

Dockerizing PHP Apps

Aurelijus Banelis



VilniusPHP
2018-11-08



Aurelijus Banelis

Backend/DevOps

aurelijus.banelis.lt

aurelijus@banelis.lt

PGP 0x320205E7**539B6203**
130D C446 1F1A 2E50 D6E3
3DA8 3202 05E7 539B 6203



Adapting PHP applications to be used with docker

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS

Introduction

Why to dockerize

What is docker

How to dockerize

Development

Context/Alternatives

What I really liked

What I do not like

Production

Deploying to AWS

Monolith (VirtualBox) → Split
Development bottleneck

Jenkins (time based) → CircleCI
Confidence/tooling bottleneck

Release coordination → AWS services
Provisioning speed bottleneck

Monolith (VirtualBox) → Split
Development bottleneck

Jenkins (time based) → CircleCI
Confidence/tooling bottleneck

Need for better virtualization tools

Release coordination → AWS services
Provisioning speed bottleneck

Monolith (VirtualBox) → Split

Development bottleneck

Jenkins (time based) → CircleCI

Confidence/tooling bottleneck

virtualization tools

Release coordination → AWS services

Provisioning speed bottleneck

docker?

We migrated to docker

Using docker extensively at



Learning curve was high

Were pioniers, added tooling around

I would do the same again

Right decision for current scale

Future opportunities

Introduction

Why to dockerize

What is docker

How to dockerize

Development

Context/Alternatives

What I really liked

What I do not like

Production

Deploying to AWS

Tool to run in isolated environment

Many ifs in kernel = cgroups

Not VirtualBox, not unikernel



Open source tool backed by Docker Inc

Container hosting and premium service

Improved by community (AWS, K8s)

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS

 `index.php`

```
1  <?php
2  echo 'Hello from docker. ';
3  echo 'PHP version: ' . phpversion();
```

Dockerfile

```
1 FROM php:7.2-apache
2
3 COPY . /var/www/html/
```

index.php

```
1 <?php
2 echo 'Hello from docker. ';
3 echo 'PHP version: ' . phpversion();
```

Dockerfile

```
1 FROM php:7.2-apache
2
3 COPY . /var/www/html/
```

index.php

```
1 <?php
2 echo 'Hello from docker. ';
3 echo 'PHP version: ' . phpversion();
```

```
docker build . -t vilniusphp-example
docker run -p 8080:80 vilniusphp-example
```

Dockerfile

```
1 FROM php:7.2-apache
2
3 COPY . /var/www/html/
```

index.php

```
1 <?php
2 echo 'Hello from docker. ';
3 echo 'PHP version: ' . phpversion();
```

```
docker build . -t vilniusphp-example
docker run -p 8080:80 vilniusphp-example
```



127.0.0.1:8080

Hello from docker. PHP version: 7.2.11

Dockerfile

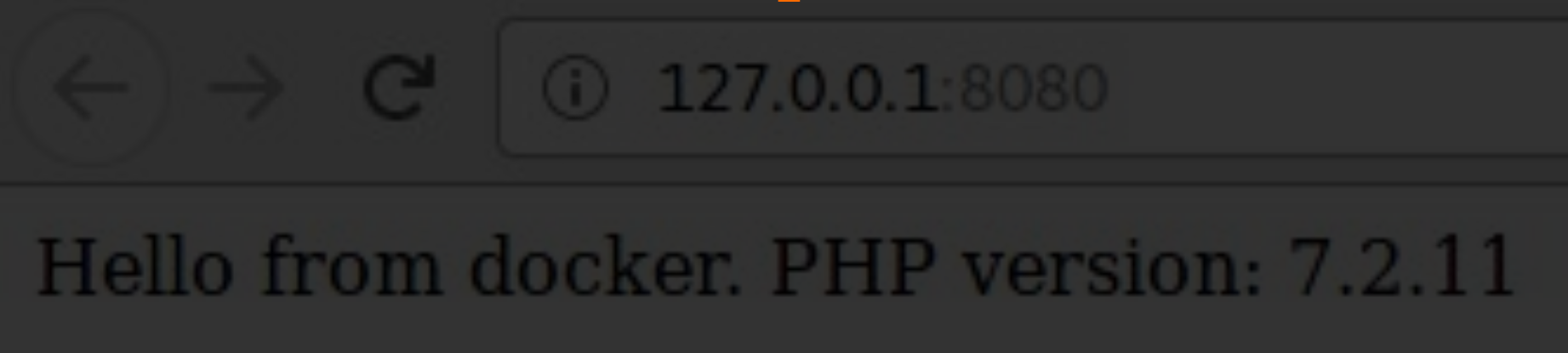
```
1 FROM php:7.2-apache
2
3 COPY . /var/www/html/
```

index.php

```
1 <?php
2 echo 'Hello from docker. ';
3 echo 'PHP version: ' . phpversion();
```

```
docker build . -t vilniusphp-example
docker run -p 8080:80 vilniusphp-example
```

Simple?



Dockerfile

```
1 FROM php:7.2-apache
2
3 COPY . /var/www/html/
```

index.php

```
1 <?php
2 echo 'Hello from docker. ';
3 echo 'PHP version: ' . phpversion();
```

```
docker build . -t vilniusphp-example
docker run -p 8080:80 vilniusphp-example
```

Simple?

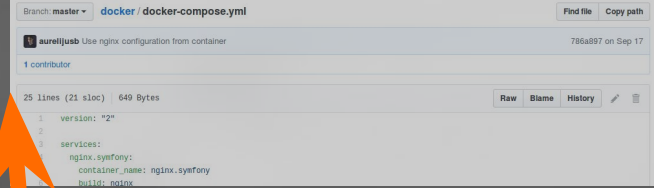
But for simple cases

Hello from docker. PHP version: 7.2.11

43 lines (34 sloc) | 1.46 KB

Raw Blame History

```
1 FROM php:7.2.4-fpm
2
3 LABEL maintainer "Aurelijus Vanelis <vanelis@vanelis.lt>"
4
5 WORKDIR /php
6
7 # Get composer: https://getcomposer.org/download/
8 RUN php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
9 RUN php -r "if (hash_file('SHA384', 'composer-setup.php') === '544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d447e2586475ca9
10 RUN php composer-setup.php
11 RUN php -r "unlink('composer-setup.php');"
12 RUN ln -s /php/composer.phar /usr/bin/composer
13
14 # Install dependencies
15 RUN apt-get update \
16 && apt-get install -y git zip unzip \
17 && rm -rf /var/lib/apt/lists/*
18
19 # Install PHP extensions
20 RUN apt-get update \
21 && apt-get install -y libzip-dev bash-completion procps nano \
22 && docker-php-ext-install -j$(nproc) zip mysqli pdo_mysql \
23 && rm -rf /var/lib/apt/lists/*
24
25 # xDebug helpers (do not use this in real production)
26 ADD enable_xdebug.sh /enable_xdebug.sh
27 ADD disable_xdebug.sh /disable_xdebug.sh
28 RUN pecl install xdebug-2.6.0 && \
29     chmod +x /enable_xdebug.sh && \
30     chmod +x /disable_xdebug.sh && \
31     touch /usr/local/etc/php/conf.d/custom-xdebug.ini && \
32     chmod 777 /usr/local/etc/php/conf.d/custom-xdebug.ini
33
34 # Not root user
35 RUN useradd -c 'PHP user' -m -d /home/php -s /bin/bash php
36 USER php
37 ENV HOME /home/php
38
39 # xDebug configuration
40 ENV PHP_IDE_CONFIG serverName=nfqKickStartDocker
41
42 WORKDIR /code
43 VOLUME /code
```



Multiple containers

Development tools

PHP extensions

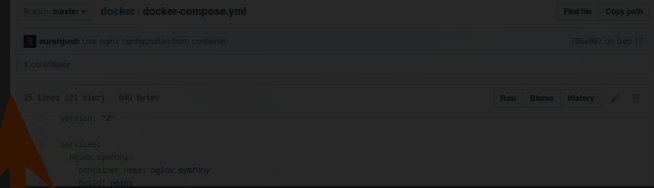
xDebug support

...

43 lines (34 sloc) 1.46 KB

Raw Blame History

```
1 FROM php:7.2.4-fpm
2
3 LABEL maintainer "Aurelijus V. Velička <velicka@phanelis.lt>"
4
5 WORKDIR /php
6
7 # Get composer: https://getcomposer.org/download/
8 RUN php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
9 RUN php -r "if (hash_file('SHA384', 'composer-setup.php') === '544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d5471f53be6475ca9
10 RUN php composer-setup.php
11 RUN php -r "unlink('composer-setup.php');"
12 RUN ln -s /php/composer.phar /usr/bin/composer
13
14 # Install dependencies
15 RUN apt-get update \
16     && apt-get install -y git zip unzip \
17     && rm -rf /var/lib/apt/lists/*
18
19 # Install PHP extensions
20 RUN apt-get install -y \
21     && docker-php-ext-install -v --with-zlib-dev --with-mcrypt --with-openssl --with-pdo-mysql nano \
22     && docker-php-ext-install mysqli pdo_mysql \
23     && rm -rf /var/lib/apt/lists/*
24
25 # xDebug support (do not use this in real production)
26 ADD enable_xdebug.sh /enable_xdebug.sh
27 ADD disable_xdebug.sh /disable_xdebug.sh
28 RUN pecl install xdebug-2.6.0 && \
29     chmod +x /enable_xdebug.sh && \
30     chmod +x /disable_xdebug.sh && \
31     touch /usr/local/etc/php/conf.d/custom-xdebug.ini && \
32     chmod 777 /usr/local/etc/php/conf.d/custom-xdebug.ini
33
34 # Not root user
35 RUN useradd -c 'PHP user' -m -d /home/php -s /bin/bash php
36 USER php
37 ENV HOME /home/php
38
39 # xDebug configuration
40 ENV PHP_IDE_CONFIG serverName=nfqKickStartDocker
41
42 WORKDIR /code
43 VOLUME /code
```



Multiple containers

Development tools

PHP extensions

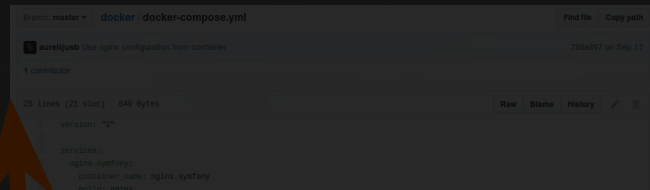
xDebug support

...

43 lines (34 sloc) 1.46 KB

Raw Blame History

```
1 FROM php:7.2.4-fpm
2
3 LABEL maintainer "Aurelijus V. Belis <@phanelis.lt>"
4
5 WORKDIR /php
6
7 # Get composer: https://getcomposer.org/download/
8 RUN php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
9 RUN php -r "if (hash_file('SHA384', 'composer-setup.php') === '544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d4847e6475ca9
10 RUN php composer-setup.php
11 RUN php -r "unlink('composer-setup.php');"
12 RUN ln -s /php/composer.phar /usr/bin/composer
13
14 # Install dependencies
15 RUN apt-get update \
16     && apt-get install -y git zip unzip \
17     && rm -rf /var/lib/apt/lists/*
18
19 # Install PHP extensions
20 RUN apt-get install -y \
21     curl \
22     libcurl4-openssl-dev \
23     libzip-dev \
24     libzip-dev \
25     # xDebug support (do not use this in real production)
26     ADD enable_xdebug.sh /enable_xdebug.sh
27     ADD disable_xdebug.sh /disable_xdebug.sh
28     RUN pecl install xdebug-2.6.0 && \
29         chmod +x /enable_xdebug.sh && \
30         chmod +x /disable_xdebug.sh && \
31         touch /etc/php/7.2/cli/conf.d/20-xdebug.ini \
32         chmod +x /php/bin/php-fpm && \
33         # Not required for CLI but for FPM
34     RUN useradd -c 'PHP user' -m -d /home/php -s /bin/bash php
35     USER php
36     ENV HOME /home/php
37
38 # xDebug configuration
39 ENV PHP_IDE_CONFIG serverName=nfqKickStartDocker
40
41 WORKDIR /code
42 VOLUME /code
```



Multiple containers

Development tools


PHP extensions

xDebug support

More configuration But infrastructure as a code

Branch: master ▾

[docker](#) / [docker-compose.yml](#)

 **aurelijusb** Small typo fix

1 contributor

25 lines (21 sloc) | 650 Bytes

```
1  version: "2"
2
3  services:
4    nginx.symfony:
5      container_name: nginx.symfony
6      build: nginx
7      volumes:
8        - ./examples:/code
9      ports:
10       - 127.0.0.1:80
11
12   php.symfony:
13     container_name: php.symfony
14     build: php
15     volumes:
16       - ./examples:/code
17
```

Branch: master ▾

[docker](#) / [php](#) / [Dockerfile](#)

 **aurelijusb** Updated versions to align with


1 contributor

43 lines (34 sloc) | 1.46 KB

```
1  FROM php:7.2.4-fpm
2
3  LABEL maintainer "Aurelijus Barlis"
4
5  WORKDIR /php
6
7  # Get composer: https://getcomposer.org/
8  RUN php -r "copy('https://getcomposer.org/installer',
9  RUN php -r "if (hash_file('SHA384', prevent Debian's PHP
10  RUN php composer-setup.php
11  RUN php -r "unlink('composer-setup.php');"
12  RUN ln -s /php/composer.phar /usr/bin/composer
13
```

Branch: master ▾

[php](#) / [7.2](#) / [stretch](#) / [fpm](#) / [Dockerfile](#)

 **docker-library-bot** Update to 7.2.11

3 contributors

259 lines (241 sloc) | 7.93 KB

```
1  #
2  # NOTE: THIS DOCKERFILE IS GENERATED VIA "update.sh"
3  #
4  # PLEASE DO NOT EDIT IT DIRECTLY.
5  #
6
7  FROM debian:stretch-slim
8
9  # prevent Debian's PHP packages from being installed
10 # https://github.com/docker-library/php/issues/121
11 RUN set -eux; \
12 { \
13
```

Branch: dist-amd64 ▾

[docker-debian-artifacts](#) / [stretch](#) / [slim](#) / [Dockerfile](#)

 **docker-library-bot** Update to 20180831 for amd64 (debuerreotype 0.8)

1 contributor

4 lines (3 sloc) | 46 Bytes

```
1  FROM scratch
2  ADD rootfs.tar.xz /
3  CMD ["bash"]
```

□ □ □ □

xDebug support

□ □ □ □

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS

Feature
toggle

Docker

IDE in
cloud

Virtual
box

Run &
pray

hometogo home24



Not many alternatives
For isolated environments

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS



Sandboxed development environment

High quality mocks (real MySQL, Wireshark)

True integration/acceptance tests

Dockerfile → docker-compose → custom tooling

Experimenting with new software safer

No trash, no sensitive information

Easy to swap (DynamoDB local vs Dynalite)

Install/compile on your machine? Seriously?



Switching between branches/tasks

Less issues with cache invalidation

Kill-it-not-heal-it

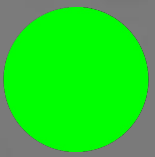
Data volumes = test data in branch

Infrastructure as a code

Fixed versions, fixed php.ini+extensions

Less “works on my machine” = reproducible

Many bash scripts, configuration via ENV



Huge ecosystem

Pioniers are using and improving
Amazon, Google (K8s) is investing

Known issues and solutions in StackOverflow

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS



High learning curve = blame docker

Misuse of the tool – isn't docker *silver bullet*?

Docker wraps – everyone blames the wrapper

Many ways to install docker incorrectly

Mounting vs copy-on-write operation

No Windows, more tooling/docs around



Not mature tooling / edge cases

Download private dependency: github token?

Password protected SSH key? (Mac+Linux)

CircleCI limited “remote docker”

Host IP for xDebug

Even more bash scripts/docs



Cache everything by design

Full HDD because of docker images

Missing log rotation (no “log-opts” by default)

“Latest” tag, that is not immutable

Network unreachable, since used by docker

Docker-clean sh, docker pull, RTFM

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

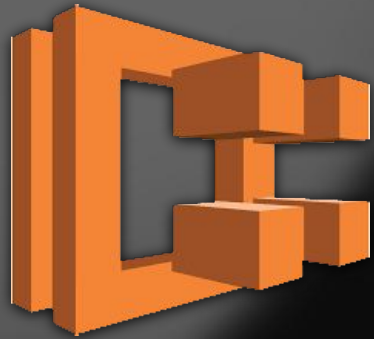
Production

Deploying to AWS

PHP + Docker in production

home 24

Migrated Tier1 service – nobody noticed



AWS ECS

PHP 7.2



docker

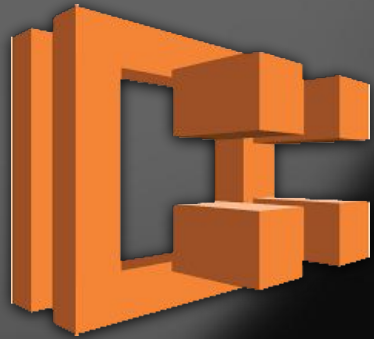
Docker

Apache



PHP + Docker in production

Migrated Tier1 service – nobody noticed



AWS ECS

PHP 7.2

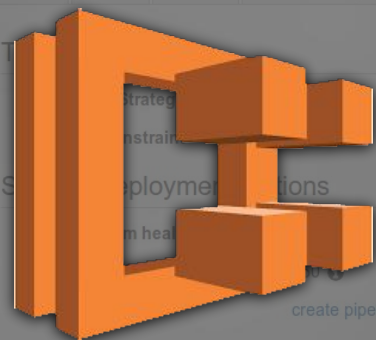


docker

Docker

Apache





Amazon Web Services Elastic Container Service

High quality AWS integration

K8s on AWS then was not mature

Know-how (our Go apps in prod)

Filter in this page

Deployment Id	Status	Desired Count	Running Count	Creation Time	Last Updated
ecs-svc/9223370495346757540	PRIMARY	1	0	2018-11-06 14:40:18 +0200	2018-11-06 14:40:18 +0200
ecs-svc/9223370495882532443	ACTIVE			2018-11-06 14:40:18 +0200	2018-11-06 14:40:18 +0200

Naming in AWS

Docker-compose → Task definition

Container → Task

Docker Hub → ECR

Run new Task Stop Stop All

Desired task status: Running Stopped

Filter in this page Launch type ALL

<input type="checkbox"/>	Task	Task Definition	Container Instance	Task Status	Desired status	Started By
<input type="checkbox"/>	83bf6032-c4f1-4247-...	-fluent-bit:5	155e69cd-4458-4fa3-...	RUNNING	RUNNING	ecs-svc/9223370495...
<input type="checkbox"/>	9ef5d30f-4171-43f4-a...	-hello-service:7	155e69cd-4458-4fa3-...	RUNNING	RUNNING	ecs-svc/9223370495...

Apache mod_php

PHP 7.2 log trimming in Nginx+FPM
Single process container – easier
Internal service (no slow loris attack)
No visible performance impact

Logging



fluentbit



elasticsearch

Replica

Daemon

External

<input type="checkbox"/>	Service Name	Status	Service type	Task Definition	Desired tasks	Running tasks	Launch type
<input type="checkbox"/>	ecs-ec2-service-1-407-MNJE2I6A	ACTIVE	REPLICAS	hello-service:9	1	1	EC2
<input type="checkbox"/>	ECSLoggerService-1RNU10SJSCB1P	ACTIVE	DAEMON	fluent-bit:5	1	1	EC2

Introduction

Why to dockerize
What is docker
How to dockerize

Development

Context/Alternatives
What I really liked
What I do not like

Production

Deploying to AWS

Feature
toggle

Docker

IDE in
cloud

Virtual
box

Run &
pray

Docker:

Good to understand

Use on demand

Thank you
Questions?

Aurelijus Banelis



Further reading/references

- <https://www.docker.com/>
- <https://hub.docker.com/>
- <https://aws.amazon.com/ecs/>
- <https://aws.amazon.com/ecr/>
- <https://fluentbit.io/>
- <https://www.home24.de/>
- <https://home24.tech.blog/category/aws/>
- <https://aws.amazon.com/blogs/opensource/network-load-balancer-support-in-kubernetes-1-9/>