



Next-Gen PHP

Aula 02

Crie ambientes escaláveis



THE TWELVE-FACTOR APP

I. Base de Código

Um base code com rastreamento utilizando controle de revisão, muitos deploys

II. Dependências

Declare e isole as dependências

III. Configurações

Armazene as configurações no ambiente

IV. Serviços de Apoio

Trate os serviços de apoio, como recursos ligados

V. Construa, lance, execute

Separe estritamente os builds e execute em estágios

VI. Processos

Execute a aplicação como um ou mais processos que não armazenam estado





THE TWELVE-FACTOR APP

VII. Vínculo de porta

Exporte serviços por ligação de porta

VIII. Concorrência

Dimensione por um modelo de processo

IX. Descartabilidade

Maximizar a robustez com inicialização e desligamento rápido

X. Dev/prod semelhantes

Mantenha o desenvolvimento, teste, produção o mais semelhante possível

XI. Logs

Trate logs como fluxo de eventos

XII. Processos de Admin

Executar tarefas de administração/gerenciamento como processos pontuais



As dores

1. Qual a versão do PHP?
2. Quais extensões?
3. Quais serviços?
4. Como instalar?
5. Está tudo rodando?
6. Como reproduzir
7. Minha entrega está de acordo?
8. Scripts padronizados?
9. E o banco? cadê o dump?



Defina a versão de TUDO

```
"require": {  
    "php": "^8.2",  
    "slim/slim": "^4.12", 4.12.0  
    "slim/psr7": "^1.6", 1.6.1  
    "doctrine/dbal": "^3.6", 3.7.0  
    "mercadopago/dx-php": "2.5.5", 2.5.5  
    "robmorgan/phinx": "^0.15", 0.15.2  
    "peroxide/container": "^1.0", v1.0.4  
    "respect/validation": "^2.2", 2.2.4  
    "mailjet/mailjet-apiv3-php": "^1.6", v1.6.0  
    "ramsey/uuid": "^4.7", 4.7.4
```

```
services:  
    web:  
        image: webdevops/php-nginx-dev:8.2-alpine  
        working_dir: /app  
  
    mysql:  
        image: mysql:8.0
```

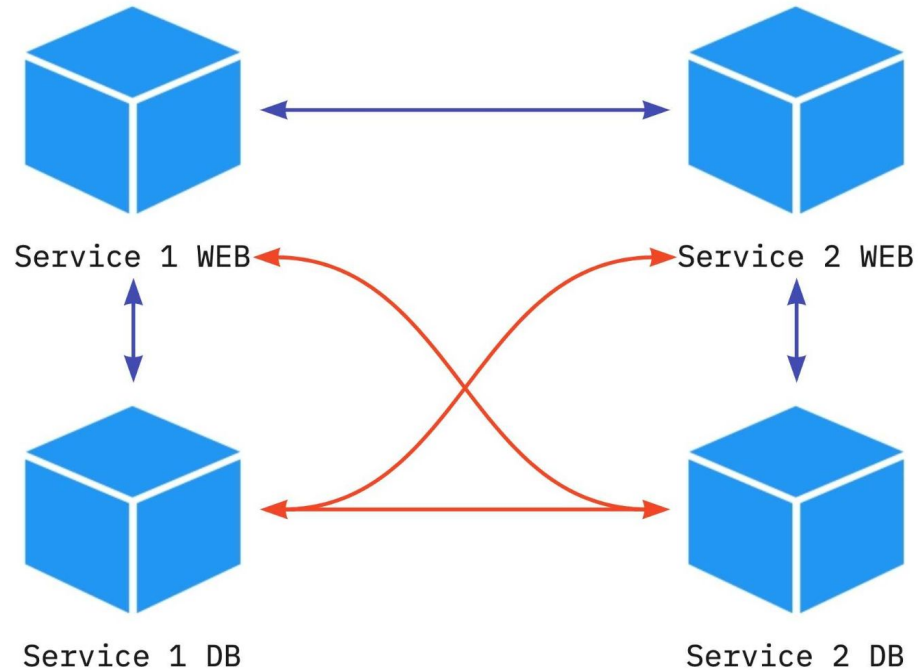


Faça o **composer** validar para você

```
{
    "require": {
        "php": "^7.4 || ^8.0",
        "ext-curl": "*",           // Qualquer versão da extensão cURL
        "ext-json": "^1.0"       // Versão específica da extensão JSON
    },
}
```



Entenda o **network** do Docker Compose



Docker garante ambientes configuráveis independentes

1. Criar sua própria imagem
2. Armazenar build da imagem
3. Instalar extensões do PHP
4. Compilar extensão
5. Configurar o Docker Compose
6. Usando envs do Docker
7. Executar scripts no container
8. Shell ou Makefile como ferramenta
9. Setup scripting



Testes rápidos

Para rodar um container PHP rápido:

```
$ docker run --rm [nome_container:tag] [comandos dentro do container]
```

Exemplo: executando um container PHP mostrando a versão

```
$ docker run --rm php:8.3-alpine php -v
```



Hands-on

Vamos montar um ambiente escalável e portátil

<https://hub.docker.com/>

```
$ docker login
```

```
$ docker tag [nome_imagem_local] [imagem_no_dockerhub]
```

```
$ docker push [imagem_no_dockerhub]
```



Desafio

Presente no repositório do nosso curso:

<https://github.com/DifferDev/NextGenPHP>



Boa noite!

Obrigado pela presença!

