PHPkonf.

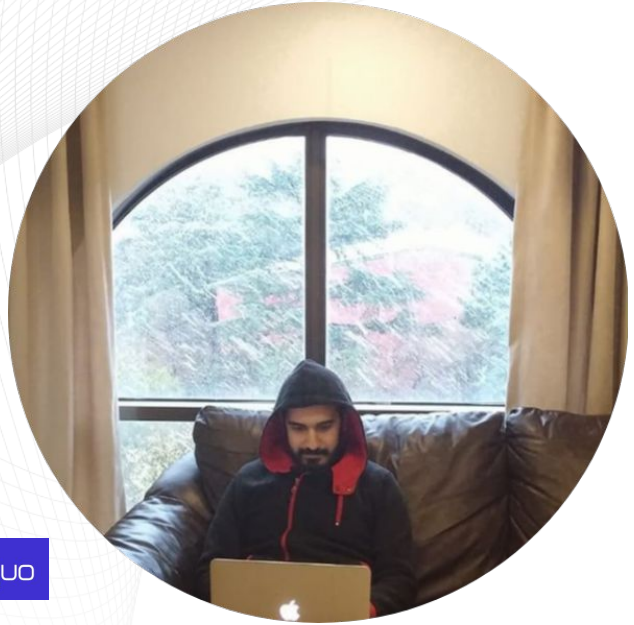
# FrankenPHP ile Alışkanlıklarımızı Değiştirelim

Mert Simsek
Software Developer, Softquo

# About

@_mertsimsek

https://mertblog.net
https://medium.com/@mertsmsk0
https://github.com/mertingen
mert.conf@gmail.com

**Kévin Dunglas**
@dunglas

The FrankenPHP website is now available in Turkish thanks to @_mertsimsek, and he will present the app server at PHPKonf 🤩

frankenphp.dev/tr/

> **Mert Simsek** @_mertsimsek · Mar 29
>
> FrankenPHP resmi doküman ve websitesi Türkçe çevirisini bitirdim. Canlı ortama alınmış. Olabildiğince teknik dili bozmamaya çalıştım. Göz atmak isteyenlere; frankenphp.dev/tr/docs/ Thanks to @dunglas
>
> Ayrıca bu hafta PHPKonf 2024 konuşma başvurumun onaylandığını öğrendim...
> Show more


9:42 PM · Mar 29, 2024 · **3,755** Views

3 33 1


Zhuojie Zhou
Jeremy Justus
Mert Simsek
Ross Sbriscia
Joshua Erney

hepsiburada

JSKonf 2024

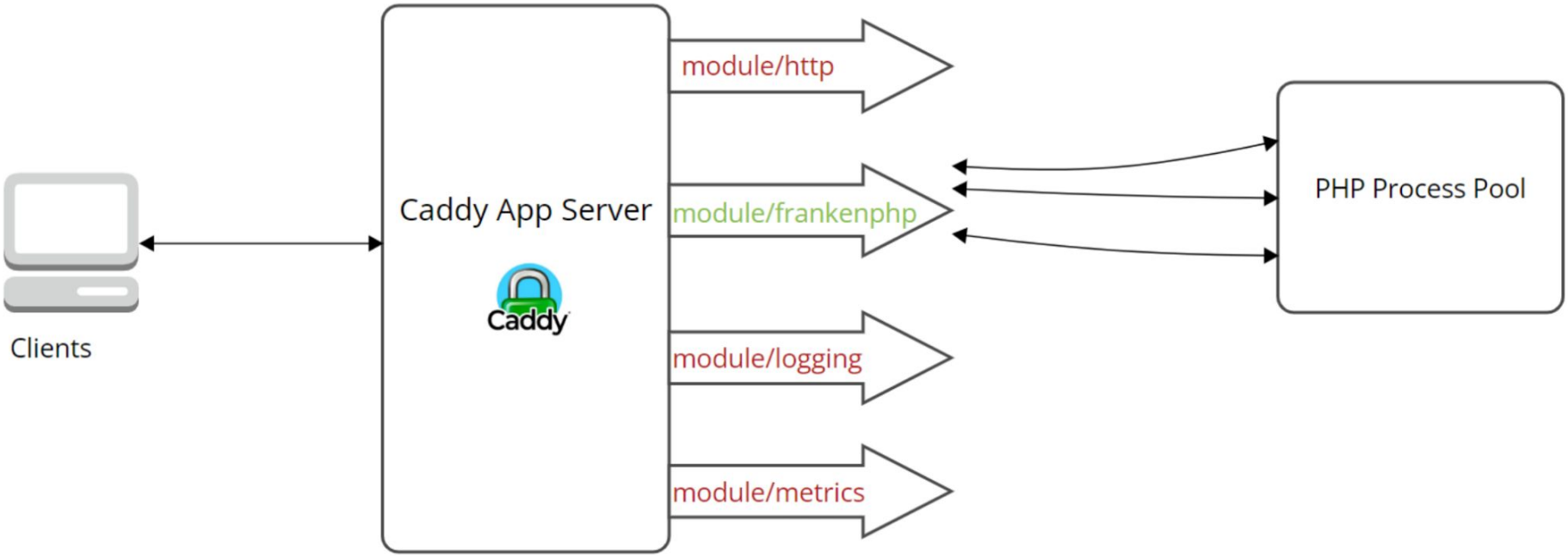
# WebAssembly ile *Back-end Servislerin Yönetimi*

**Mert Şimşek**

Software Developer, Softquo

# Modern PHP Her Zamankinden Daha İyi!

- Son teknoloji ürünü bir web sunucusuna gömülü resmi PHP yürütücüsünü kullanır: **Caddy**
- HTTP/1.1, HTTP/2 ve **HTTP/3** için yerel destek
- Otomatik **HTTPS** sertifikaları oluşturma ve yenileme (Let's Encrypt veya ZeroSSL)
- PHP dosyalarınızı belge kök dizinine kopyalayın, hepsi bu!
- PHP web uygulamalarınız ve komut satırı araçlarınız için **bağımsız**, kendi kendine çalıştırılabilir binary dosyalar oluşturun.
- **OPcache** ve **XDebug** dahil olmak üzere popüler PHP eklentileri yerel olarak desteklenmektedir!


Clients
Caddy App Server
Caddy
module/http
module/frankenphp
module/logging
module/metrics
PHP Process Pool

```go
package main

import (
    "io"
    "log"
    "net/http"
    )

func main() {
    helloHandler := func(w http.ResponseWriter, req *http.Request) {
        // PHP Uygulaması Burada...
    }

    http.HandleFunc("/hello", helloHandler)
    log.Fatal(http.ListenAndServe(":8080", nill))
}
```

The Go Blog

# C? Go? Cgo!

*Andrew Gerrand*
*17 March 2011*

## Introduction

Cgo lets Go packages call C code. Given a Go source file written with some special features, cgo outputs Go and C files that can be combined into a single Go package.

To lead with an example, here's a Go package that provides two functions - `Random` and `Seed` - that wrap C's `random` and `srandom` functions.

```
package rand

/*
#include <stdlib.h>
*/
import "C"

func Random() int {
    return int(C.random())
}

func Seed(i int) {
    C.srandom(C.uint(i))
}
```

Let's look at what's happening here, starting with the import statement.

# FrankenPHP for Octane: HTTPS

```
# Generates (and renew) a TLS certificate
# Starts an HTTPS server
# supporting HTTP/1, HTTP/2, HTTP/3
# and Early Hints
php artisan octane:start \
    --host=example.com \
    --port=443 \
    --admin-port=2019
```

## # Creating a Linux Binary

The easiest way to create a Linux binary is to use the Docker-based builder we provide.

1. Create a file named `static-build.Dockerfile` in the repository of your app:

```
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
```

> ⊗ ***CAUTION***
>
> *Some `.dockerignore` files (e.g. default Symfony Docker `.dockerignore`) will ignore the `vendor/` directory and `.env` files. Be sure to adjust or remove the `.dockerignore` file before the build.*

2. Build:

```
docker build -t static-app -f static-build.Dockerfile .
```

3. Extract the binary:

```
docker cp $(docker create --name static-app-tmp static-app):/go/src/app/dist/frankenphp-linux-x86_64 my-app ; docker rm static-app-tmp
```

The resulting binary is the file named `my-app` in the current directory.

## # Using The Binary

This is it! The `my-app` file (or `dist/frankenphp-<os>-<arch>` on other OSes) contains your self-contained app!

To start the web app run:

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

If your app contains a **worker script**, start the worker with something like:

```
./my-app php-server --worker public/index.php
```

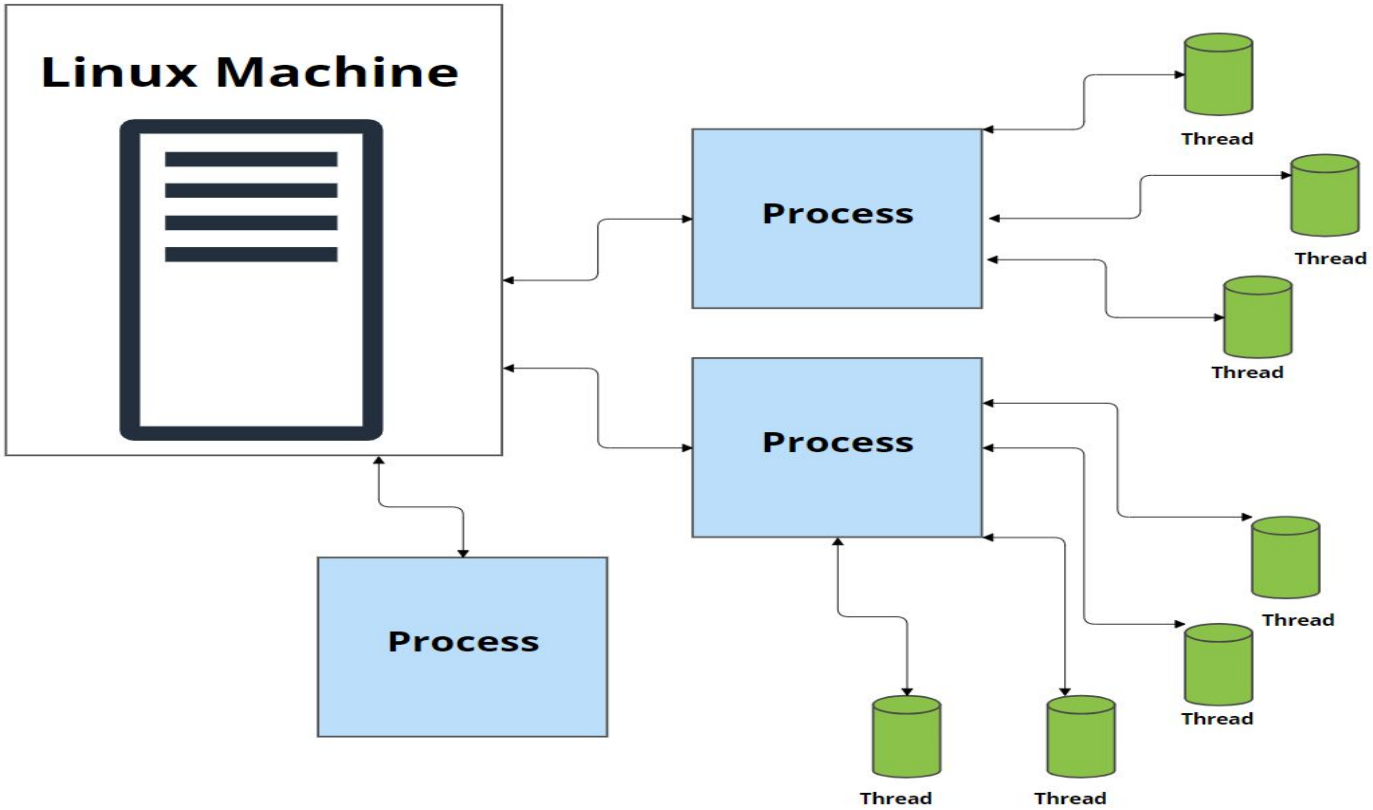
To enable HTTPS (a Let's Encrypt certificate is automatically created), HTTP/2 and HTTP/3, specify the domain name to use:

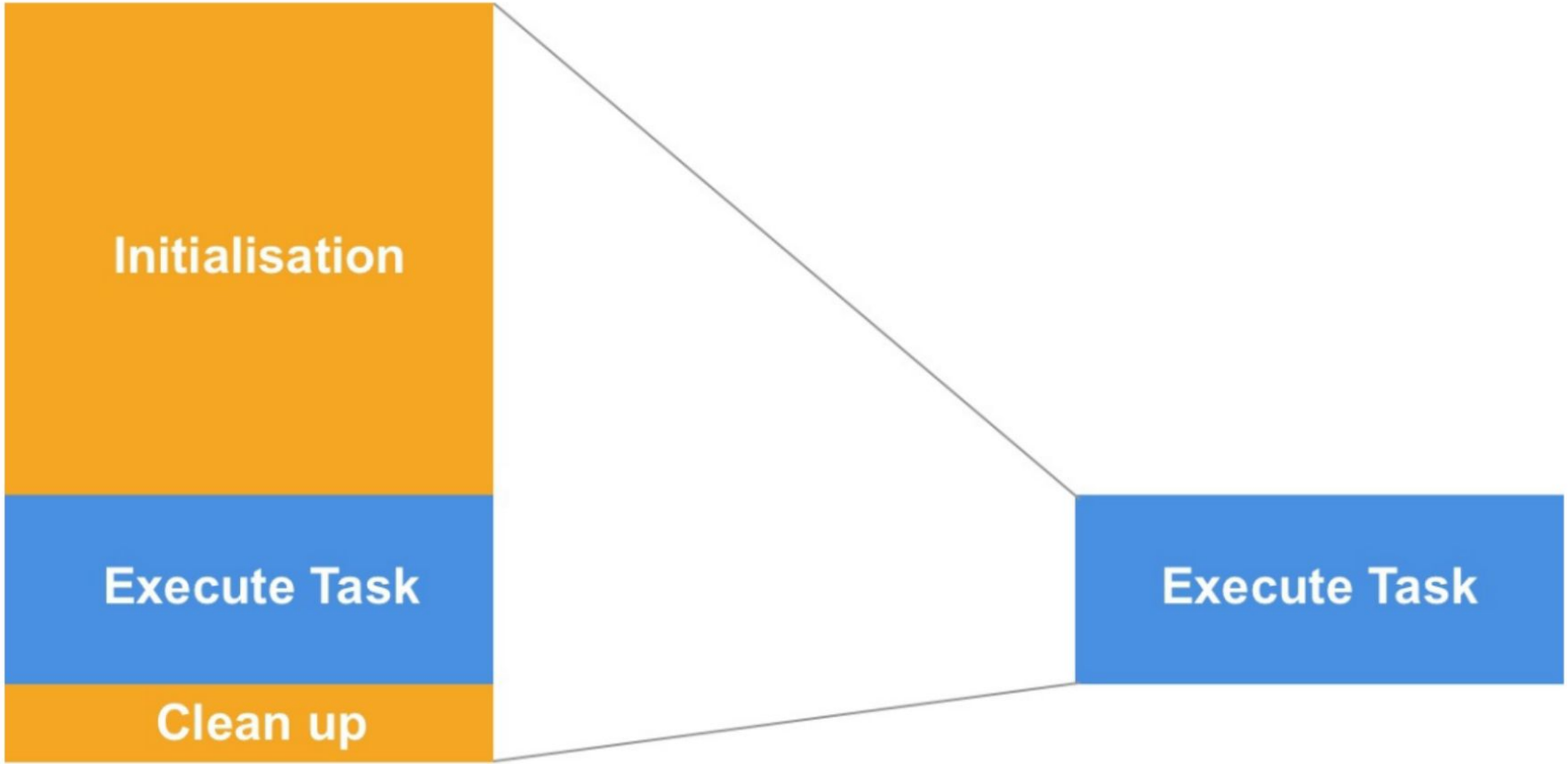
```
./my-app php-server --domain localhost
```

You can also run the PHP CLI scripts embedded in your binary:

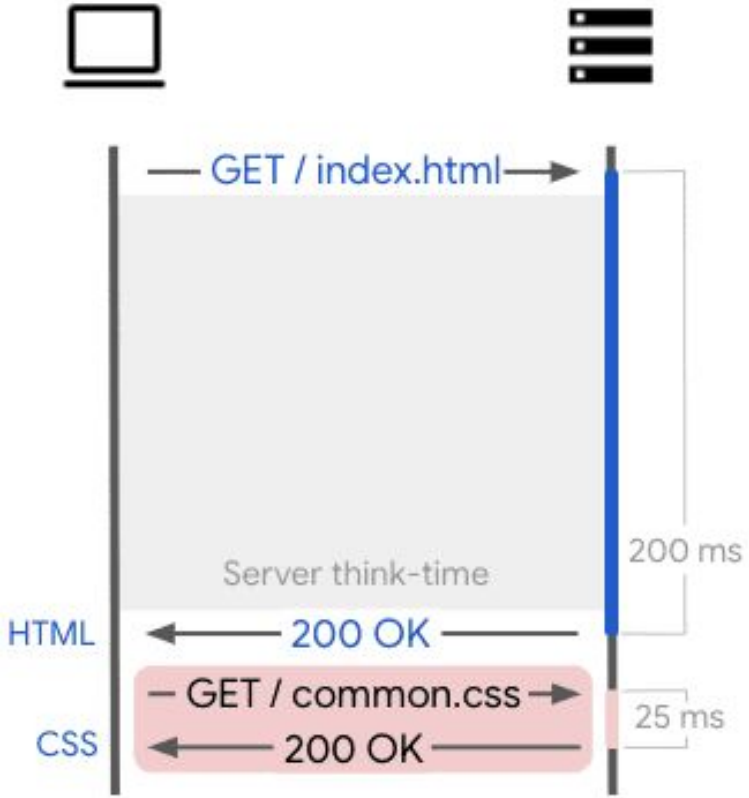
```
./my-app php-cli bin/console
```

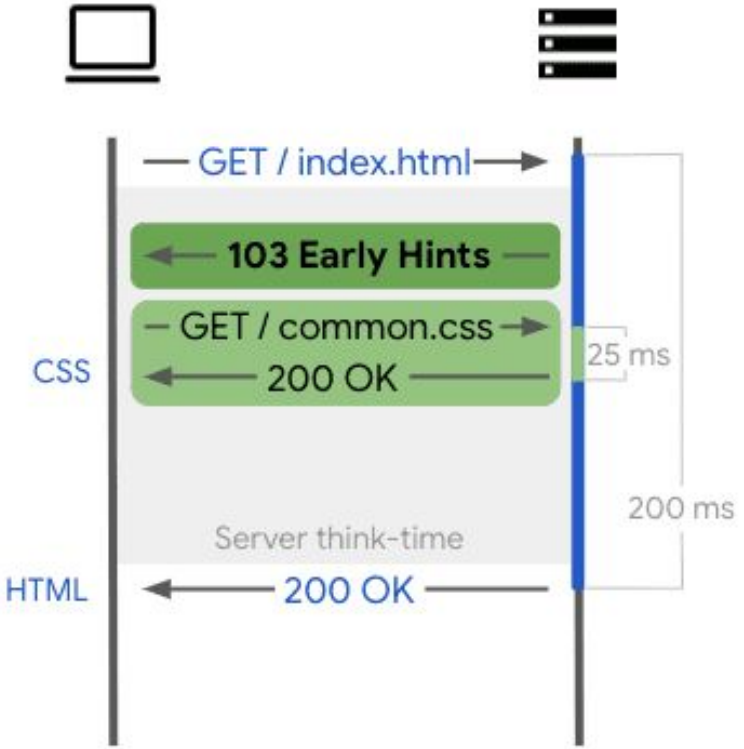
# WORKER MODE


Linux Machine
Process
Process
Process
Thread
Thread
Thread
Thread
Thread
Thread
Thread

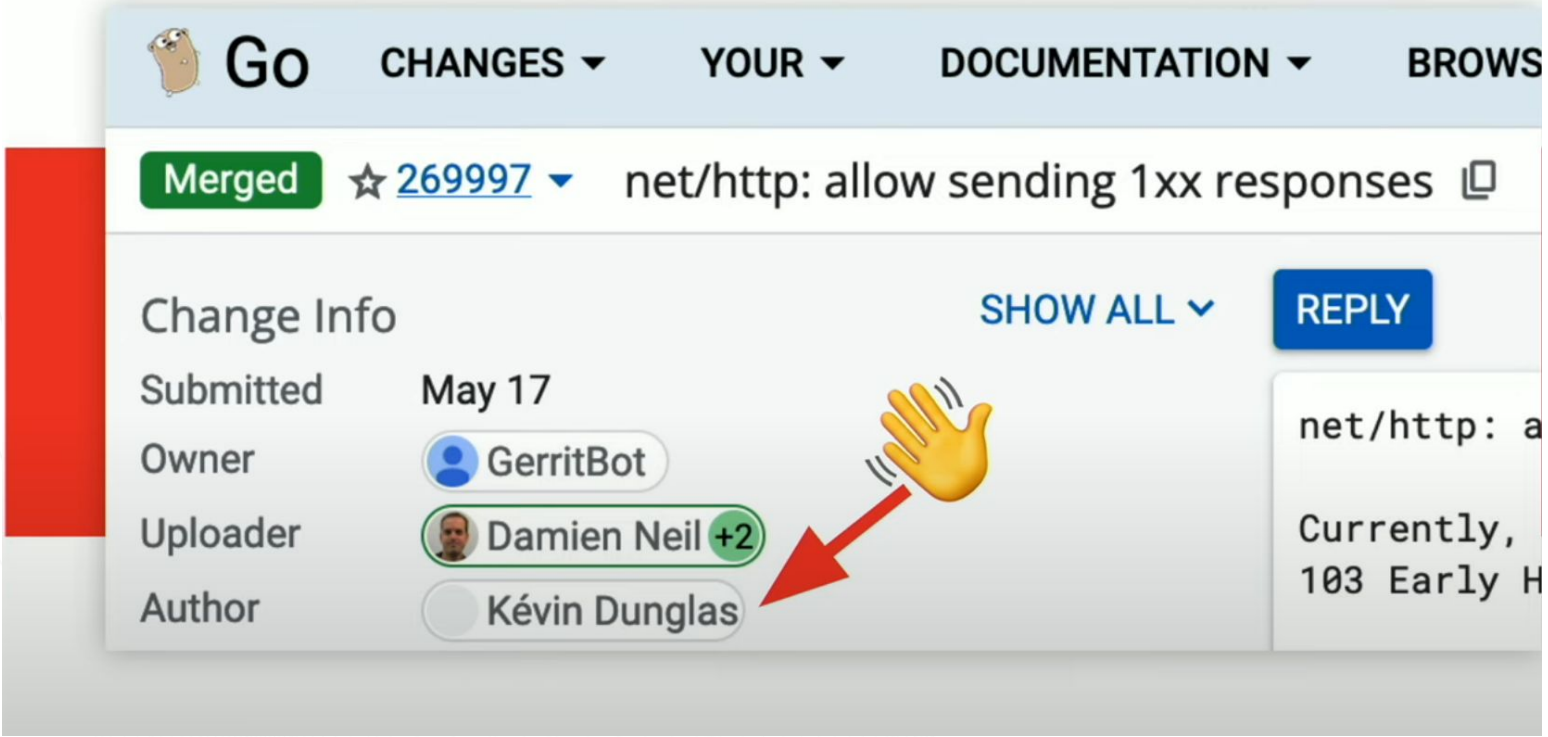

Initialisation
Execute Task
Clean up
Execute Task
PHP-FPM, PHP-CLI
FrankenPHP

# Early Hints support (103 HTTP status code)

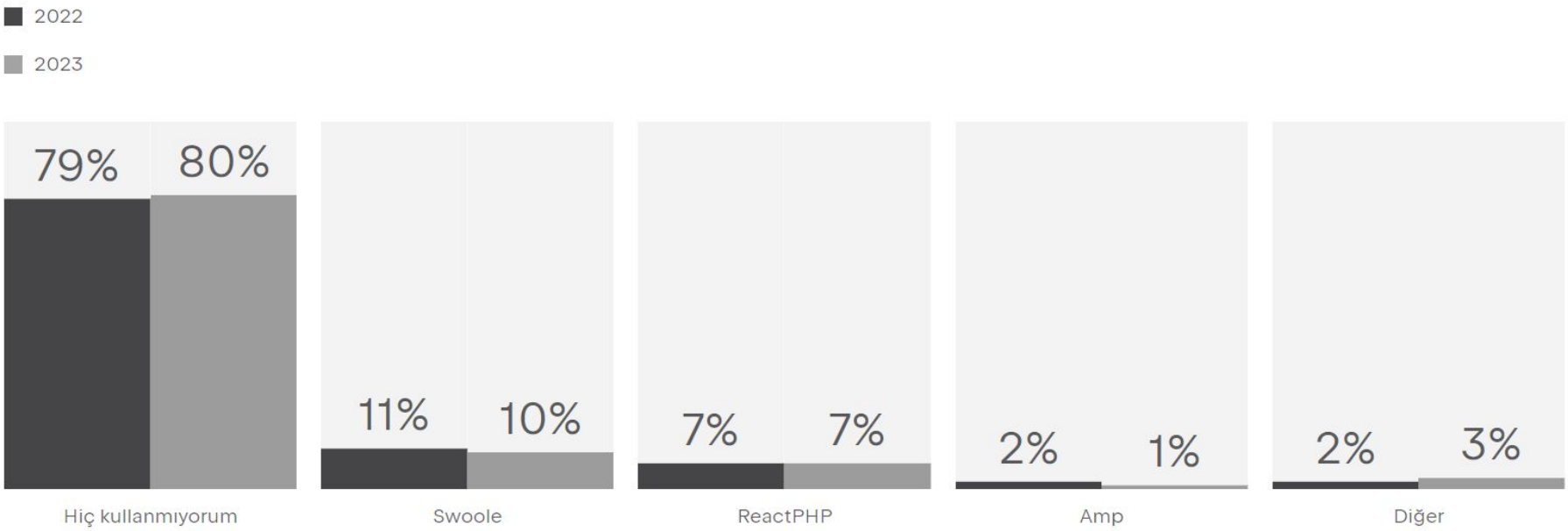


# Early Hints: Possible With Go



## Zaman uyumsuz PHP için herhangi bir kitaplık veya çerçeve kullanıyor musunuz?



# Çözmemiz gereken yeni sorunlar…

# 1-) OOM (out-of-memory, memory leaks)

## # Belirli Sayıda İstekten Sonra Worker’ı Yeniden Başlatın

PHP başlangıçta uzun süreli işlemler için tasarlanmadığından, hala bellek sızdıran birçok kütüphane ve eski kod vardır.

Bu tür kodları worker modunda kullanmak için geçici bir çözüm, belirli sayıda isteği işledikten sonra worker betiğini yeniden başlatmaktır:

“**MAX_REQUESTS**” adlı bir ortam değişkeni ayarlayarak işlenecek maksimum istek sayısını yapılandırmaya izin verir.

# 2-) MySQL Server Has Gone Away

## MySQL error 2006: mysql server has gone away

Asked 12 years, 6 months ago  Modified 28 days ago  Viewed 684k times

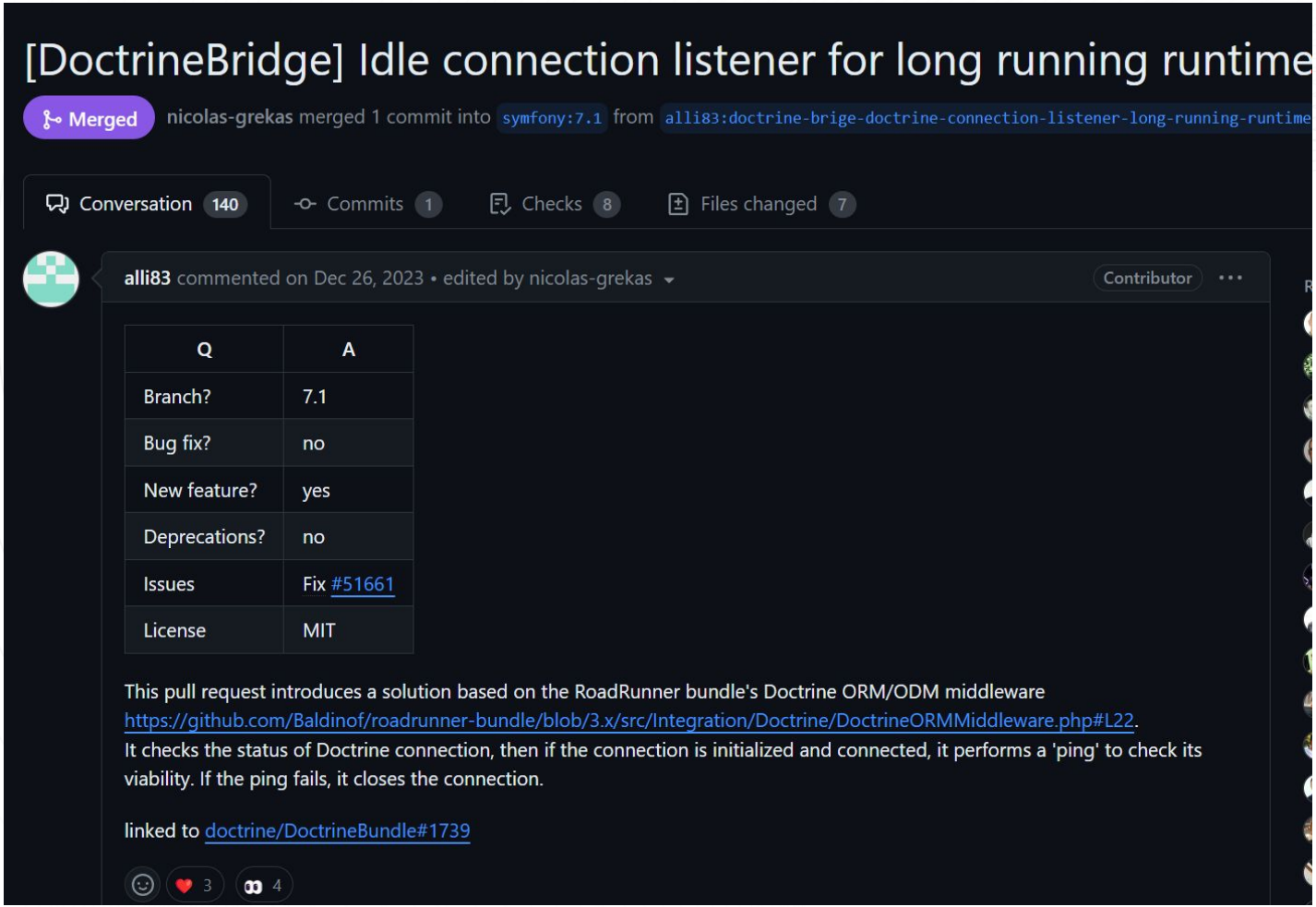
331

I'm running a server at my office to process some files and report the results to a remote MySQL server.

The files processing takes some time and the process dies halfway through with the following error:

```
2006, MySQL server has gone away
```

I've heard about the MySQL setting, **wait_timeout**, but do I need to change that on the server at my office or the remote MySQL server?


[DoctrineBridge] Idle connection listener for long running runtime
Merged
nicolas-grekas merged 1 commit into symfony:7.1 from alli83:doctrine-brige-doctrine-connection-listener-long-running-runtime
Conversation 140
Commits 1
Checks 8
Files changed 7
alli83 commented on Dec 26, 2023 • edited by nicolas-grekas
Contributor
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
3
4

```php
try {
    $this->entityManager->getConnection()->executeQuery( sql: "SELECT 1")->fetchOne();
} catch (\Exception $e) {
    if (str_contains($e->getMessage(), 'gone away')) {
        $this->entityManager->getConnection()->close();
        $this->entityManager->getConnection()->connect();
    }
}
```

# 3-) Redis Idle Timeout (read error on connection)

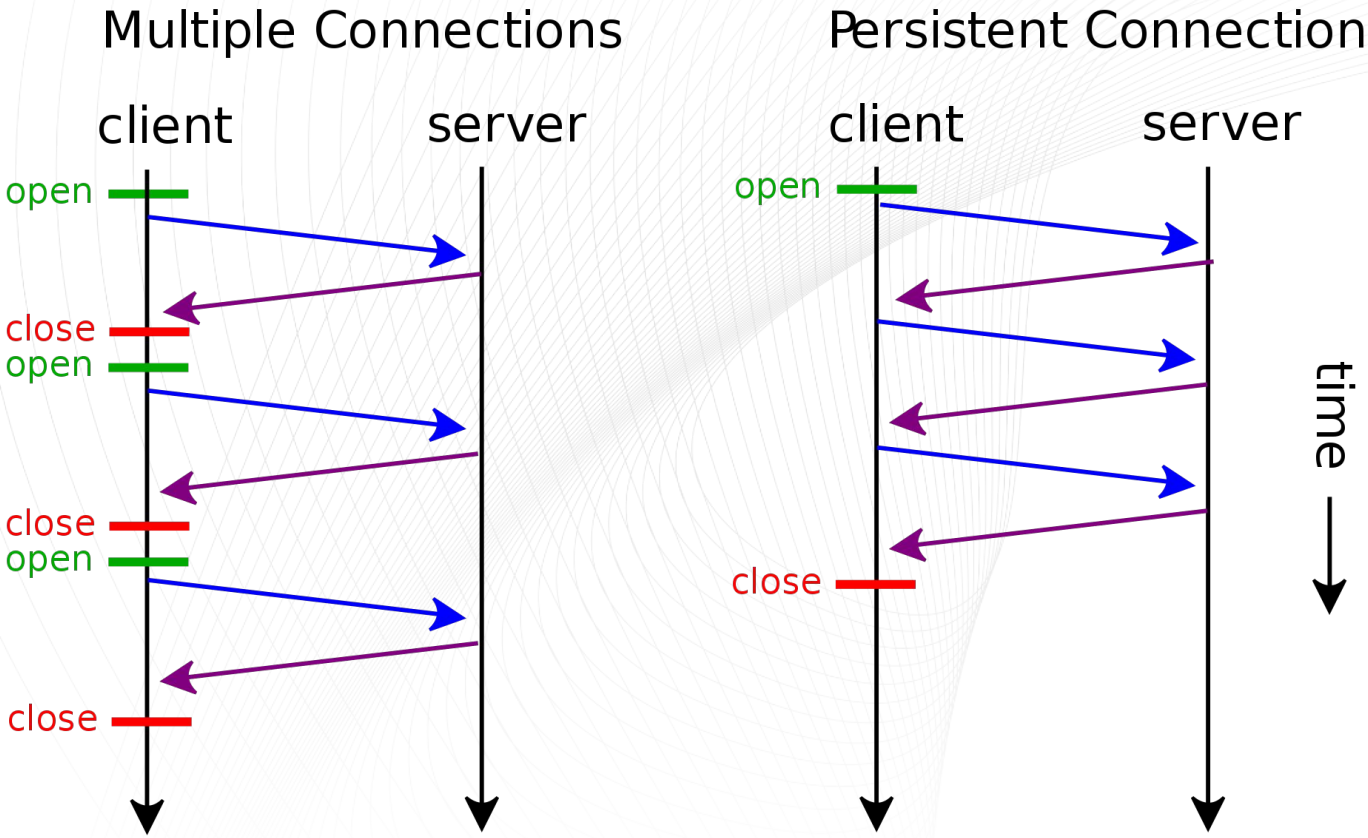
- “default_socket_timeout” (önerilmeyen)
  - ini_set('default_socket_timeout', -1);

- “default_socket_timeout” (önerilmeyen)
  - $redis->setOption(Redis::OPT_READ_TIMEOUT, -1);

- Kopan bağlantı için tekrar bağlan

```php
$rds = new Redis();
try {
    $redisConn = $rds->pconnect( host: "127.0.0.1",  port: 6390);
    if ($redisConn == false) {
        echo "Redis client couldn't be initialized.";
        exit;
    }
} catch (Exception $e) {
    $redisConn = $rds->pconnect( host: "127.0.0.1",  port: 6390);
}
```

# 4-) HTTP Keep-Alive (TCP time wait, TCP close wait)



### Persisting Singletons

Only singletons that are resolved during the application bootstrapping will persist between requests. Singletons that are resolved during request handling will be registered in the sandbox container, this container is destroyed after handling the request.

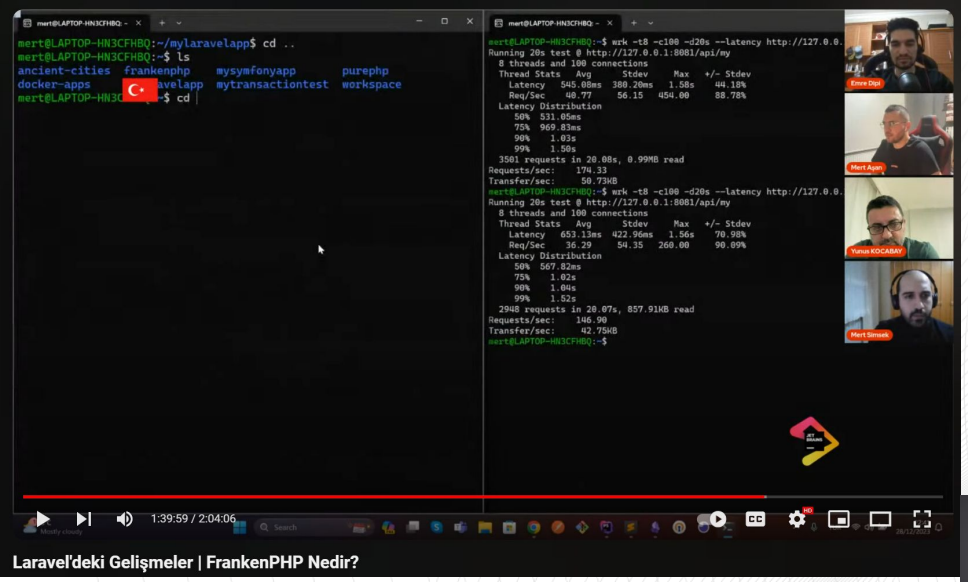
To persist singletons between requests, you can either resolve them in your service providers or add them to the `warm` array inside the Octane configuration file:

```
'warm' => [
    ...Octane::defaultServicesToWarm(),
    Service::class
],
```

https://divinglaravel.com/laravel-octane-bootstrapping-the-application-and-handling-requests

- Bir önceki slaytta olduğu gibi sürekli HTTP isteği yaptığımız aynı “IP:PORT” değerlerindeki TCP bağlantılarını korumalıyız.

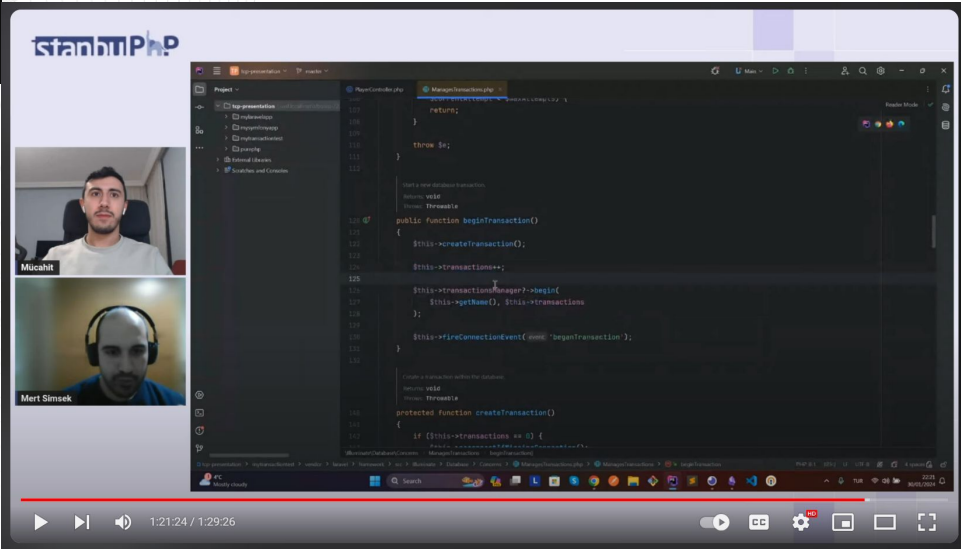
- Çok sık HTTP isteği yapmadığımız e-mail, log, rapor servisleri veya herhangi bir hedef için de bu TCP bağlantılarını başarılı bir şekilde sonlandırmalıyız.




Laravel'deki Gelişmeler | FrankenPHP Nedir?

# Laravel'deki Gelişmeler | FrankenPHP Nedir?

# İstanbulPHP - TCP Bağlantı Pratikleri




İstanbul PHP - TCP Bağlantı Pratikleri 2, Mert Şimşek

PHPkonf.

# FrankenPHP ile Alışkanlıklarımızı Değiştirelim

Mert Simsek
Software Developer, Softquo