Jan Mikeš

17.5.2024
#PHPLIVE2024

omnicado.com

myspeedpuzzling.com

peon.dev

peon.dev

Modern PHP Web Servers

@honza_mikes

omnicado.com

Sell from your e-shop on all marketplaces and boost your sales.

MySpeedPuzzling.com

World first speed puzzling times database.
github.com/MySpeedPuzzling

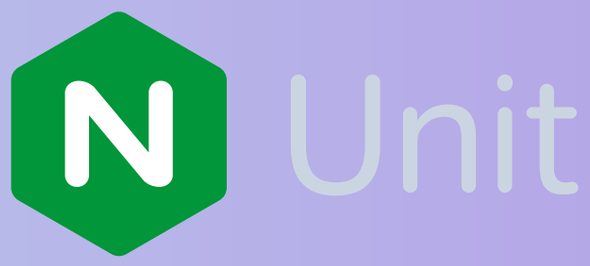
peon.dev

Save your time. Automation for developer's daily tasks.
github.com/peon-dev


APACHE


NGINX


caddy


Unit


FRANKENPHP

Reminder

- PHP is interpreted language (Zend Engine interpreter is written in C)
- Webserver needs SAPI (Server Application Programming Interface) - interface between webserver and PHP Interpreter
- Embed SAPI, for example Apache (mod_php)
- SAPI can be external like PHP-FPM communicating over FastCGI protocol

README.md

The embed SAPI

A server application programming interface (SAPI) is the entry point into the Zend Engine. The embed SAPI is a lightweight SAPI for calling into the Zend Engine from C or other languages that have C bindings.

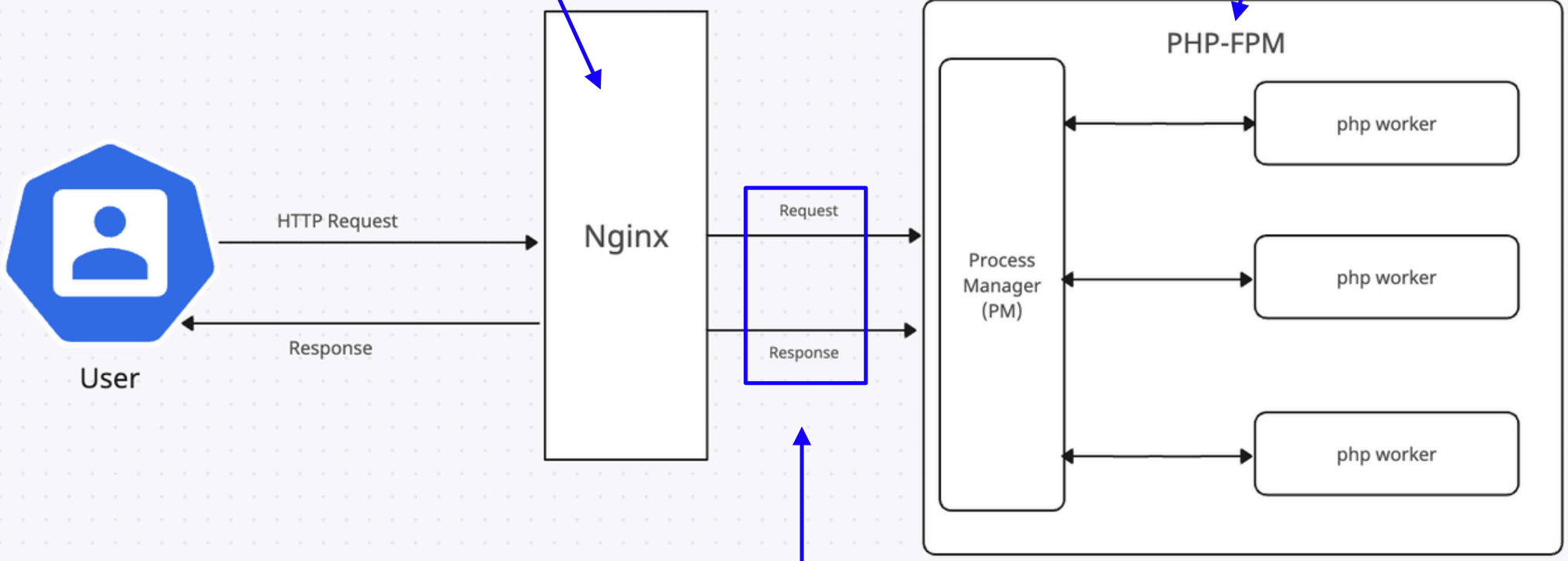
Basic Example

Below is a basic example in C that uses the embed SAPI to boot up the Zend Engine, start a request, and print the number of functions loaded in the function table.

```c
/* embed_sapi_basic_example.c */

#include <sapi/embed/php_embed.h>

int main(int argc, char **argv)
{
        /* Invokes the Zend Engine initialization phase: SAPI (SINIT), modules
         * (MINIT), and request (RINIT). It also opens a 'zend_try' block to catch
         * a zend_bailout().
         */
        PHP_EMBED_START_BLOCK(argc, argv)

        php_printf(
                "Number of functions loaded: %d\n",
                zend_hash_num_elements(EG(function_table))
        );

        /* Close the 'zend_try' block and invoke the shutdown phase: request
         * (RSHUTDOWN), modules (MSHUTDOWN), and SAPI (SSHUTDOWN).
         */
        PHP_EMBED_END_BLOCK()
}
```

Web server

SAPI



FastCGI protocol

/etc/php/8.3/fpm/pool.d/www.conf

```
[www]
user = www-data
group = www-data

listen = 127.0.0.1:9000
```

Webserver expectations

- Stability
- Performance
- Easy to configure + run
- PHP 8.3
- Open source
- Good documentation
- Community
- Cloud friendly
- Observability (prometheus metrics)

APACHE

- Since 1995 (29 years)
- SAPI: PHP-FPM / embed mod_php, single container
- Stable
- Written in C
- Not so great performance
- Legendary .htaccess
- Painful to configure

/etc/apache2/conf.d/example.com.conf

```
<VirtualHost *:80>
    ServerName example.com

    <FilesMatch \.php$>
        SetHandler proxy:fcgi://127.0.0.1:9000
    </FilesMatch>

    DocumentRoot /var/www/project/public
    <Directory /var/www/project/public>
        AllowOverride None
        Require all granted
        FallbackResource /index.php
    </Directory>
</VirtualHost>
```

SERVED HERE

NGIN

A PERFECT MATCH

- Since 2004 (20 years)
- SAPI: PHP-FPM, external, not so container friendly
- Industry standard
- Written in C
- Great performance, awesome reverse-proxy
- Complex

/etc/nginx/conf.d/example.com.conf

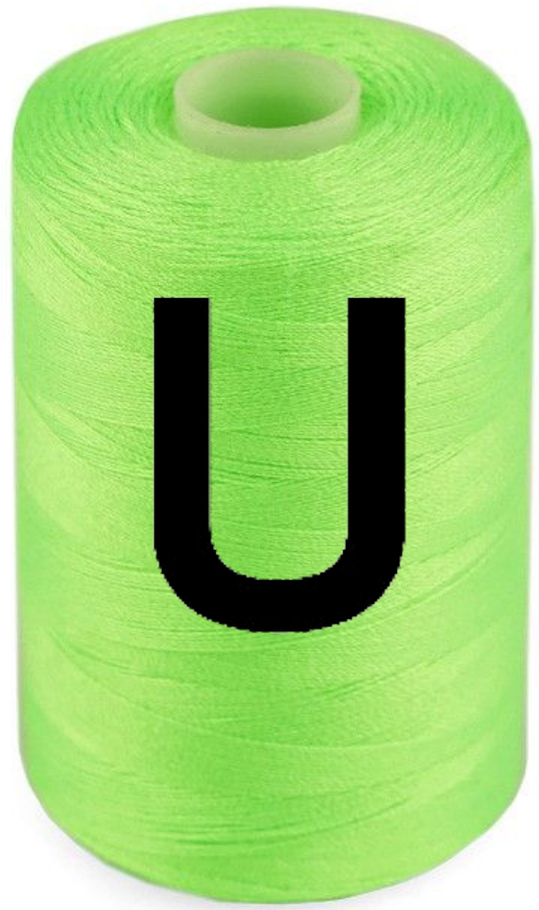
```
server {
    server_name example.com;
    root /var/www/project/public;

    location / {
        try_files $uri /index.php$is_args$args;
    }

    location ~ ^/index\.php(/|$) {
        fastcgi_pass 127.0.0.1:9000;

        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
    }
}
```

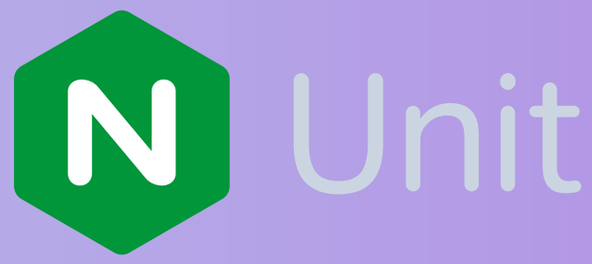
caddy®

- Since 2015 (9 years)
- SAPI: PHP-FPM, external, not so container friendly
- Automatic SSL
- Nice reverse proxy
- Zero downtime config reloads - Caddyfile, REST API
- Implicit HTTP 2/3 support
- Defaults are good enough for production
- Great community, Modularity, Written in GO
- Slower than Nginx
- Super easy to start quickly

/etc/caddy/Caddyfile

```
example.com {
	root * /var/www/project/public

	php_fastcgi 127.0.0.1:9001
	file_server
}
```


U



- Since 2017 (7 years)
- Written in C
- Very capable reverse-proxy
- Dynamic multi-language support
  (PHP, Python, Node, GO, Java, Perl, Ruby, WebAssembly)
- Configuration via REST API -> OpenAPI spec + app
- Levels of routing, Listener -> Route -> Application/Upstream

- Great performance and stability, low resources footprint
- Zero downtime reloads
- NJS scripting - powerful routing
- Quite easy TLS support

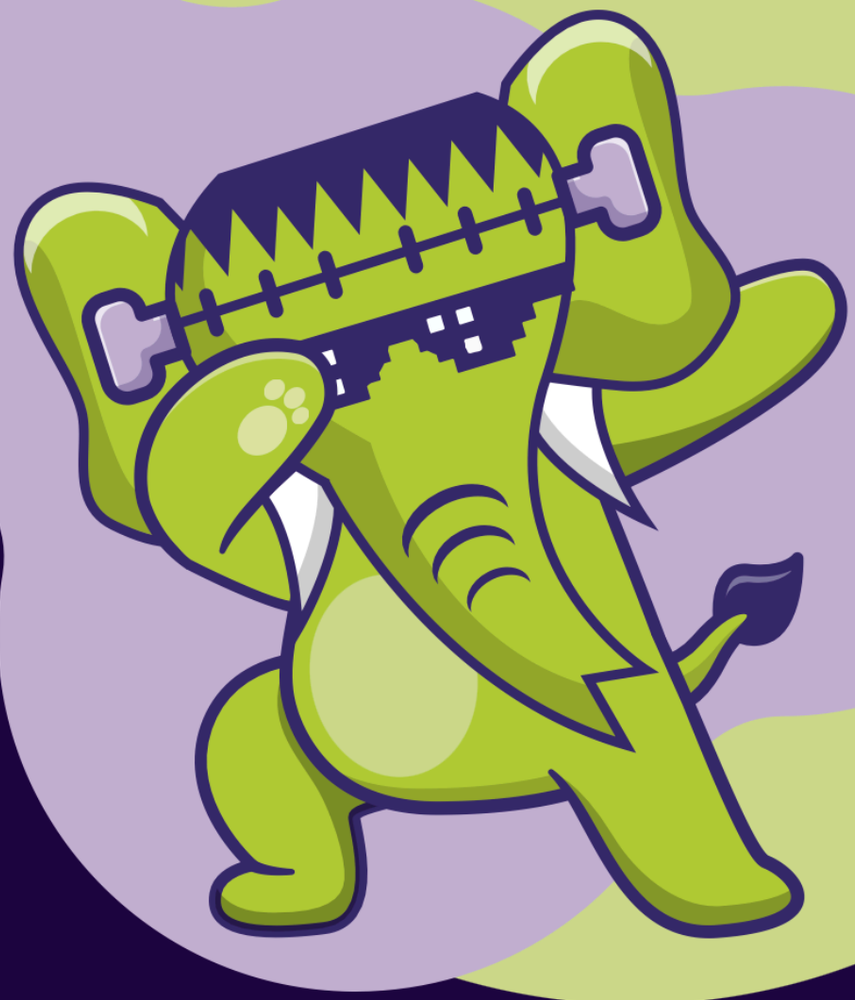
- Security
- Prometheus not natively supported
- Little bit slower release cycle than expected
- Some advanced features might be missing

/etc/unit/config.json

```
{
    "listeners": {
        "*:80": {
            "pass": "applications/my-app"
        }
    },

    "applications": {
        "my-app": {
            "type": "php",
            "root": "/var/www/project/public/"
        }
    }
}
```

```
curl -X PUT \
     --data-binary @config.json \
     --unix-socket /var/run/unit/control.sock \
     http://localhost/config
```

FRANKENPHP



Kévin Dunglas

- https://Vulcain.rocks
- https://Mercure.rocks
- https://api-platform.com
- Symfony components

FrankenPHP

- Caddy + FrankenPHP + static-php-cli (optional)
- Caddy: WebServer
- FrankenPHP: SAPI Embed into Caddy
- static-php-cli: statically compiles PHP binaries

FrankenPHP

- All the Caddy offers (HTTP 2/3, SSL, Zero downtime)
- Embed SAPI!! Written in GO using //go:embed
- 103 early hints
- Worker mode (super-fast!)
- Integrated Mercure Hub (real-time apps)
- Convention over configuration !!!
- Self-contained single binary app
- Native support for Symfony, Laravel, API Platform, Drupal, Wordpress
- Bus factor 1
- Stability?

/etc/caddy/Caddyfile

```
{
	frankenphp
	order php_server before file_server
}

example.com {
	root * /var/www/project/public
	php_server
}
```

```
docker run \
    -p 80:80 -p 443:443 -p 443:443/udp \
    -v $PWD:/app \
    dunglas/frankenphp
```

docker-compose.yml

```yaml
services:
  php:
    image: dunglas/frankenphp
    ports:
      - "80:80" # HTTP
      - "443:443" # HTTPS
      - "443:443/udp" # HTTP/3
    volumes:
      - ./:/app/public
      - caddy_data:/data
      - caddy_config:/config
    tty: true

volumes:
  caddy_data:
  caddy_config:
```

Dockerfile

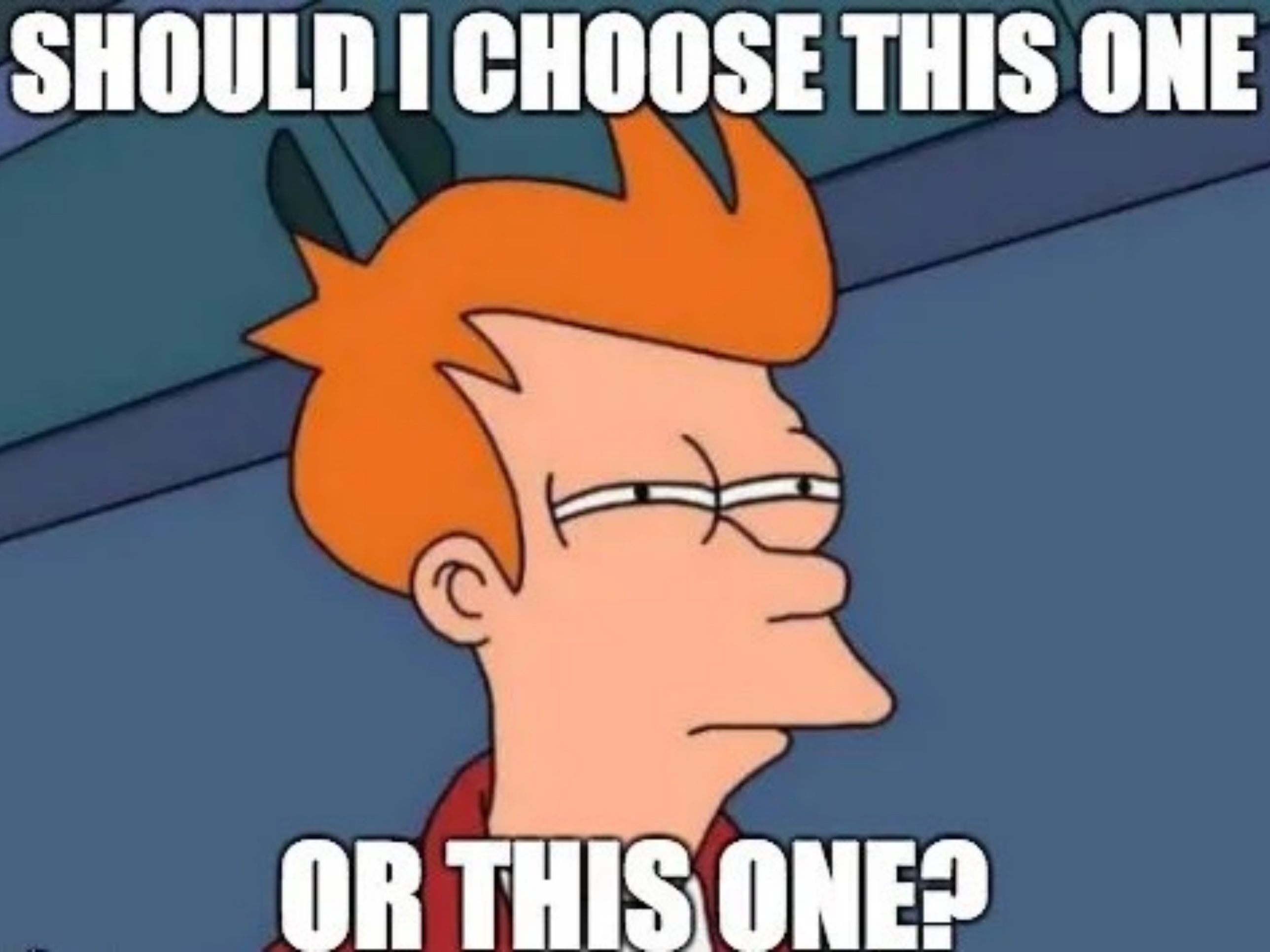
```
1  FROM dunglas/frankenphp
2
3  COPY . /app/public
```

Dockerfile

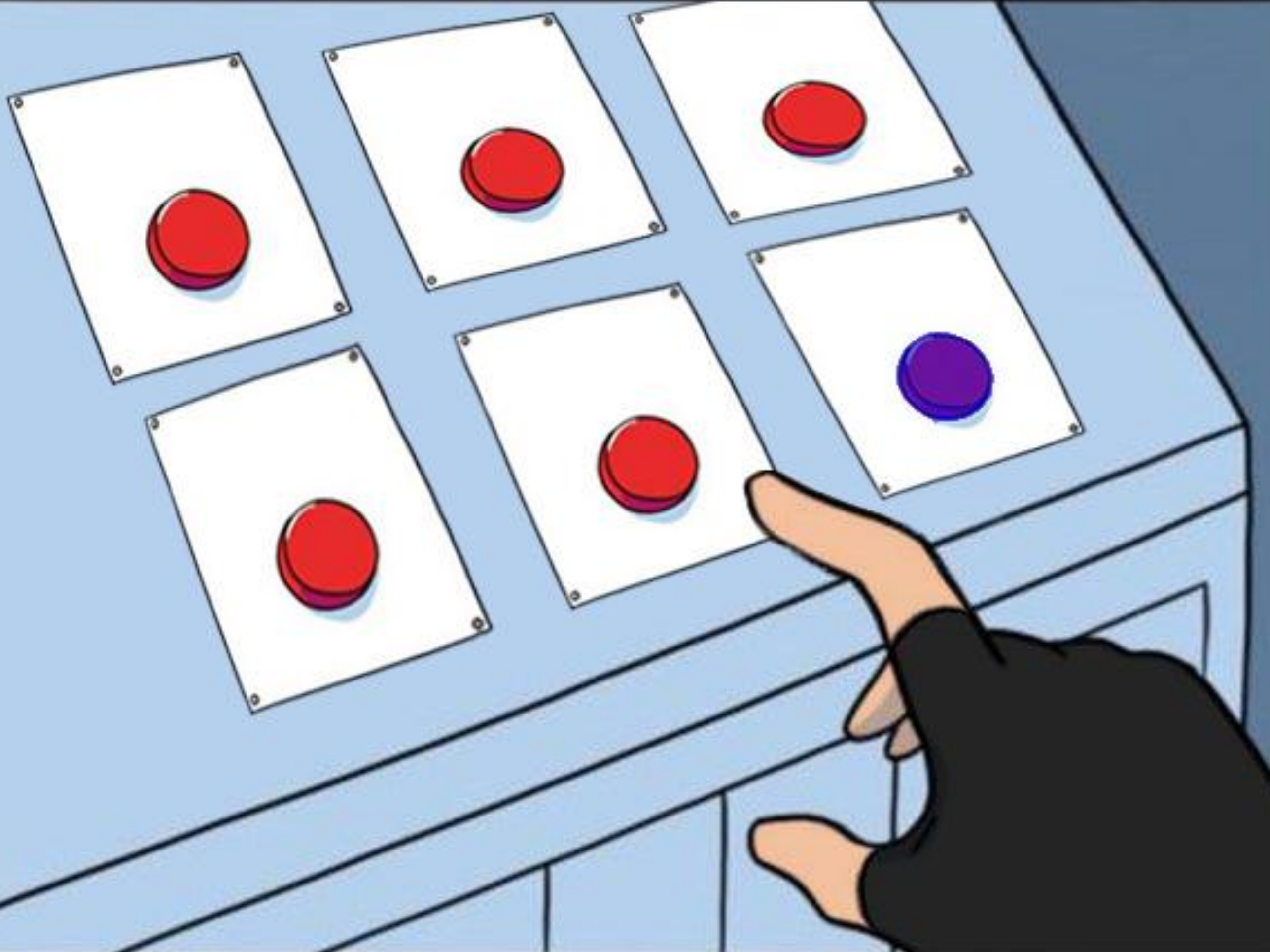
```
FROM dunglas/frankenphp
COPY . /app/public

# need more extensions?
RUN install-php-extensions \
    pdo_mysql \
    gd \
    intl
```

```
./my-app php-server --domain example.com

./my-app php-cli bin/console
```

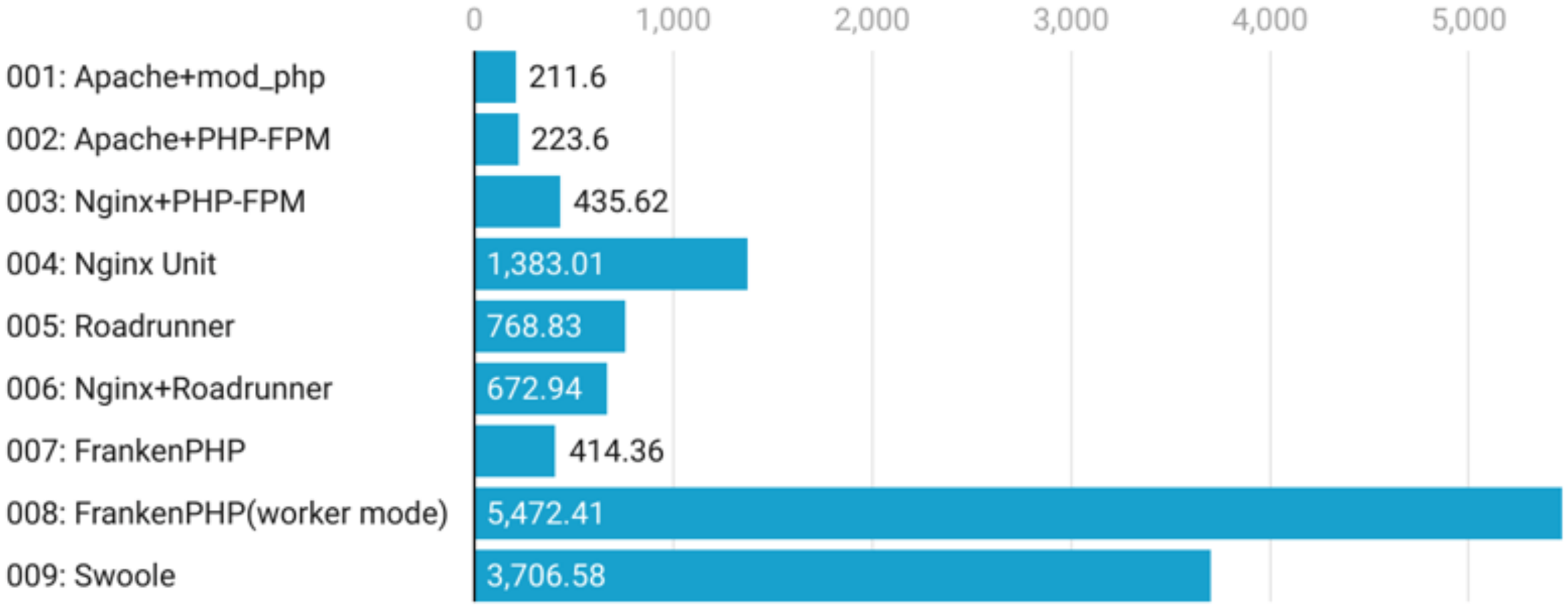
SHOULD I CHOOSE THIS ONE



OR THIS ONE?



Average requests per second, concurrency 1000

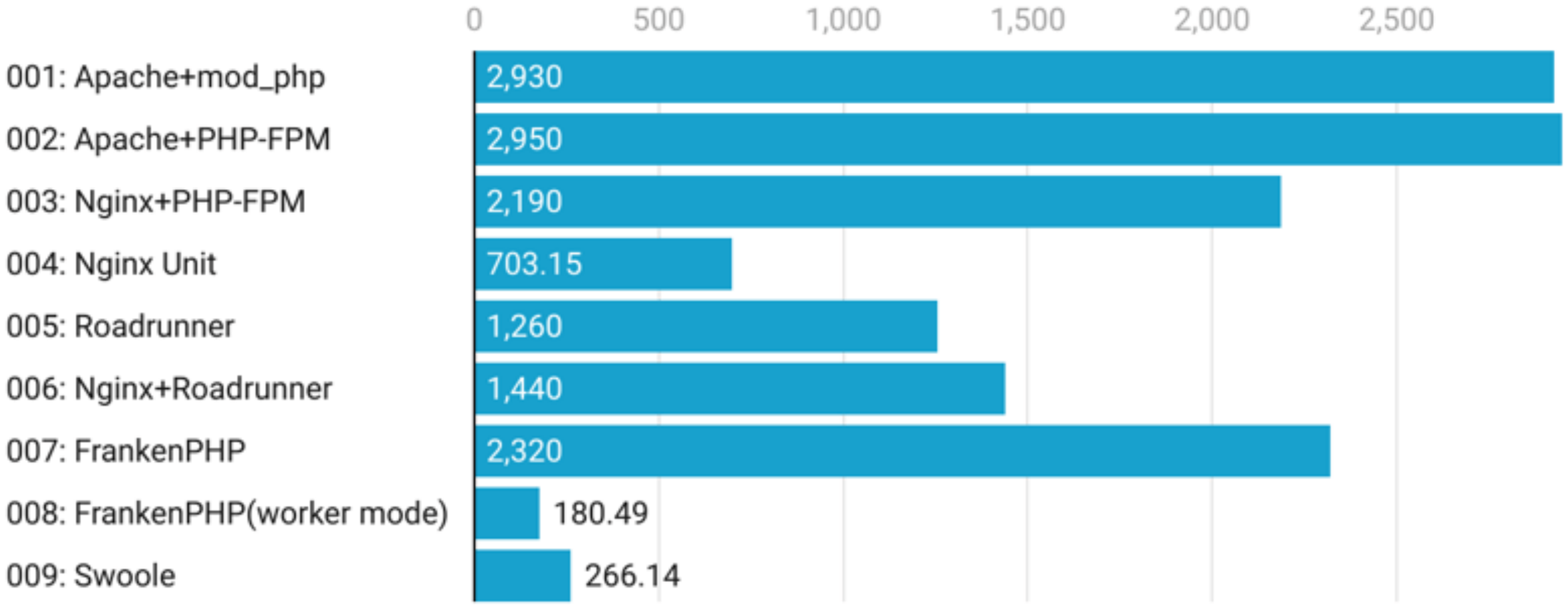


Docker container: CPU: 1, RAM: 1GB. Testing utility: K6(https://k6.io/open-source/)

Source: GitHub • Created with Datawrapper

Source: https://dev.to/dimdev/performance-benchmark-of-php-runtimes-2lmc

Average response time(ms), concurrency 1000



Docker container: CPU: 1, RAM: 1GB. Testing utility: K6(https://k6.io/open-source/)

Source: GitHub • Created with Datawrapper

Source: https://dev.to/dimdev/performance-benchmark-of-php-runtimes-2lmc

Stress test

- Unit vs Nginx+FPM vs Octane (Swoole)
- Unit was the only web server that did not drop a single request
- Unit is very fast, low resources usage, very fast cold starts

Source: https://habr.com/en/articles/646397/

- Nginx vs Apache
  - More performant
  - Load balancing + reverse proxy
  - Apache consumes more resources due to its process-driven architecture
- Nginx vs Caddy
  - Caddy is the extremely easy to use and start quickly
  - Nginx is more performant and industry standard
  - Caddy compared to Nginx has smaller ecosystem at the moment
  - Caddy is the go-to choice for begginers or small sites for its simplicity and easy of use

- Unit
  - Very fast and realiable, go for it!
- FrankenPHP
  - Supercool, go for it!

- https://github.com/janmikes/phplive-2024
- https://caddyserver.com
- http://unit.nginx.org
- https://frankenphp.dev
- https://static-php.dev

THANK YOU!

@honza_mikes

janmikes.cz