Deploying a web app: the road less travelled

Riccardo Magliocchetti

Torino Coding Society 18/01/15

whoami

Consultant

Free software developer

- maintainer: django-admin-bootstrapped, uwsgitop, bootchart2
- contributor: uwsgi, LibreOffice

Deploy

What does it even mean?

deploy

to organize and send out (people or things) to be used for a particular purpose

aka

put stuff from my laptop to the Internet :)

Building blocks

Our Code

Other people's software

Automation

Code

SCM

A build pipeline -> artifact

Other people's software

HTTP server / proxy

Application server (or not)

Database server

In memory store

Queue system

all this stuff really?

short answer: yes

Plus monitoring :)

Options: from more painful to more lame

Bare metal (Github)

Public Cloud (Netflix)

PAAS <- sweet spot for most people?

90s LAMP stack (hi PHP hosting!)

automation

provisioning

deployment

in other words

DevOps Borat

```
To make error is human. To
propagate error to all server in
automatic way is #devops
```

mythical figures

sysadmin

devops

uwsgi.it

uwsgi.it

- free software PAAS (MIT)
- C + perl + nginx + django + postgres
- linux containers based
- REST API
- based on uwsgi

https://github.com/unbit/uwsgi.it

Features

- Choose your own distribution for rootfs
- SSL is not a costly update (hey heroku)
- Clustering out of the box
- Third party alarms included

Missing features

- No git integration / cool cli (as heroku)
- No marketplace

uwsgi

uwsgi

An application server container (GPL)

A way to run and manage python, ruby, lua, mono, jvm, js and even php apps

https://github.com/unbit/uwsgi
https://github.com/unbit/uwsgi-docs

uwsgi

Created by an ISP

PROs

- focus on low usage resources
- many different stacks support

CONs

- every possible options possible to make customers app work

options digression

I meant for real:

```
$ ./uwsgi --help | wc -l
937
```

Features

- fastrouter: proxy / load balancer / router
- socket less workers aka mules
- async task spooler
- cron-like interface
- caching framework and caching cookbok
- external services management
- routing system
- metrics
- lot more stuff that doesn't fit here: native http, signals, rpc, ...
- lot of plugins

some functionality exposed as APIs

- python, perl, lua
- ruby dsl

API:

- cache, queue, spooler, mules, cron, timer, ...

Fancy architecture

overview

emperor + vassals + http server + fastrouter + external daemon

```
$ find .
.
./emperor.ini
./vassals
./vassals/webserver.ini
./vassals/fastrouter.ini
./vassals/one.ini
./vassals/two.ini
./vassals/redis.ini
```

emperor

```
[uwsgi]
emperor = %d/vassals
```

http server

```
[uwsgi]
plugins = http
master = true
# listen for http
http = 0.0.0.0:8080
# forward requests via uwsgi
protocol
http-to = 127.0.0.1:3031

logto = %d/webserver.log
```

fastrouter

```
[uwsgi]
plugin = fastrouter

;create a shared socket (the webserver will
connect to it)
fastrouter = 127.0.0.1:3031

; our subscription server
fastrouter-subscription-server =
127.0.0.1:4040

logto = %d/fastrouter.log
```

workers: python

```
[uwsgi]
plugins = python
master = true
socket = 127.0.0.1:3041
subscribe-to =
127.0.0.1:4040:127.0.0.1:8080
wsgi-file = %d/../app.wsgi
processes = 1
logto = %d/one.log
```

workers: ruby

```
[uwsgi]
plugins = rack
master = true
socket = 127.0.0.1:3042
subscribe-to =
127.0.0.1:4040:127.0.0.1:8080
rack = %d/../app.ru
; required by rack specification
post-buffering = 4096
processes = 1
logto = %d/two.log
```

redis

```
[uwsgi]
smart-attach-daemon = %d/redis.pid
redis-server --unixsocket
%d/redis.socket --port 0 --pidfile
%d/redis.pid --daemonize yes
```

Let's run it

```
$ uwsgi emperor.ini
```

Demo time!

Happy Hacking :)

Riccardo Magliocchetti

riccardo@menodizero.it

@rmistaken

http://menodizero.it

https://github.com/xrmx

Torino Hacknight

Easiest way to start hacking on free software

@tohacknight - http://torino.hacknight.it