Mo Lawson

Replace Pow on Mavericks (10.9) with Nginx, Dnsmasq, and Foreman

Jan 28, 2014

I'm a big fan of Pow. I've been using it for a couple of years to run various Rails and Sinatra apps locally. It makes setting up a dev environment *so* easy, and I love having separate–even multiple–.dev domains for each app.

About a year ago, the company I work for rewrote our main app on Rails. Per the usual, I've been using Pow to run the app in development, but there have been a few times where I've needed to get the app in "production-ish" mode (turn on caching, use Unicorn to serve the app, etc.) to test or debug something. I've tried to avoid doing that as much as possible, mostly because it's a pain to set some of that up and it gets me out of my normal workflow.

This last week, I was working on adding some features to the way we serve downloadable content to our customers, and I needed more than Pow could offer. In production, we use nginx with the mod_zip module to zip various files into a single download for our customers. I finally broke down and decided to setup my dev environment to more closely mirror what we've got in production so I can test anything and everything I need.

The problem was, doing this meant I'd need to set Pow aside and find another solution. But Pow has spoiled me. I love those .dev domains! As vain as that may sound, I find them really useful (and I do believe the have real, practical value). So, I googled around looking for someone with a Pow-like setup that included nginx. I found a few blog posts where people had done something similar, but it was all pre-Mavericks (OS X 10.9). And for some reason, Apple decided to get rid of bind in 10.9, which most people used for inserting the DNS settings necessary to get the .dev domains. I decided it was time to put on my big boy pants and try to piece together a solution that worked on Mavericks.

Other than the particular setup steps required for Mavericks, the only *major* change I made to others' solutions was using dnsmasq instead of bind. I started down the road of using bind, but I found dnsmasq to be much more straightforward. So, carry on reading for all the gritty details.

Homebrew

If you don't already have homebrew installed, go get it now.

```
ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/go/inst
```

That was nice! With that one command you're setup for the easiest OS X package management there is (Depending on your level familiarity with the command line and UNIX packages "easy" may not quite be the word you'd use. But I'm assuming that if you've come to this article looking for solutions to this particular problem, you either already have homebrew installed or you've messed with these things just enough that you will soon agree that, compared to other ways of managing packages on OS X, homebrew is indeed *easy*.).

Combine homebrew's relative ease of use with two of its lesser known features, brew bundle and brew services, and hopefully your development life just changed for the better, especially if you've been using something like MacPorts for the better part of last decade ;)

If you already have homebrew installed, make sure it's up to date.

```
brew update
```

Now, on to the real work.

Dnsmasq

Install dnsmasq via homebrew.

```
brew install dnsmasq
```

Add a dnsmasq config to route requests for all .dev domains to your localhost.

```
mkdir -p $(brew --prefix)/etc/
```

```
echo 'address=/.dev/127.0.0.1' > $(brew --prefix)/etc/dnsmasq.conf
```

Configure OS X to launch dnsmasq at startup and go ahead and load it now.[1]

```
do cp $(brew --prefix dnsmasq)/homebrew.mxcl.dnsmasq.plist /Library/Lau
do launchctl load -w /Library/LaunchDaemons/homebrew.mxcl.dnsmasq.plist
```

Create a resolver for .dev domains on your localhost. This tells OS X to check with dnsmasq if it's looking for a .dev domain.

```
sudo mkdir /etc/resolver
sudo bash -c 'echo "nameserver 127.0.0.1" > /etc/resolver/dev'
```

Nginx

Before we install nginx, we're going to disable Apache. You may decide to keep Apache around for whatever reason, but I found it simpler to just replace it altogether.

```
sudo launchctl unload -w /System/Library/LaunchDaemons/org.apache.htt
```

Now we can install nginx via homebrew.[2]

```
brew install nginx
```

Configure nginx to listen on port 80 by default. The homebrew install uses port 8080 as the default to prevent it clashing with Apache. Since we're replacing Apache anyway, we'll put things back the way they should be.

```
ew --prefix)/etc/nginx/nginx.conf && rm $(brew --prefix)/etc/nginx/ngin
```

We'll then create a directory for out individual app configs.

```
mkdir -p $(brew --prefix)/etc/nginx/sites
```

And configure nginx to include all config files from the new sites directory (That's a single command, so make sure you copy it all at once).

```
.bak "/^    server {/i\\
```

```
nclude $(brew --prefix)/etc/nginx/sites/*;\\

ew --prefix)/etc/nginx/nginx.conf && rm $(brew --prefix)/etc/nginx/ngin
```

Configure OS X to launch nginx at startup and to go ahead and load it now.[1]

```
sudo cp $(brew --prefix nginx)/homebrew.mxcl.nginx.plist /Library/Lau
sudo launchctl load -w /Library/LaunchDaemons/homebrew.mxcl.nginx.pli
```

App configuration

You'll need to do this part of the process for every app you want use like this. You could use a tool like nginx-app to speed up the process, but we'll do it manually this time around.

First, setup foreman to start your server. If you aren't already using foreman, I highly recommend checking it out. To get you started, run these commands.

```
cd /path/to/app
gem install foreman
echo 'web: thin start -p 3001' > Procfile
```

This setup will use thin as the server and configure it to listen on port 3001. Feel free to swap either of those out if you prefer a different server or need to use a different port.

Now, you'll need to create an nginx config in your sites directory.

```
touch $(brew --prefix)/etc/nginx/sites/app_name.conf
open $(brew --prefix)/etc/nginx/sites/
```

After those commands you should have a Finder window open with your new, empty config file. Open that file in your text editor of choice, and paste in the following config.

```
server {
  listen       80;
  server_name  app_name.dev;
  client_max_body_size 4G;
  keepalive_timeout 5;

  root /full/path/to/app;
```

```
  location / {
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_pass_header X-Accel-Redirect;
    proxy_read_timeout 300s;
    if (!-f $request_filename) {
      proxy_pass   http://127.0.0.1:3001;
      break;
    }
  }
}
```

You'll need to update that config with your app's actual path and desired hostname. You may also need to update the port on the `proxy_pass` line to match the port you specified in your Procfile.

There are plenty of other things you can do to customize this config to match your production environment, but I'll leave that to you. This should at least get you up and running.

Now, all you need to do is restart nginx.[1]

```
sudo nginx -s reload
```

Start your app.

```
cd /path/to/app && foreman start
```

And you should be able to view your app at http://app_name.dev.

Conclusion

This certainly requires considerably more effort to setup than Pow, but once you've got it going, you shouldn't need to mess with much. The main pain point now is the setup required for each app, and the nginx knowledge required to make changes. While I'll certainly work on automating any repeatable steps, I'm looking forward to learning more about nginx, so I see that as a plus.

I'm excited about being able to have a development environment that lets me more easily

test our app from top to bottom and to tweak things from our database query performance to our browser performance as a part of my normal workflow.

---

[1] These commands are necessary because the `sudo brew services` commands were causing problems. I have a pull request open that should fix the issue.

[2] This is just a standard nginx install. If you need the mod_zip module–or any of a number of other modules–check out the marcqualie/nginx tap.

Mo Lawson

Mo Lawson
mo@molawson.com

molawson
molawson