Help Scout

Continuous Integration Engineer

Along with managing our CI infrastructure, you'll be helping our developers become more efficient at their work. You should have already received a link to 'hello13.tgz'. This is a small PHP app that you'll be sprucing up the development workflow on.

You'll be bringing this application into version control, getting it hooked into TravisCI, and finally build out a Vagrant box for developer use.

Basics

1. If you don't already have them, create accounts at GitHub and TravisCI. (Don't worry, both are free).
2. Create a new public repository on GitHub for the application, and configure TravisCI to run the phpunit tests on the application.
3. Update the Vagrantfile to make 'hello13' available inside the vagrant box at '/srv/hello13'.
4. Update the Vagrantfile to provision the Vagrant box using a Chef provisioner.
5. Update the 'hello13' chef cookbook to configure the Vagrant box for serving 'hello13'. We're fans of Nginx with php-fpm, but feel free to use the webserver of your choice. The webserver should be listening on port 8181, and should be configured to serve the PHP application at "/", and the static assets at "/static". (No need to get fancy here -- Just install the services, make sure they start at boot, and drop a minimal webserver site config in place.)
6. Ensure your Vagrantfile is configured to allow the application to be accessed from port 8181 on the Vagrant host.

Possible Enhancements

1. Configure any shared directories such that edits to files are immediately visible both inside and outside the Vagrant box.
2. Get fancy with "Basics" number 5: Update your cookbook to allow finer grain control of php-fpm. Specifically, configure the number of workers using a Chef node attribute.

Deliverables

1. Repository on GitHub with:
    a. TravisCI integration
    b. A functional Chef cookbook
    c. A functional Vagrantfile
2. A brief document explaining why you built it the way you did.
3. A postmortem summarizing your findings. Did it go smoothly? Any surprises or lessons learned? Should we implement a production-ready version of what you build, or would you do things differently next time?