

Accélérez vos applications PHP avec FrankenPHP



AFUP Montpellier - 19 nov. 2024



Stéphane WOUTERS

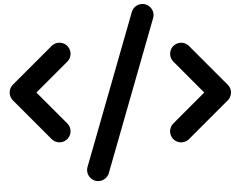
afup  MONTPELLIER



Stéphane WOUTERS

Développeur Full stack
Montpellier

<https://stephanewouters.fr/>



Développeur web passionné

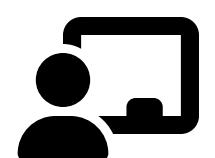
Full stack depuis 15+ ans.

*PHP, PostgreSQL, NodeJS, Golang
React, React Native
Docker, AWS*



Lead Développeur

Agence web Poisson Soluble depuis 2015



Formateur

École EPSI pour étudiants post-bac



poisson
soluble

POISSON SOLUBLE

Agence web

Développement des outils web de la société **Predict Services**, dans le domaine du **risque hydro-météorologique**.

Notre stack technique

- Architecture **microservice**
 - **Backends Symfony** orientés traitement/API
 - **150+ repos PHP à héberger**
- Front-end **Angular**
- Hébergement **Docker Swarm**

FrankenPHP

Le serveur HTTP moderne écrit en GO



FrankenPHP

Le serveur HTTP moderne écrit en GO



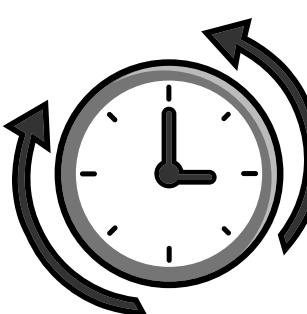
Remplace Nginx/Apache

Avec support du HTTP/2 et HTTP/3.



Facile à déployer

Toute votre app dans une seule image docker ou un binaire standalone.



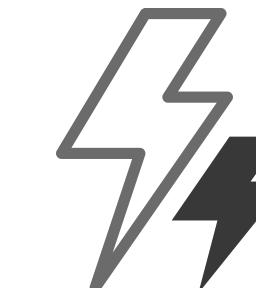
Événements temps réel

HTTP Push avec Mercure Hub intégré.



3.5x plus rapide

Grâce au mode Worker.



Support des 103 Early Hints

Envoyer les assets au browser avant de calculer le contenu de la page.



HTTPS intégré

Génération automatique du certificat SSL.

FrankenPHP



Créé par Kévin Dunglas

- Core team Symfony
- Créeur de API Platform et de Mercure
- Contributeur Go/Caddy

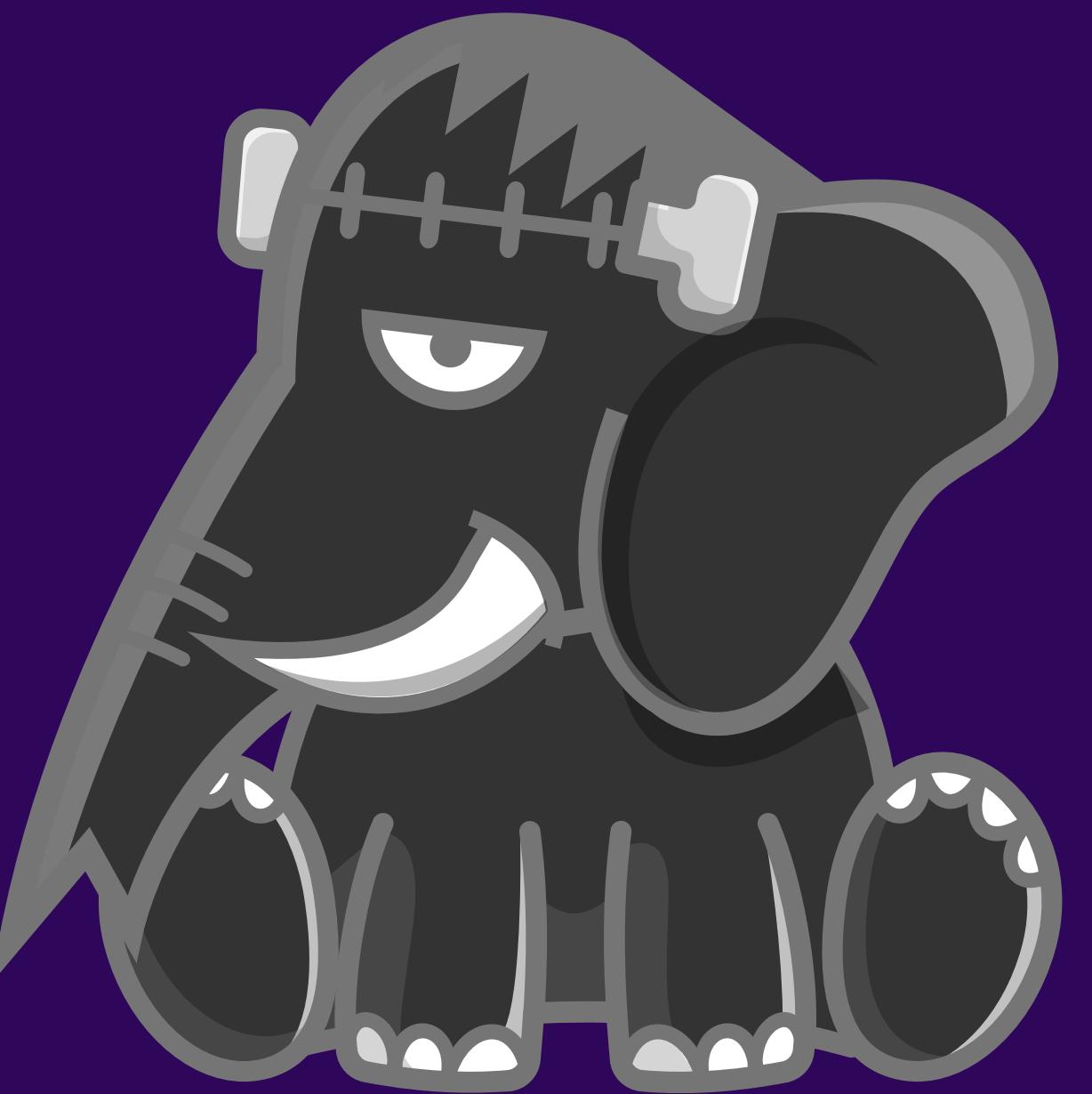


FrankenPHP

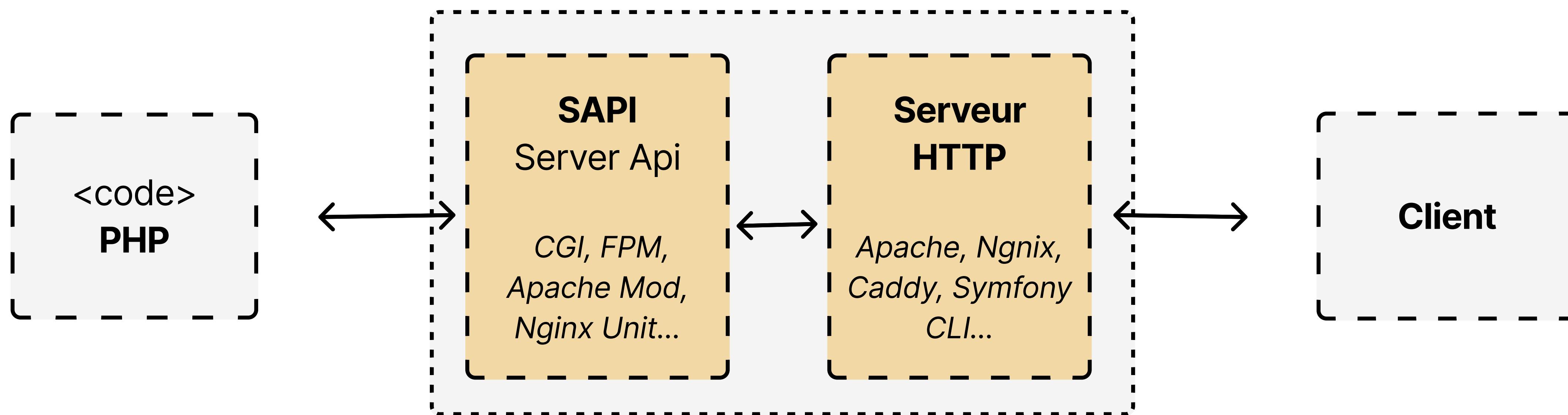
- Présenté une première fois au [ForumPHP Paris 2022](#)
- **Version 1.0** sortie en décembre 2023
- **Mises à jour régulières**
- **Aujourd'hui en 1.3** et déjà utilisé en production

Serveur HTTP et SAPI

Fonctionnement et solutions disponibles



Serveur HTTP et SAPI



Solutions disponibles

Avant FrankenPHP



Apache Module

Solution par défaut, la plus simple à mettre en place

Moins rapide que FPM



FPM

FastCGI Process Manager

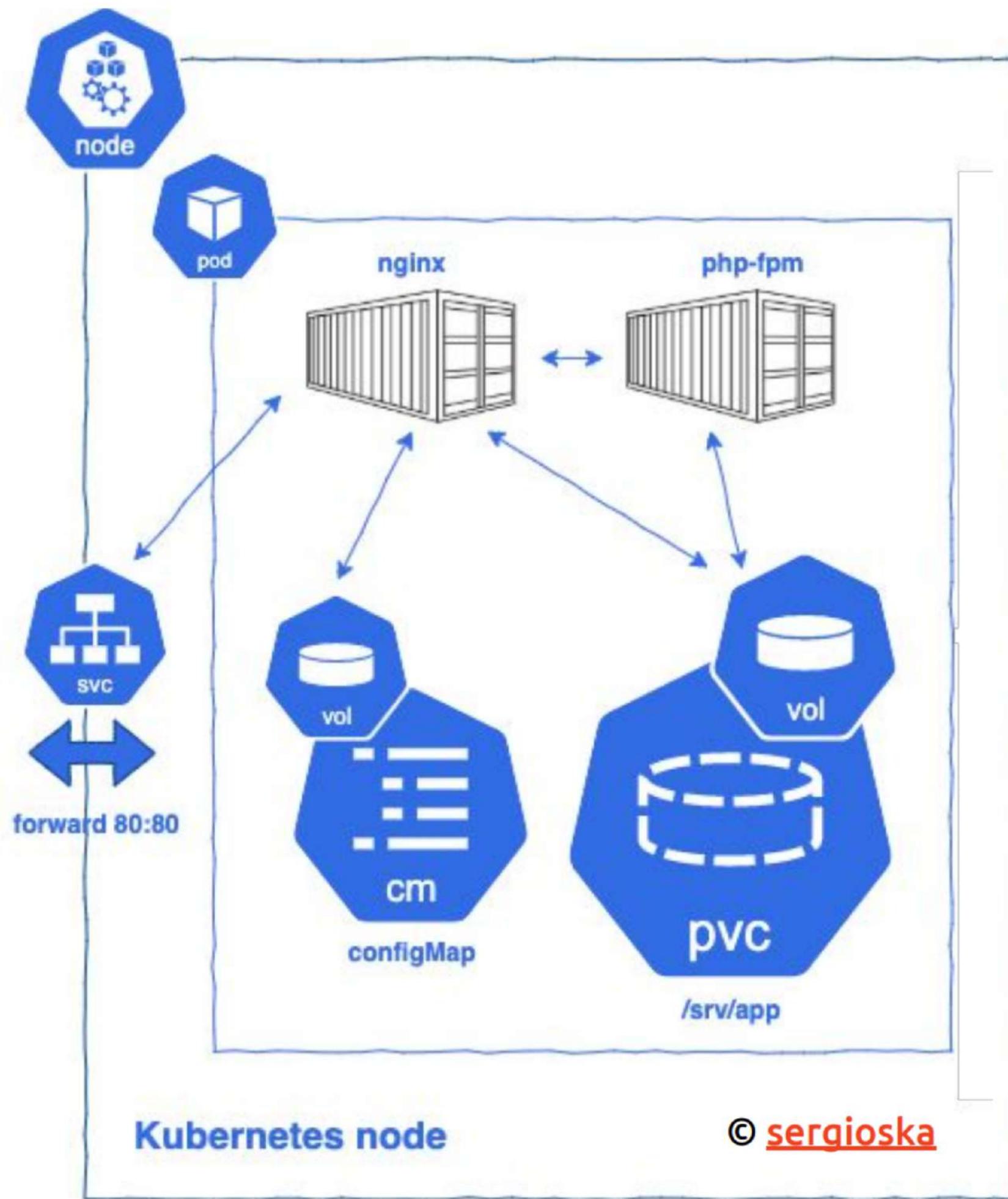
Sépare le serveur HTTP de la SAPI

Fonctionne avec Apache, Nginx, Caddy, Symfony CLI...

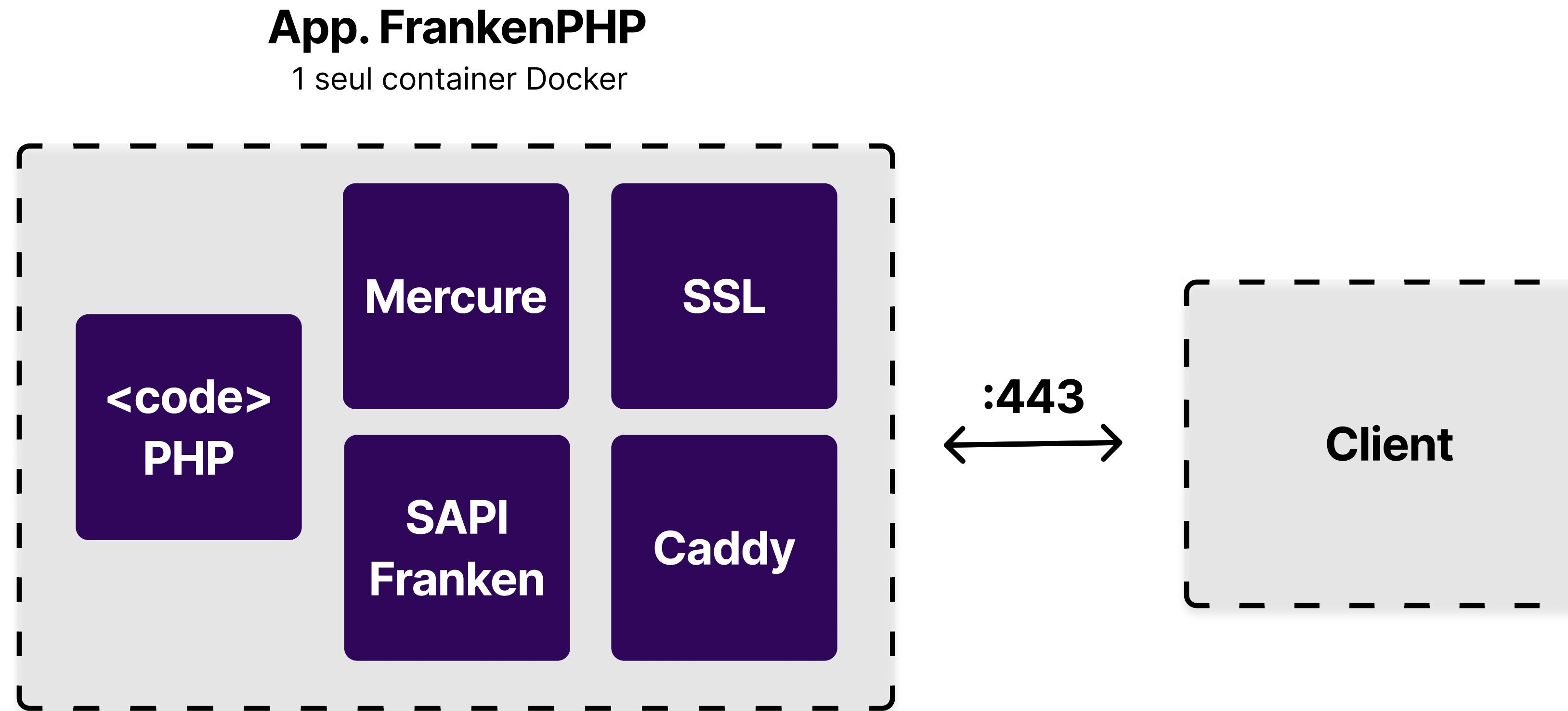
Difficile à dockeriser

Nginx + PHP FPM

Mise en place avec Docker



Déploiement avec FrankenPHP



SAPI Performances

Nombre de requête / seconde

211 req/s

Apache mod.

223 req/s

Apache + FPM

435 req/s

Nginx + FPM

414 req/s

FrankenPHP
Sans mode worker



SAPI Performances

Nombre de requête / seconde

211 req/s

Apache mod.

223 req/s

Apache + FPM

435 req/s

Nginx + FPM

414 req/s

FrankenPHP
Sans mode worker

5472 req/s

FrankenPHP
Mode worker

SAPI Performances

Temps de réponse moyen

26ms

Apache mod.

28ms

Apache + FPM

30ms

Nginx + FPM

26ms

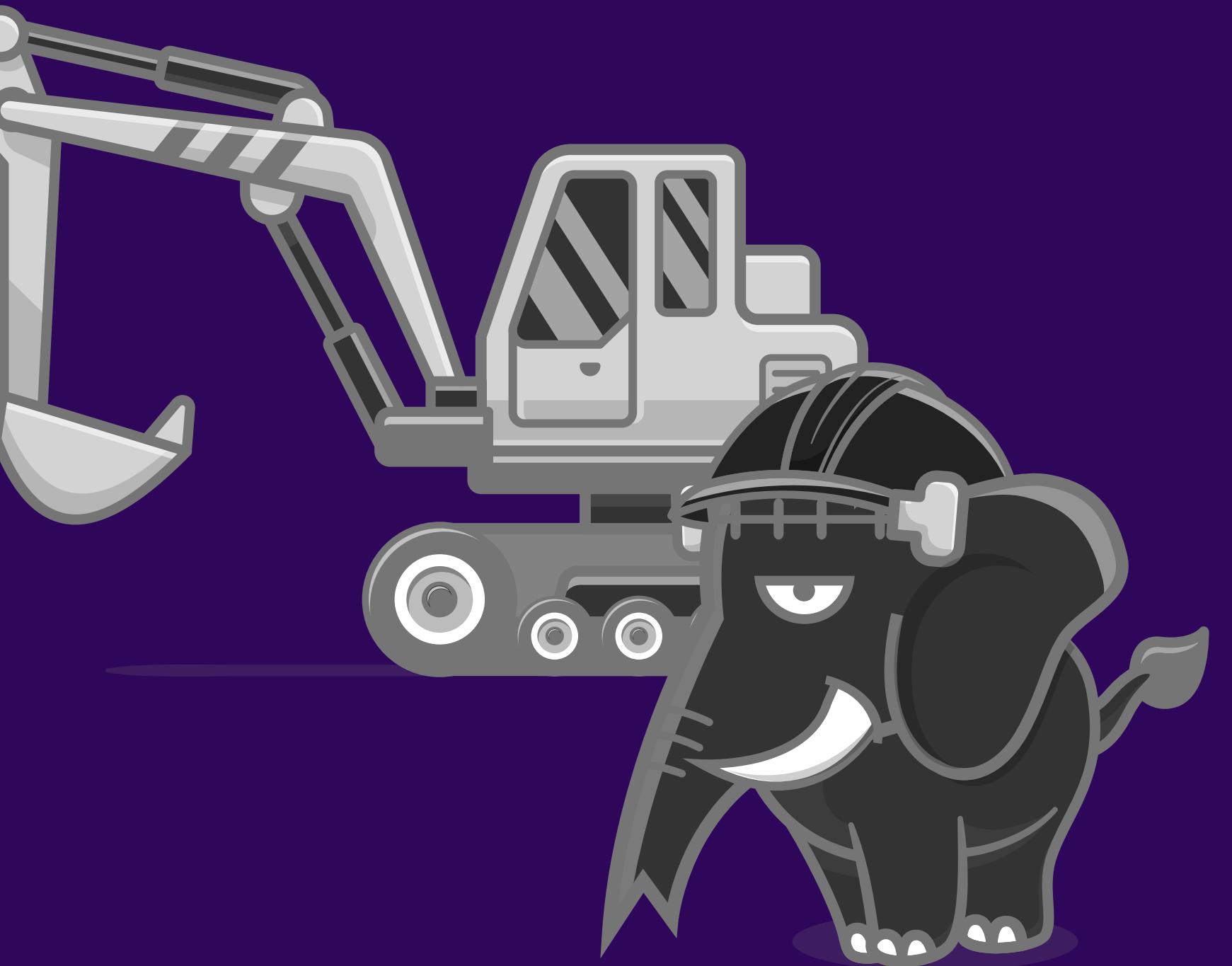
FrankenPHP
Sans mode worker

4ms

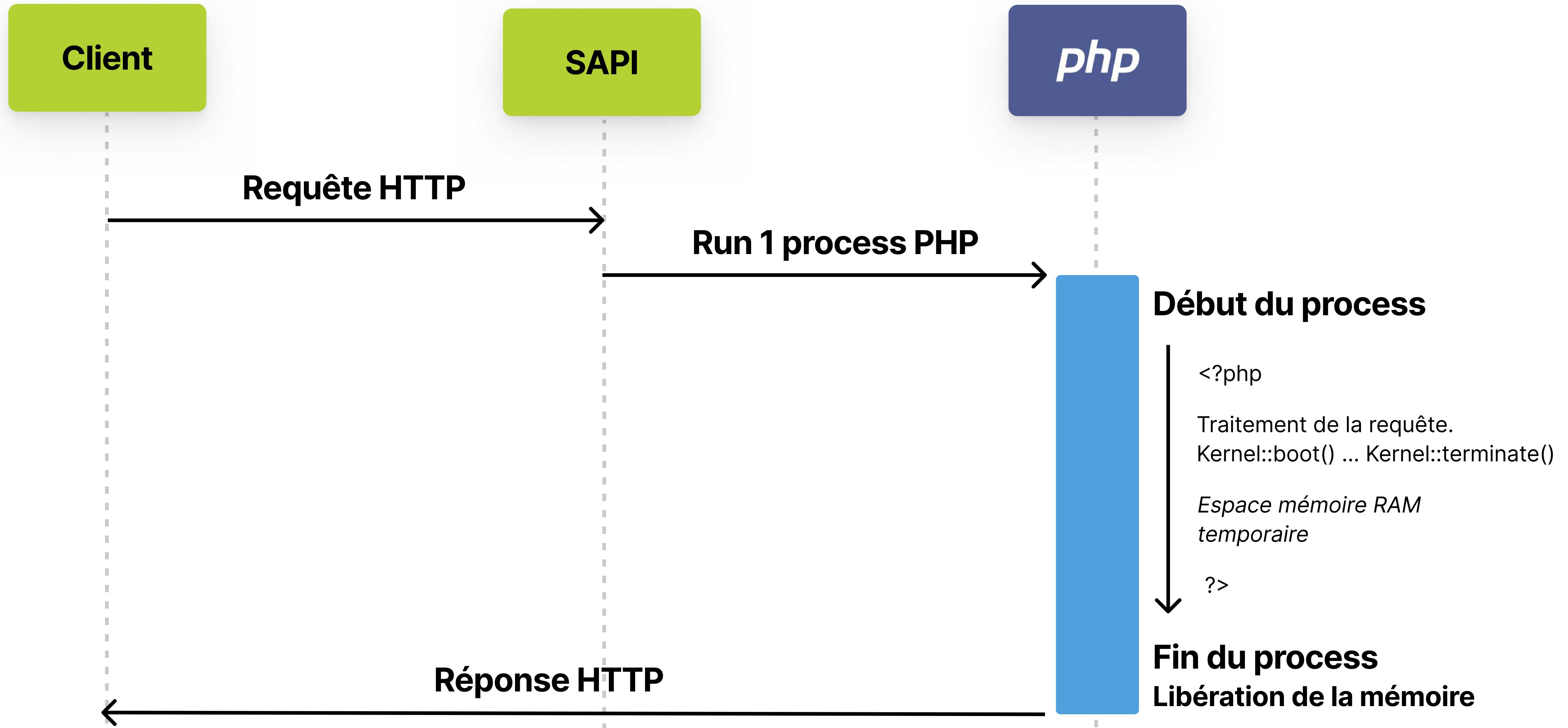
FrankenPHP
mode worker

Mode Worker

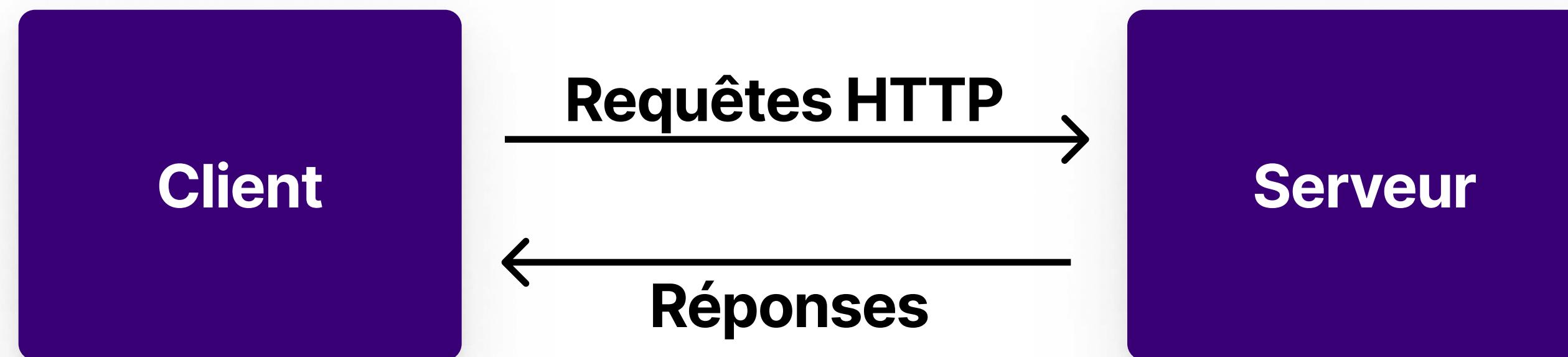
Et la révolution commence



Quand j'étais petit, j'ai appris que...



Chez nos voisins



1 seul programme lancé

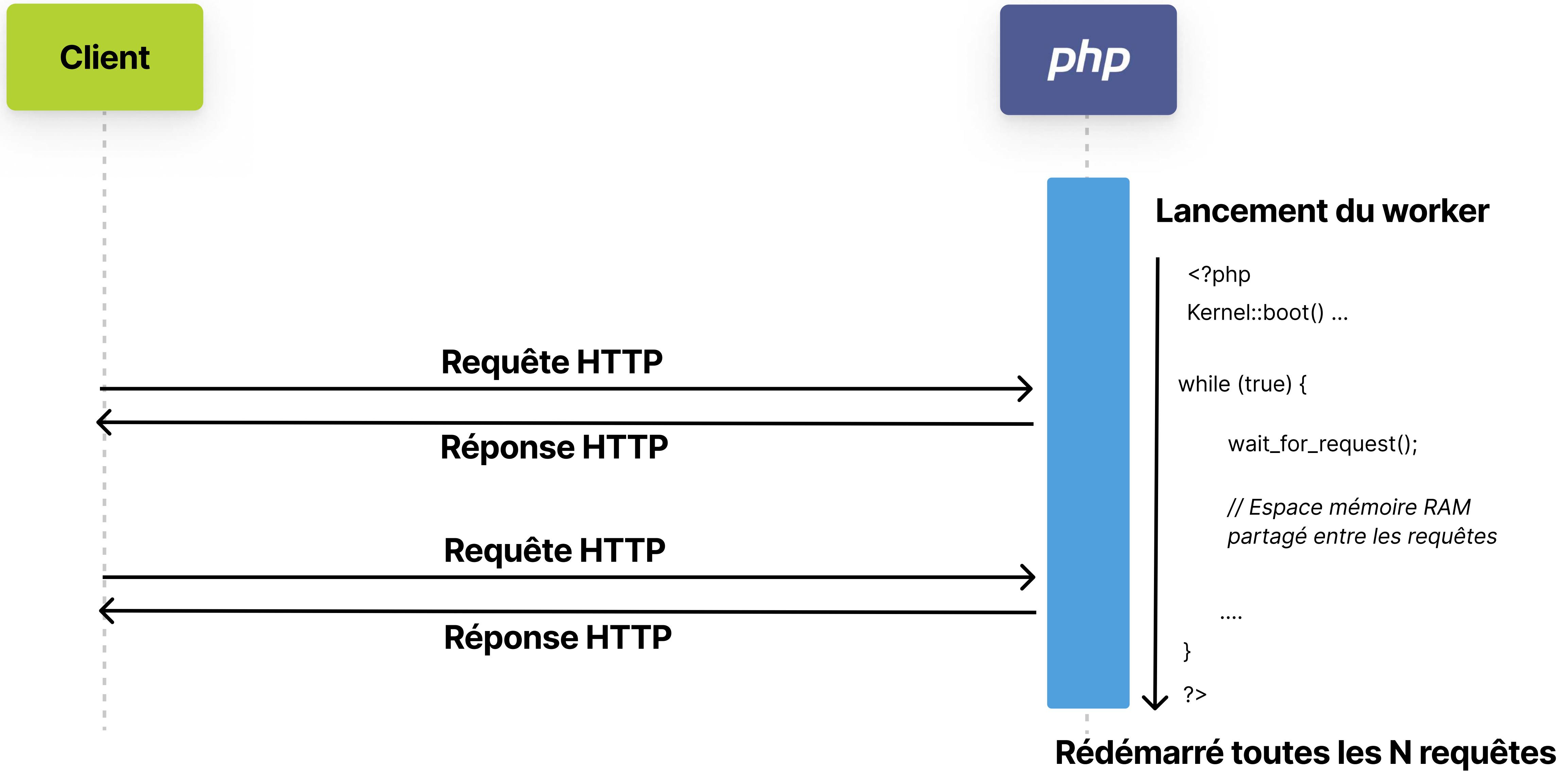
\$> node server.js

Mémoire RAM partagée
entre les requêtes



Et si on faisait comme eux ?

Mode worker de FrankenPHP



Démo effet de bord



```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class CounterController extends AbstractController
{
    private static int $counter = 0;

    #[Route("/counter")]
    public function counter()
    {
        self::$counter++;

        $counter = self::$counter;
        return new Response("Counter: $counter");
    }
}
```

Hébergement classique avec Apache Mod.

<https://frankenphp-demo-apache.doelia.fr/counter>

Hébergement avec FrankenPHP mode worker

<https://frankenphp-demo.doelia.fr/counter>

Démo effet de bord

```
<?php  
  
namespace App\Controller;  
  
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\Routing\Attribute\Route;  
  
class CounterController extends AbstractController  
{  
    private static int $counter = 0;  
  
    #[Route("/counter")]  
    public function counter()  
    {  
        self::$counter++;  
  
        $counter = self::$counter;  
        return new Response("Counter: $counter");  
    }  
}
```

Hébergement classique avec Apache Mod.

<https://frankenphp-demo-apache.doeilia.fr/counter>

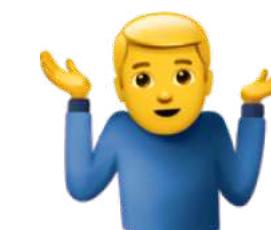
Le compteur sur la page sera toujours "1"

Hébergement avec FrankenPHP mode worker

<https://frankenphp-demo.doeilia.fr/counter>

Le compteur sera incrémenté à chaque F5
(ou presque)

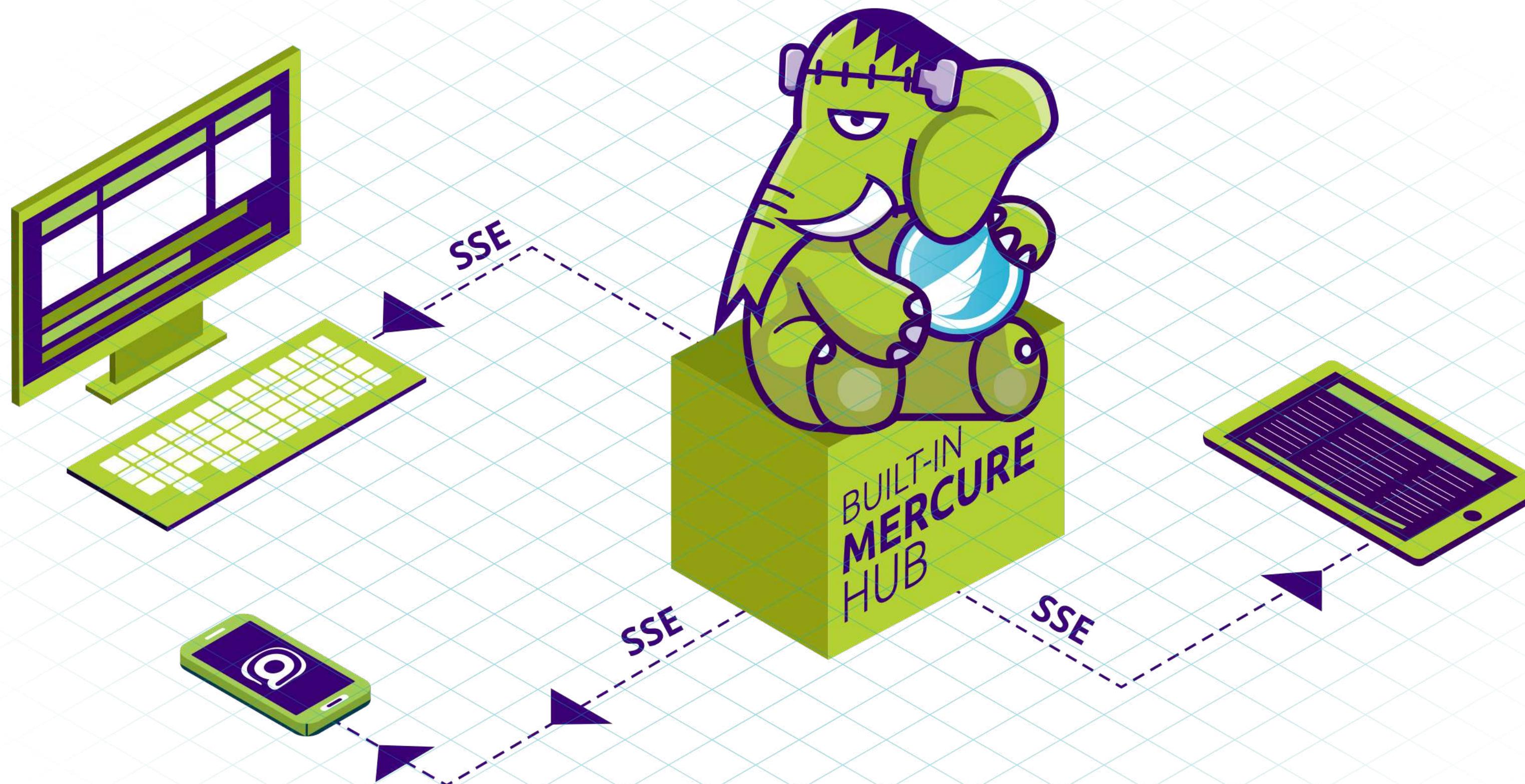
Quel est le mieux ?



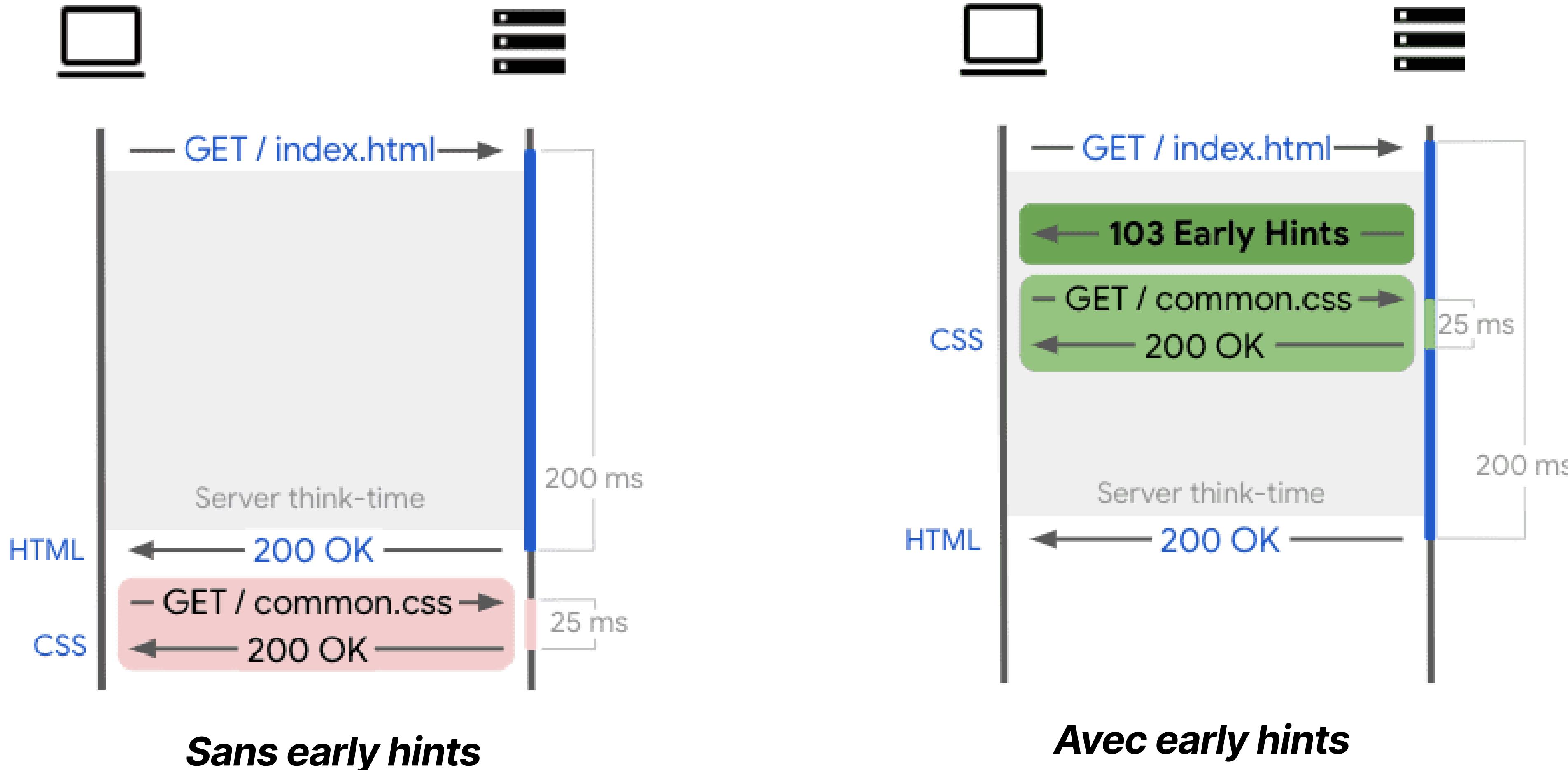
Autres fonctionnalités



Temps réel intégré avec Mercure



Early Hints



<https://frankenphp.dev/docs/early-hints/>

Standalone binary

Embarquez votre application PHP en un binaire portable

```
FROM --platform=linux/amd64 dunglas/frankenphp:static-builder

# Copy your app
WORKDIR /go/src/app/dist/app
COPY . .

# Build the static binary
WORKDIR /go/src/app/
RUN EMBED=dist/app/ ./build-static.sh
```

```
$> ./my-app php-server
```

<https://frankenphp.dev/docs/embed/>

Mise en production

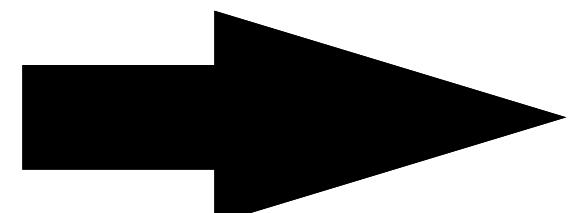
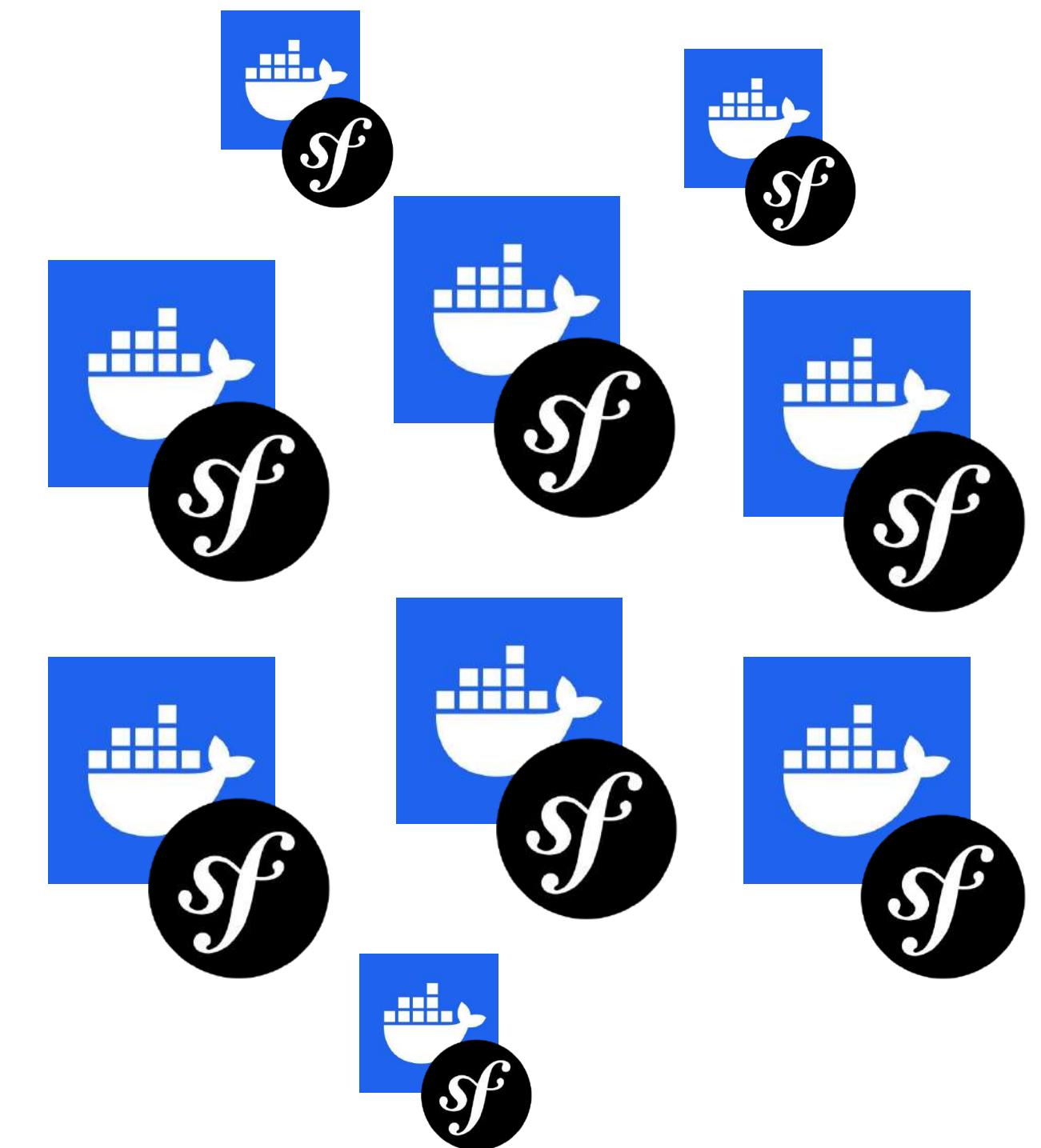
Ça passe ou ça casse ?



Contexte

Notre environnement chez Poisson Soluble avant FrankenPHP

- **Micro service**, 1 image docker par repo git
- Framework **Symfony LTS** (5.4 ou 6.4)
- Connexion SQL à un cluster **Postgres** (PgPool)
- Peu de dépendances / code dans chaque micro service
- Des microservices moins critiques que d'autres
- Docker Base image **FROM php:8.3-apache**
- Hébergement sur un cluster **Docker Swarm** (12cpu 16go x2)
- Proxy HTTP **Traefik**



Migration vers FrankenPHP en douceur

Installation sur Symfony

Installation du mode worker

```
$> composer require runtime/frankenphp-symfony
```

+ Review du code pour vérifier la compatibilité Worker

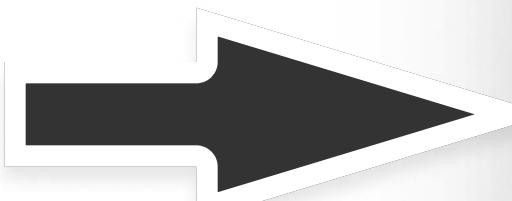
C'est tout côté Symfony...

Dockerfile

Évolution par rapport à utilisation de php-apache

Construction ISO avec FROM:php-apache

Nous n'avons rien modifié sur la construction du /app



```
FROM dunglas/frankenphp:latest-php8.3 as build_php  
  
COPY --from=composer/composer:2-bin /composer /usr/bin/composer  
  
COPY bin bin  
COPY config config  
COPY public public  
COPY src src  
COPY .env .env  
  
COPY composer.json composer.json  
COPY composer.lock composer.lock  
COPY symfony.lock symfony.lock  
  
ENV APP_ENV="prod"  
RUN composer install --optimize-autoloader --no-scripts --no-dev  
RUN php bin/console cache:clear
```



```
FROM dunglas/frankenphp:latest-php8.3-alpine  
  
COPY --from=build_php /app /app  
  
ENV FRANKENPHP_CONFIG="worker ./public/index.php 2"  
ENV CADDY_GLOBAL_OPTIONS="auto_https off"  
ENV SERVER_NAME=":80"  
ENV APP_RUNTIME="Runtime\\FrankenPhpSymfony\\Runtime"  
ENV APP_ENV="prod"
```

Possibilité d'utiliser alpine

Le poids de nos images docker ont diminué de 70%

Désactivation du SSL

Nous laissons notre proxy gérer le HTTPS

Progression vers la prod



Choix d'un microservice peu sensible et avec peu de dépendances

Traitement API, pas de BDD



Migration de l'app vers Franken en quelques heures

Tests OK



Exécution en recette pendant plusieurs jours.

Aucune erreur remontée



Passage en production

Uptime +30d, tout est OK

Gain en performance

Gain x3.5 en temps de réponse par requête

100ms
/req

Projet PHP Apache

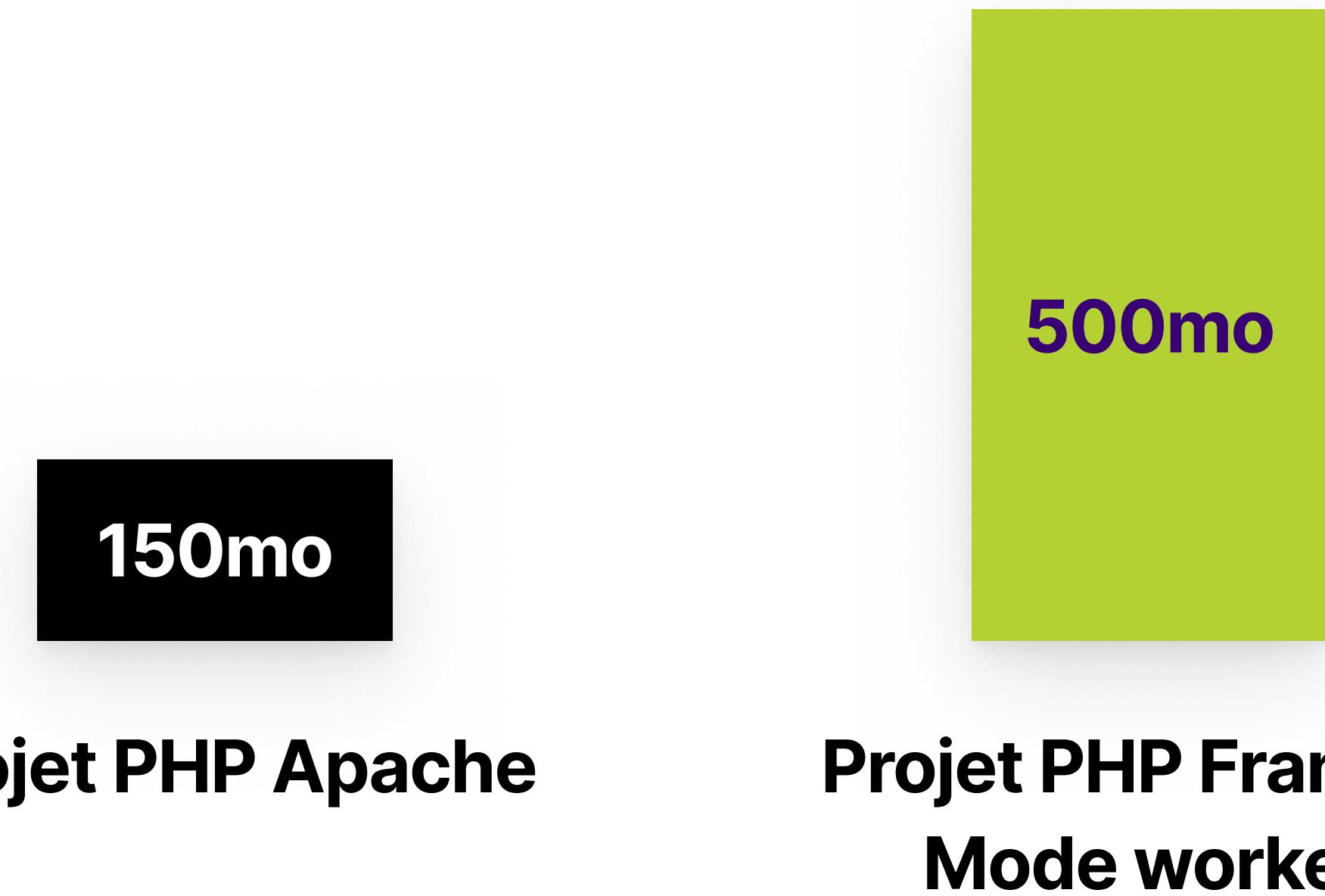
30ms/req

Projet PHP Franken
Mode worker

*App Symfony - Docker Swarm 12 cpu 16go
Proxy Front Nginx + Proxy Docker Traefik*

Consommation de mémoire

Évolution de la mémoire observée en production



App Symfony - Docker Swarm 12 cpu 16go



Une prod plus rapide

Des images docker plus légères

Un pas vers des apps plus modernes (http3, real time)

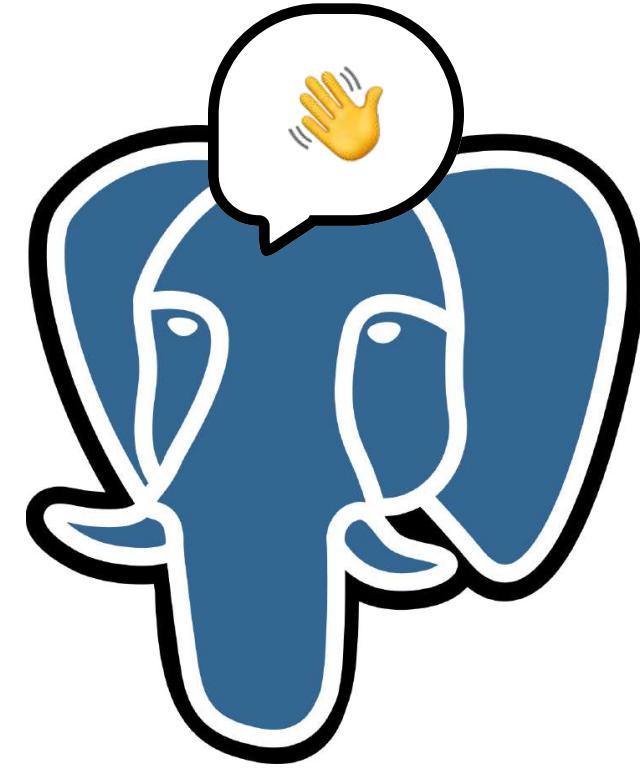
On continue...

Deuxième application



Microservice avec BDD

Base de donnée PostgreSQL avec driver Doctrine



Migration de l'app vers Franken en quelques heures

Tests OK



Éxecution en recette pendant plusieurs jours.

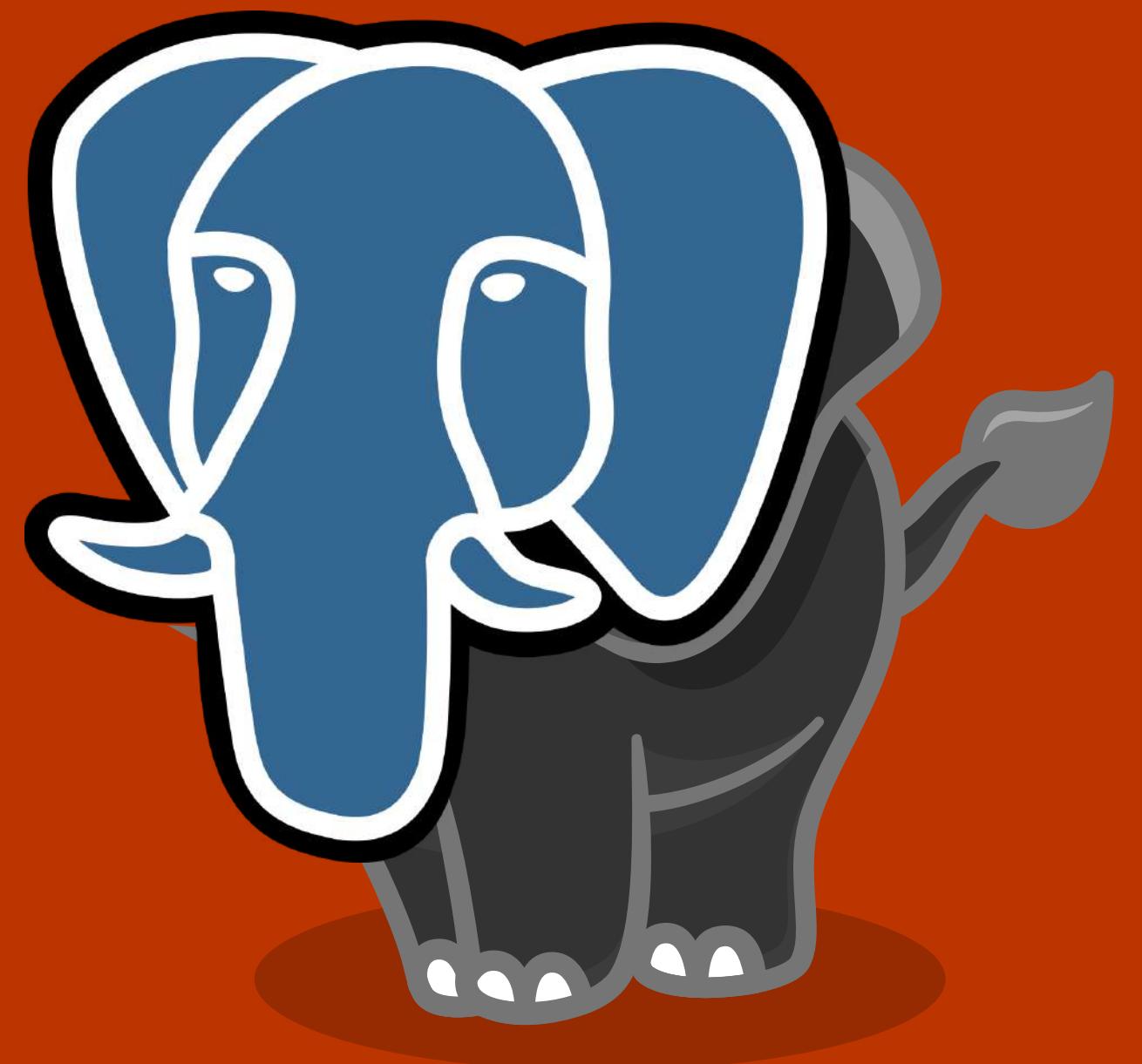
Aucune erreur remontée.

Passage en production...

Error 500

**"An exception occurred while executing a query:
SQLSTATE[HY000]: General error: 7 no connection to
the server"**

Survient aléatoirement. Disparaît temporairement après un reboot des containers.



En mode worker, Symfony garde la connexion SQL ouverte

Évite d'ouvrir/fermer la connexion entre chaque requête SQL pour gagner en performances.

```
$> select * from pg_stat_activity;
```

pid	datname	state
347918	pgpool-poisson	IDLE
289489	pgpool-poisson	IDLE
7848	pgpool-poisson	IDLE
... 24x par app franken		

⚠ En prod on utilise PGPool. Il timeout les IDLE après 10 minutes

Bug identifié dans Symfony

[DoctrineBridge] Idle connection listener for long running runtime #53214

Merged nicolas-grekas merged 1 commit into symfony:7.1 from alli83:doctrine-brige-doctrine-connection-listener-long-running-runtime on Apr 25

Conversation 143 Commits 1 Checks 8 Files changed 7 +192 -8

alli83 commented on Dec 26, 2023 • edited by nicolas-grekas

Q	A
Branch?	7.1
Bug fix?	no
New feature?	yes
Deprecations?	no
Issues	Fix #51661
License	MIT

This pull request introduces a solution based on the RoadRunner bundle's Doctrine ORM/ODM middleware <https://github.com/Baldinof/roadrunner-bundle/blob/3.x/src/Integration/Doctrine/DoctrineORMMiddleware.php#L22>. It checks the status of Doctrine connection, then if the connection is initialized and connected, it performs a 'ping' to check its viability. If the ping fails, it closes the connection.

linked to [doctrine/DoctrineBundle#1739](#)

carsonbot added Status: Needs Review, Bug, DoctrineBridge labels on Dec 26, 2023

carsonbot added this to the 6.4 milestone on Dec 26, 2023

alli83 force-pushed the doctrine-brige-doctrine-connection-listener-long-running-runtime branch 3 times, most recently from 07aa62f to ba1eefd last year

alli83 mentioned this pull request on Dec 26, 2023

Register the idle connection listener doctrine/DoctrineBundle#1739

Merged

Reviewers

- calinblaga
- alexandre-daubois
- andersonamuller
- OskarStark
- gregOire
- DavidBadura
- stof
- nicolas-grekas
- ostrolucky
- dunglas
- lyrixx
- yceruto
- welcoMatic
- kbond
- chalars
- jderusse
- xabbuh

Assignees

No one assigned

Labels

- Bug
- DoctrineBridge
- Ready
- Feature Freeze
- Status: Reviewed

Projects

None yet

“Ce n'est pas un bug, c'est une feature”

<https://github.com/symfony/symfony/pull/53214>

Patché en :

- Symfony 7.1 (mai 2024)
- Symfony 7.4 LTS (nov. 2025)

L'écosystème PHP n'est pas prêt pour le mode worker ?

Workaround

On force la fermeture de la connexion SQL
en fin de requête HTTP

**Retour en mode
1 requête = 1 connexion SQL**

Impact sur les performances : **-2ms / req.**

Solution temporaire... ou pas ?



```
<?php
// src/EventListener/OnTerminateEventCloseConnection.php

namespace App\EventListener;

use Doctrine\DBAL\Connection;
use Symfony\Component\EventDispatcher\Attribute\AsEventListener;
use Symfony\Component\HttpKernel\Event\TerminateEvent;

#[AsEventListener]
final class OnTerminateEventCloseConnection
{
    public function __construct(
        private Connection $conn
    ) {}

    public function __invoke(TerminateEvent $event): void
    {
        $this→conn→close();
    }
}
```

Conclusion

Sur base de notre usage chez Poisson Soluble

😊 Facilité de mise en place

Possibilité de revenir en arrière si besoin

😎 Adapté pour du microservice

1 seule image docker, dont mercure

🚀 Gain en performance réel

x3.5 observé

🐳 Possibilité d'utiliser Docker Alpine

Image plus légères, facilité pour configurer des crons

🥾 Un pas vers des apps plus modernes

Http3, Early hints, real time

👯 Mémoire partagée

Nécessite de revoir nos habitudes de développement et une review de code plus fine.

🧐 Écosystème PHP et mode worker

Écosystème encore peu mature pour le mode worker.
Forcé de désactivé les connexions SQL persistantes.
Doute sur certaines libs plus anciennes.

😴 Configuration spécifique difficile

Nécessité de comprendre Caddy pour désactiver le HTTPS.

🚀 **Chez Poisson Soluble, on continue de migrer nos apps vers frankenPHP**

Merci !

Crédits

<https://frankenphp.dev/>

K. DUNGLAS - PHP Tour Paris 2022 - FrankenPHP, dans les entrailles de l'interpréteur PHP, de machines virtuelles et des threads

https://www.youtube.com/watch?v=aAwBrz8zdbY&ab_channel=AFUPPHP

K. DUNGLAS - FrankenPHP - Rennes 2024

<https://rennes2024.drupalcamp.fr/programme/conferences/slides/frankenphp-drupal-sliides-visibles.pdf>

<https://stephanewouters.fr/>

