud (Important!)

```
# You MUST have the file environments/deployment.yml, symlink dancer's production.yml to deployment.yml

NewApp$ ln -s environment/production.yml environment/deployment.yml


# Important for PSGI apps! From: http://blog.leecarmichael.com/2011/05/my-first-custom-dancer-app-deployment.html
```

# Local: Getting Dancer ready for Dotcloud (Important!)

```
NewApp$ vim environments/deployment.yml

plugins:
    Database:
        driver: 'mysql'
        database: 'words'
        host: 'mysql.blurg.dotcloud.com'
        # dotcloud info blurg.mysql if you've forgotten this
        port: '4705'
        username: 'root'
        password: 'CRAZY_LONG_PASSWORD'
        connection_check_threshold: 10
        dbi_params:
            RaiseError: 1
            AutoCommit: 1
            on_connect_do: ["SET NAMES 'utf8'", "SET CHARACTER SET 'utf8'" ]
            #log_queries: 1
```

# Local: Getting Dancer ready for Dotcloud (Important!)

```
# Put ALL of your dependencies in your Makefile.
# Dotcloud will load these when you deploy.
# Here's an example. Dotcloud REQUIRES Plack here:
NewApp$ vim Makefile.PL
[ ... ]
```

[ ... ]

# Deployment!

```
# If you use git for version control, then dotcloud will use it to deploy. If dotcloud can't
find .git, then it will use rsync so version control is not neccessary (although probably a
good idea!)

NewApp$ dotcloud push blurg.silly .
[ ... ]
# Dotcloud will load all dependencies here
# There should be MANY lines here (CPAN, etc).
# If there's not, something's wrong.
[ ... ]
<== Installed dependencies for .. Finishing.
21 distributions installed
uwsgi: stopped
uwsgi: started
Connection to silly.blurg.dotcloud.com closed.
$
```

# Problems? Check the logs.

```
$ dotcloud logs blurg.silly
#  tail -F /var/log/{supervisor,nginx}/*.log
==> /var/log/supervisor/supervisord.log <==
2011-06-11 00:45:21,998 INFO spawned: 'uwsgi' with pid 27333
2011-06-11 00:45:23,384 INFO success: uwsgi entered RUNNING state, process has
stayed up for > than 1 seconds (startsecs)

==> /var/log/supervisor/uwsgi.log <==

==> /var/log/nginx/access.log <==

==> /var/log/nginx/error.log <==

==> /var/log/nginx/blurg-silly.access.log <==

==> /var/log/nginx/blurg-silly.error.log <==
^CConnection to silly.blurg.dotcloud.com closed.
Abort.
$
```

# Conclusions

* During the course of my experimenting with both Dotcloud and Dancer, there are been changes in the way things are done. Both of these tools seem to be moving targets, Dancer less so.

* I spent a lot of time thinking I was doing something wrong when in fact dotcloud was having issues (it's still in beta after all)

* That being said, using Dotcloud + Dancer made programming and deploying in perl more fun then I'd had in a while.

* I wish Google docs had automatic syntax highlighting for code snippets.