

# Hi there!

I'm Michael Schramm

[github.com/wodka](https://github.com/wodka)  
[twitter.com/wodkamichi](https://twitter.com/wodkamichi)

born in Salzburg

living in Vienna

php development since 2003

# PHP on HEROKU

by Michael Schramm

# Alternatives

(Heroku runs on AWS)

AWS Elastic Beanstalk

AWS OpsWorks (Chef)

IBM BlueMix

Classic Hosting (V-Server, Root-Server)

(+ many more)

# Heroku

deploy > manage > scale

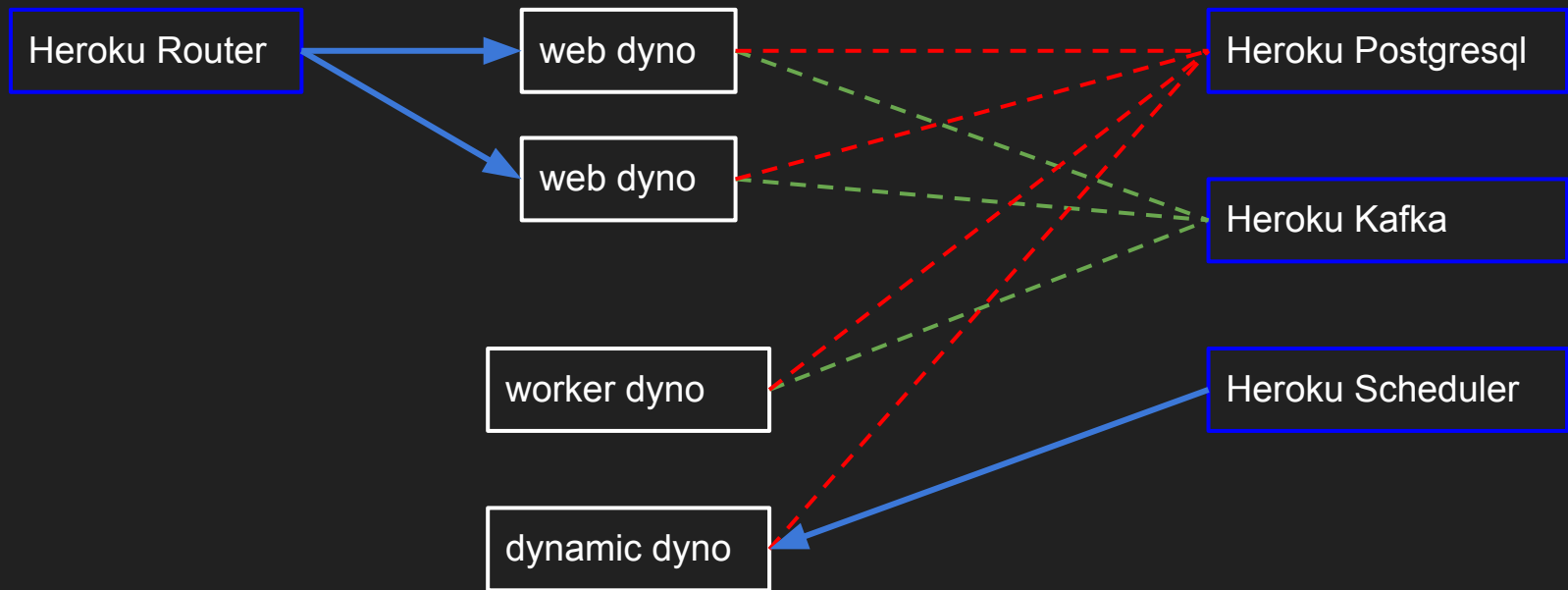
Focus on code

One repository for your app

Forces good design!

- + many addons
- + simple interface
- rel. high cost (if sys admin is excluded)

# Heroku Structure



# Heroku Dynos

How to make them fit

Addons

Buildpacks

Buttons

<https://elements.heroku.com/>

# Heroku Dynos

How they tick

Dynos will be reseted every 24h

No persistent storage

Settings from environment

Compile configuration in buildpacks

Application from Composer

# Heroku Dynos



## Free

Ideal for experimenting with cloud applications in a limited sandbox.

### CORE PLATFORM FEATURES

SLEEPS AFTER 30 MINS OF INACTIVITY

USES AN ACCOUNT-BASED POOL  
OF FREE DYNOS HOURS

CUSTOM DOMAINS

512 MB RAM | 1 web/1 worker

Free



## Hobby

Perfect for small scale personal projects and hobby apps.

### CORE PLATFORM FEATURES

NEVER SLEEPS

FREE SSL & AUTOMATED CERTIFICATE  
MANAGEMENT FOR CUSTOM DOMAINS

APPLICATION METRICS

MULTIPLE WORKERS FOR MORE  
POWERFUL APPS

512 MB RAM | 10 Process Types

**\$7** per dyno/month  
prorated to the second



## Standard 1X 2X

Enhanced visibility, performance, and availability for powering your professional applications.

### ALL HOBBY FEATURES +

SIMPLE HORIZONTAL SCALABILITY

THRESHOLD ALERTS

PREBOOT

LANGUAGE RUNTIME METRICS

512MB OR 1GB RAM

∞ Process Types



**\$25 - \$500** per dyno/month  
prorated to the second



## Performance M L

Superior performance when it's most critical for your super scale, high traffic apps.

### ALL STANDARD FEATURES +

MIX WITH STANDARD 1X, 2X DYNOS

DEDICATED

AUTOSCALING

2.5GB OR 14GB RAM



# Cost

Web Dynos ~ 50\$

Worker ~ 25\$

Postgresql ~ 50\$

Total of 125\$ per month!

- + No server maintenance
- + Simple Deploy

Compare this to setting up

- Own infrastructure
- Deployment Tools
- Time it took (>> 2h per month)

scale



Heroku

AWS OpsWorks / Bluemix / ...

Your own Datacenters ;)

# Configuration

## Environment

parameters.yml.dist

```
parameters:
  database_host:      %env(DATABASE_HOST)%
  database_port:      %env(DATABASE_PORT)%
  database_name:      %env(DATABASE_NAME)%
  database_user:      %env(DATABASE_USER)%
  database_password:  %env(DATABASE_PASSWORD)%
```

The screenshot shows the AWS IAM console interface for the 'gosepp-production' environment. The top navigation bar includes tabs for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. The 'Settings' tab is selected. Below the navigation bar, the environment name 'gosepp-production' is displayed with an 'Edit' link. The main section is titled 'Config Variables' and contains a description: 'Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.' To the right of this description is a 'Hide Config Vars' button. Below the description, there is a table of configuration variables:

Config Vars		
AWS_KEY	[REDACTED]	[Edit] [Delete]
AWS_SECRET	[REDACTED]	[Edit] [Delete]
BOTFRAMEWORK_HANDLE	[REDACTED]	[Edit] [Delete]
BOTFRAMEWORK_ID	[REDACTED]	[Edit] [Delete]

# Configuration

## Dyno Configuration

Procfile in root defines Dynos

```
web: vendor/bin/heroku-php-nginx -C nginx_app.conf web/  
worker: php --php-ini web/.user.ini bin/console thread:pool --threads=8 queue:worker
```

# Configuration

“web” dyno

web: vendor/bin/heroku-php-nginx -C nginx\_app.conf web/

## Nginx with custom configuration

(could as well be Apache)

nginx\_app.conf

```
client_max_body_size 20M;

location /config {
    rewrite ^(.*)$ /config.php/$1 last;
}

location / {
    # try to serve file directly, fallback to rewrite
    try_files $uri @rewriteapp;
}

location @rewriteapp {
    # rewrite all to app.php
    rewrite ^(.*)$ /app.php/$1 last;
}

location ~ ^/(app|app_dev|config)\.php(/|$) {
    try_files @heroku-fcgi @heroku-fcgi;
    internal;
}
```

<https://devcenter.heroku.com/articles/custom-php-settings#web-server-settings>

# Configuration

“web” dyno

web: vendor/bin/heroku-php-nginx -C nginx\_app.conf web/

PHP Configuration is taken from  
.user.ini file in working dir (web/)

```
upload_max_filesize = 20M  
  
opcache.max_accelerated_files = 20000  
  
realpath_cache_size=10M  
realpath_cache_ttl=7200  
  
default_socket_timeout=86400  
  
serialize_precision = -1
```

<https://devcenter.heroku.com/articles/custom-php-settings#php-runtime-settings>

# Configuration

“web” dyno

web: vendor/bin/heroku-php-nginx -C nginx\_app.conf web/

composer install is run automatically

Important: add **composer.lock** to repo!

- + PHP Extensions
- + PHP Version

Add compile script to  
execute during  
deployments

```
"require": {  
  "php": "~7.0",  
  "ext-apcu": "*",  
  "ext-curl": "*",  
  "ext-fileinfo": "*",  
  "ext-gd": "*",  
  "ext-intl": "*",  
  "ext-mbstring": "*",  
  "ext-mcrypt": "*",  
  "ext-pcntl": "*",  
  "ext-redis": "*",
```

```
"scripts": {  
  "symfony-scripts": [...],  
  "post-install-cmd": [...],  
  "post-update-cmd": [...],  
  "compile": [  
    "rm web/app_dev.php",  
    "bin/console assetic:dump"  
  ]  
},
```

# Configuration

“worker” dyno

worker: php --php-ini web/.user.ini bin/console thread:pool --threads=8 queue:worker

All non web dynos start a single process!

Fails to take php config automatically!

Requires all of memory? Threading?



# Workers can do more!

Our worker is for a SQS Queue and has to process many but short tasks.

# Heroku Workers

make them robust

No automatic restart after a crash!

why not use a loop?

```
worker: while true; do php bin/console queue:worker; sleep 1; done
```

It works, but still idling a lot!

# Options:

- Switch to a language that has threads
- Execute php from some thread manager
- Write a php solution XD

# Heroku Workers

Of course a PHP solution!

Basic idea:

`worker: bin/console thread:pool --threads=4 queue:worker`

i.E. start the “queue:worker” 4 times and keep track of them.

Problems arising:

- Log handling
- Signal Handling

<https://blog.ms07.at/2017/04/15/heroku-php-worker-symfony3/>

# Heroku Workers

Of course a PHP solution!

Using symfony process component

makes it easy to forward output to the main process!

PCNTL for signal handling

Heroku sends first a SIGTERM and gives the dyno 30 seconds to end. If this fails a SIGKILL is sent. We have to terminate all processes.

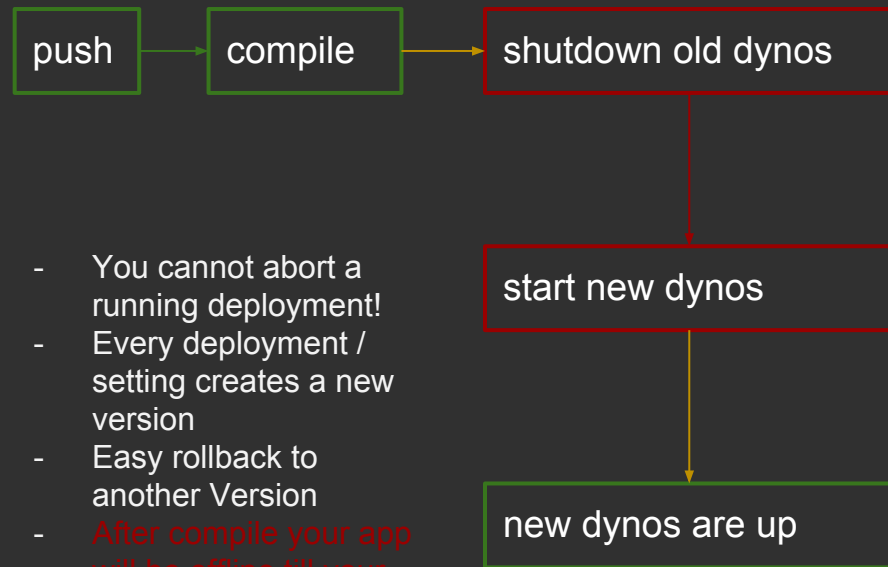
thread:pool in gist is used @GoSEPP for all SQS processing

<https://gist.github.com/wodka/23475d36cf13e956b8db7578bf6251ed>  
<http://symfony.com/doc/current/components/process.html>

# Heroku Deploy


Just a git push away

Push to git to trigger a deploy



- You cannot abort a running deployment!
- Every deployment / setting creates a new version
- Easy rollback to another Version
- After compile your app will be offline till your app is started (takes a few seconds)

# Seamless Deployments

```
wodka@smith:~$ heroku features --app gosepp-production
=== App Features  gosepp-production
[ ] http-session-affinity  Enable session affinity for all requests [general]
[+] preboot                Provide seamless web dyno deploys [general]
[ ] spaces-dns-discovery   Enable DNS service discovery [general]
```

# Heroku & Symfony

## Folder Structure

/composer.json  
/composer.lock  
/Procfile  
/nginx\_app.conf  
/web/.user.ini  
/web/index.php

Dyno Configuration  
Nginx router  
PHP Settings  
Web entry point



Requiring more flexibility?

I like  CHEF™

<https://www.chef.io/chef/>

# Managed Chef for AWS

## Fine grained control of infrastructure

## All EC2 types available

```
##### OpsWorks Summary #####
Operating System: Ubuntu 14.04.5 LTS
OpsWorks Instance: php-app2
OpsWorks Instance ID: i-4f81988c
OpsWorks Layers: PHP App Server
OpsWorks Stack: front-end
EC2 Region: us-east-1
EC2 Availability Zone: us-east-1a
EC2 Instance ID: i-4f81988c
Public IP: 54.166.100.100
Private IP: 10.144.100.100

Visit http://aws.amazon.com/opsworks for more information.
Last login: Thu Oct 5 11:13:31 2017 from 10.144.100.100
michaelschramm@gmail.com@php-app2:~$
```

# AWS OpsWorks Configuration

OpsWorks Stacks > front-end

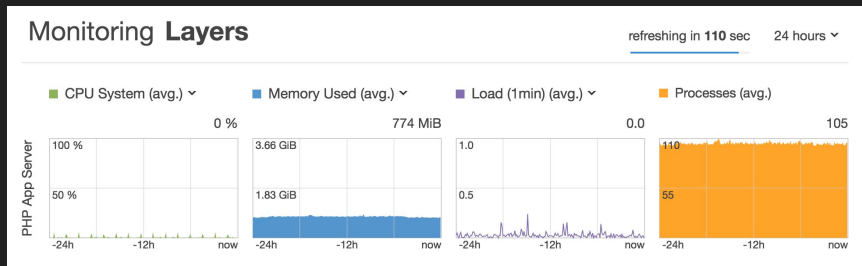
Stack Layers

ELB: www

PHP App Server

RDS: production

+ Layer



## Layer PHP App Server

General Settings Recipes Network EBS Volumes Security CloudWatch Logs Tags

### Built-in Chef Recipes

We have defined 18 built-in Chef recipes for your layer.

- 8 Setup opsworks\_initial\_setup ssh\_host\_keys ssh\_users mysql::client dependencies ebs opsworks\_ganglia::client mod\_php5\_apache2
- 5 Configure opsworks\_ganglia::configure-client ssh\_users mysql::client agent\_version php::configure
- 2 Deploy deploy::default deploy::php
- 1 Undeploy deploy::php-undeploy
- 2 Shutdown opsworks\_shutdown::default apache2::stop

### Custom Chef Recipes

Repository URL git@github.com:mycookbooks:chef-cookbooks.git (change)

- 0 Setup mycookbook::myrecipe, mycookbook::mycookbook
- 0 Configure mycookbook::myrecipe, mycookbook::mycookbook
- 1 Deploy mycookbook::myrecipe, mycookbook::mycookbook mycookbook::deploy
- 0 Undeploy mycookbook::myrecipe, mycookbook::mycookbook
- 0 Shutdown mycookbook::myrecipe, mycookbook::mycookbook

# Cost

ELB ~ 26\$

Web 2x ~ 10\$

Worker ~ 5\$

Postgresql (multi AZ) ~ 36\$

Total of 77\$ per month!

- server maintenance
- manual deployment (trigger)
- many hours for setup

# questions?

Slides soon available on <https://github.com/viennaphp>