PHP MEETS DOCKER

A WHALE AND AN ELEPHANT

WHO AM I?

I'm Steve McDougall

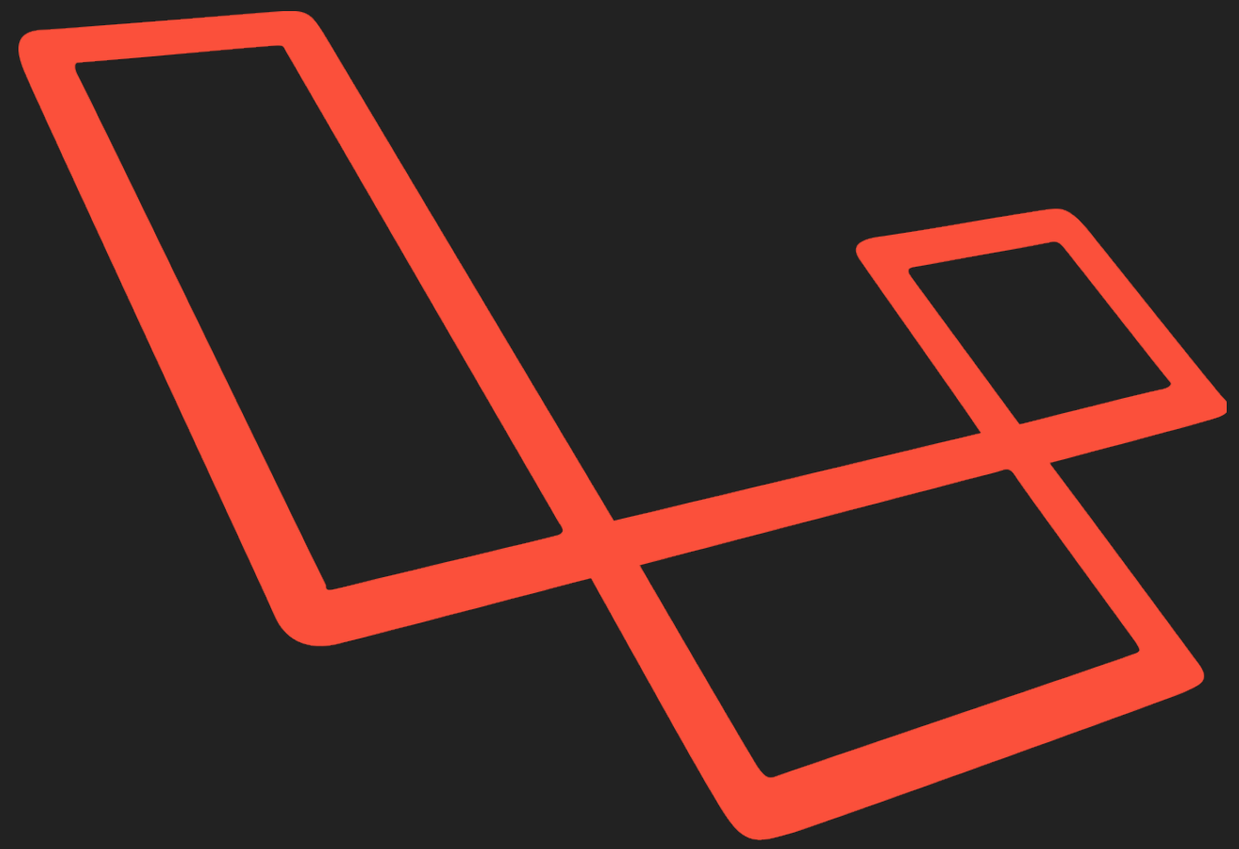
- PHP User Group Organiser
- Conference Organiser
- Experienced Developer
- PHP Advocate

A MODERN WORKFLOW FOR PHP DEVELOPERS

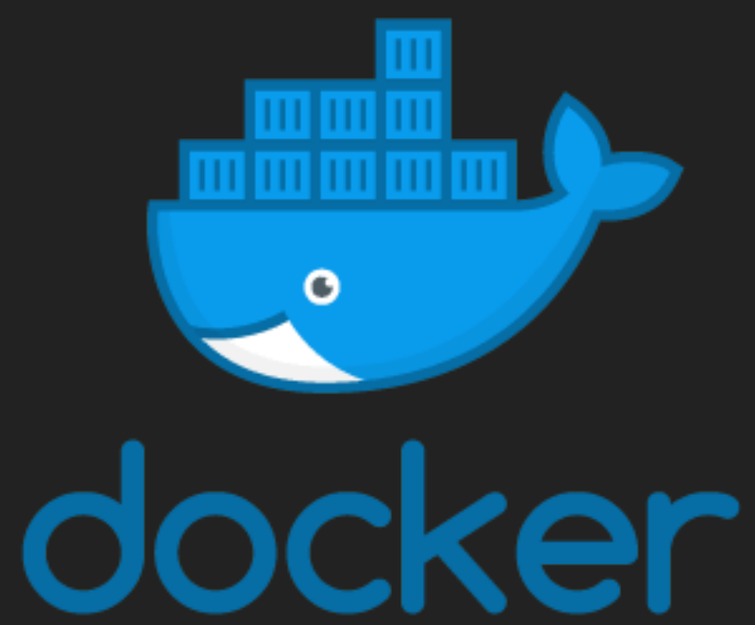
LETS START SIMPLE

At this point frameworks aren't important

For this talk I will use Laravel as it has an easy installer



INTRODUCING DOCKER



There are many ways to add Docker to your PHP projects, for this talk I will be using DDEV

SET UP ENVIRONMENTS IN MINUTES; SWITCH CONTEXTS AND PROJECTS QUICKLY; SPEED YOUR TIME TO DEPLOYMENT

DDEV

WHAT IS DDEV

DDEV is an amazing little tool to add Docker into any PHP application, with a great command line and easy to use configuration it is my go to for containers.

LETS ADD IT TO OUR APP

```
[ ~/sites/bookshelf  master  ddev config
Creating a new ddev project config in the current directory (/Users/steve/sites/bookshelf)
Once completed, your configuration will be written to /Users/steve/sites/bookshelf/.ddev/config.yaml

Project name (bookshelf):

The docroot is the directory from which your site is served.
This is a relative path from your project root at /Users/steve/sites/bookshelf
You may leave this value blank if your site files are in the project root
Docroot Location (public):
Found a php codebase at /Users/steve/sites/bookshelf/public.
Project Type [backdrop, php, drupal6, drupal7, drupal8, wordpress, typo3] (php):
Project type has no settings paths configured, so not creating settings file.
Configuration complete. You may now run 'ddev start'.
~/sites/bookshelf  master
```

WE DID IT!

With one line we added Docker to out application

```yaml
APIVersion: v1.5.2
name: bookshelf
type: php
docroot: public
php_version: "7.1"
webserver_type: nginx-fpm
router_http_port: "80"
router_https_port: "443"
xdebug_enabled: false
additional_hostnames: []
additional_fqdns: []
mariadb_version: "10.2"
webcache_enabled: false
provider: default
```

HOW DO WE CONFIGURE DDEV

LETS START OUR APPLICATION

```
~/sites/bookshelf  master  ddev start
Starting bookshelf...
ddev needs to add an entry to your hostfile.
It will require administrative privileges via the sudo command, so you may be required
to enter your password for sudo. ddev is about to issue the command:
    sudo /usr/local/bin/ddev hostname bookshelf.ddev.local 127.0.0.1
Please enter your password if prompted.
Running Command  Command=sudo /usr/local/bin/ddev hostname bookshelf.ddev.local 127.0.0.1
Password:
Creating volume "bookshelf-mariadb" with default driver
Pulling db (drud/ddev-dbserver:v1.6.0-10.2)...
Pulling web (drud/ddev-webserver:v1.6.0)...
Pulling dba (drud/phpmyadmin:v1.6.0)...
Creating ddev-bookshelf-db ... done
Creating ddev-bookshelf-dba ... done
Creating ddev-bookshelf-web ... done

Pulling ddev-router (drud/ddev-router:v1.6.0)...
Recreating ddev-router ... done

Successfully started bookshelf
Project can be reached at http://bookshelf.ddev.local:8081, https://bookshelf.ddev.local, http://127.0.0.1:32771
~/sites/bookshelf  master
```

ONTO CI/CD

TESTING

We all know Laravel has a nice testing suite, so let's add some *very* basic tests, to prove a point.

```php
<?php

namespace Tests\Unit;

...

class AuthorTest extends TestCase
{
    use DatabaseTransactions;

    public function testThatAuthorsCanBeCreated()
    {
        $author = factory(Author::class)->create();

        $this->assertDatabaseHas('authors', $author->toArray());
    }

    public function testThatAuthorsCanBeUpdated()
    {
        $author = factory(Author::class)->create();
        $author->update([
            'name' => 'PHP Unit'
        ]);

        $this->assertDatabaseHas('authors', $author->toArray());
    }
}
```

NEXT STOP: GITLAB

Next stop is to get all this magic up on GitLab. We all know how git works, so fast forward a couple of minutes.

GitLab

TIME TO CONFIGURE OUR SERVER

```
# Create user elephant
sudo adduser elephant
# Give the read-write-execute permissions to elephant user for directory /var/www
sudo setfacl -R -m u:elephant:rwx /var/www

# As the elephant user on server
#
# Copy the content of public key to authorized_keys
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
# Copy the private key text block
cat ~/.ssh/id_rsa
```

GITLAB NEEDS TO KNOW ABOUT OUR PRIVATE KEY

Environment variables

Collapse

Environment variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. You can use environment variables for passwords, secret keys, or whatever you want. You may also add variables that are made available to the running application by prepending the variable key with `K8S_SECRET_`. More information

SSH_PRIVATE_KEEY	Input variable value	Protected
Input variable key	Input variable value	Protected

Save variables Hide values

MAKE SURE TO ADD IT TO YOUR REPO!

Deploy Keys

Collapse

Deploy keys allow read-only or read-write (if enabled) access to your repository. Deploy keys can be used for CI, staging or production servers. You can create a deploy key or add an existing one.

Create a new deploy key for this project

Title

elephant

Key

Paste a machine public key here. Read more about how to generate it here

☑ **Write access allowed**

Allow this key to push to repository as well? (Default only allows pull access.)

Add key

CONFIGURE NGINX

```
server {
    root /var/www/app/current/public;
    server_name example.com;
    # Rest of the configuration
}
```

ENTER ENVOY

LARAVEL ENVOY PROVIDES A CLEAN, MINIMAL SYNTAX FOR DEFINING COMMON TASKS YOU RUN ON YOUR REMOTE SERVERS.

Laravel Envoy

A WHALE AND AN ELEPHANT

- **Tasks:** Tasks are commands to be ran on your remote server
- **Stories:** Stories are groups of tasks
- **Notifications:** Send notifications to Slack or Discord

```
@servers(['web' => 'elephant@remote_host'])

@task('git')
    git pull origin master
@endtask

@task('composer')
    composer install
@endtask

@task('migrate')
    php artisan migrate
@endtask

@story('deploy', ['on' => 'web', 'confirm' => true])
    cd site
    git
    composer
    migrate
@endstory

@finished
    @slack('webhook-url', '#bots')
@endfinished
```

MANAGING RELEASES

Envoy has a useful setup directive, which allows you to set initial variables to use.

```
@setup
    $repository = 'git@gitlab.example.com:juststeveking/bookshelf.git';
    $releases_dir = '/var/www/app/releases';
    $app_dir = '/var/www/app';
    $release = date('YmdHis');
    $new_release_dir = $releases_dir .'/'. $release;
@endsetup
```

GITLAB: CONTAINER REPOSITORY

CREATE A .GITLAB-CI.YML TO MANAGE THE PROCESS

```yaml
image: registry.gitlab.com/juststeveking/bookshelf:latest

services:
  - mysql:5.7

variables:
  MYSQL_DATABASE: homestead
  MYSQL_ROOT_PASSWORD: secret
  DB_HOST: mysql
  DB_USERNAME: root

stages:
  - test
  - deploy

unit_test:
  stage: test
  script:
    - cp .env.example .env
    - composer install
    - php artisan key:generate
    - php artisan migrate
    - vendor/bin/phpunit

deploy_production:
  stage: deploy
  script:
    - 'which ssh-agent || ( apt-get update -y && apt-get install openssh-client -y )'
    - eval $(ssh-agent -s)
    - ssh-add <(echo "$SSH_PRIVATE_KEY")
    - mkdir -p ~/.ssh
    - '[[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'

    - ~/.composer/vendor/bin/envoy run deploy --commit="$CI_COMMIT_SHA"
  environment:
    name: production
    url: http://192.168.1.1
  when: manual
  only:
    - master
```

TEXT

CI/CD IS EASIER THAN YOU THINK, AND WITH LOADS OF TOOLS OUT THERE YOU HAVE NO EXCUSE NOT TO USE THEM

Steve McDougall

GitHub: JustSteveKing
Twitter: @JustSteveKing

THANK YOU FOR LISTENING