# slim-in-docker-hello-world

Cél: Docker konténeres környezetben szeretnék készíteni egy Slim framework projektet. Mindezt szeretném eltárolni egy GitHub repozitoriban.

1. **Docker konténeres környezet:** Hordozható és könnyen konfigurálható fejlesztési környezetet biztosít. A Docker egy platform, amely lehetővé teszi, hogy alkalmazásokat és azok összefüggő komponenseit konténerekben futtasd. A konténerek izolált, hordozható környezetet biztosítanak, amelyben könnyen kezelhetőek a különböző alkalmazások, például a Slim framework.
    - **Letöltés és telepítés:** https://www.docker.com/products/docker-desktop
    - **Dokumentáció és útmutatók:** https://docs.docker.com/
2. **Slim Framework projekt:** Ideális választás, ha gyorsan szeretnél modern webes vagy API-alapú alkalmazást fejleszteni. A Slim egy könnyű, PHP alapú mikrokeretrendszer, amely gyors és hatékony webalkalmazások fejlesztésére alkalmas. Ideális API-k és egyszerű weboldalak készítésére.
    - **Hivatalos oldal:** https://www.slimframework.com/
    - **Dokumentáció:** https://www.slimframework.com/docs/v4
3. **GitHub repozitoriban tárolás:** A GitHub egy felhőalapú verziókezelő platform, amely lehetővé teszi a kódjaid tárolását, verziózását és másokkal való megosztását. A repozitori (repo) az a hely, ahol a projekt összes fájlja és annak előzményei tárolódnak.
    - **GitHub regisztráció és hozzáférés:** https://github.com/
    - **GitHub Desktop:** https://desktop.github.com/

Szint: Abszolút kezdő

## 1. GitHub repozitori elkészítése, klónozása a GitHub Desktoppal és a projekt megnyitása Visual Studio Code-ban

Ez a lépés bemutatja, hogyan hozhatsz létre egy GitHub repót, klónozhatod azt a gépedre a GitHub Desktop segítségével, és nyithatod meg Visual Studio Code-ban a fejlesztéshez.

### 1.1. Készíts a GitHub-on egy publikus repót

1. Lépj be a GitHub oldalára, vagy regisztrálj egy fiókot, ha még nincs.
2. A jobb felső sarokban kattints a **"+" ikonra**, majd válaszd a **New repository** lehetőséget.
3. Töltsd ki a következő mezőket:
    - **Repository name**: Add meg a repó nevét, például slim-in-docker-hello-world.
    - **Public**: Válaszd a nyilvános opciót (ha mások számára elérhetővé akarod tenni).
4. Kattints a **Create repository** gombra.

### 1.2. Klónozd le a GitHub Desktop-pal

1. Töltsd le és telepítsd a **GitHub Desktop** alkalmazást: https://desktop.github.com/
2. Indítsd el az alkalmazást, és jelentkezz be a GitHub fiókoddal.

3. A GitHub Desktop-ban válaszd a **File > Clone repository** menüt, vagy kattints a **Clone a repository** opcióra.
4. A listából válaszd ki az előzőleg létrehozott repót, vagy add meg annak URL-jét (például https://github.com/felhasznalonev/slim-in-docker-hello-world.git).
5. Válaszd ki a helyi mappát, ahova klónozni szeretnéd a repót, majd kattints a **Clone** gombra.

**1.3. Nyisd meg a VSC-vel a mappát**

1. Töltsd le és telepítsd a **Visual Studio Code**-ot: https://code.visualstudio.com/
2. Indítsd el a Visual Studio Code-ot.
3. Kattints a **File > Open Folder** menüpontra, és válaszd ki azt a mappát, ahova a repót klónoztad (például: ~/Documents/slim-in-docker-hello-world).
4. A projekted most megnyílik, és készen állsz a fejlesztésre.

**Visual Studio Code útmutató**: https://code.visualstudio.com/docs

## 2. A konténer előkészítése a docker-compose.yml és a Dockerfile segítségével

### 2.1. Hozd létre: docker-compose.yml a projekt gyökerében, és legyen ez a tartalma:

```
services:
  app:
    image: php:8.2-apache
    container_name: slim-app
    ports:
      - "8080:80"
    volumes:
      - ./app:/var/www/html
    working_dir: /var/www/html
    build:
      context: .
      dockerfile: Dockerfile
```

Magyarázat:

- **services**: Egy szolgáltatáscsoportot definiál. Itt az app nevű szolgáltatás a PHP alkalmazásunk.
- **image: php:8.2-apache**: A konténer a PHP 8.2-es verzióját és az Apache webszervert használja.
- **container_name: slim-app**: Meghatározza a konténer nevét.
- **ports: "8080:80"**: A 8080-as portot a gépen összeköti a konténer 80-as portjával, ahol az Apache fut.
- **volumes: ./app:/var/www/html**: A helyi ./app mappa tartalmát hozzákapcsolja a konténer /var/www/htmlmappájához.
- **working_dir: /var/www/html**: Beállítja az alapértelmezett munkakönyvtárat a konténeren belül.
- **build**: Megadja, hogy a konténer képét helyben, a megadott Dockerfile alapján építse fel.

### 2.2. Hozd létre: Dockerfile a projekt gyökerében, és legyen ez a tartalma:

```
FROM php:8.2-apache
```

```
RUN apt-get update && apt-get install -y \
    unzip \
    git \
    libzip-dev && \
    docker-php-ext-install zip

COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
```

Magyarázat:

- **FROM php:8.2-apache**: Az alap image, amely tartalmazza a PHP 8.2-t és az Apache webszervert.
- **RUN apt-get update && apt-get install -y ...**: Telepíti a szükséges csomagokat:
  - unzip: Fájlok kicsomagolásához.
  - git: Verziókezeléshez (például a Composer függőségeihez).
  - libzip-dev: A ZIP támogatáshoz szükséges.
  - docker-php-ext-install zip: Aktiválja a PHP ZIP bővítményt.
- **COPY --from=composer:latest /usr/bin/composer /usr/bin/composer**: A legfrissebb Composer verziót egy másik Docker image-ből másolja át.

## 3. A konténer indítása és a Slim keretrendszer telepítése

A következő lépések segítenek a Slim Skeleton keretrendszer telepítésében. A Visual Studio Code-ban nyiss egy terminált és add ki az alábbi parancsokat:

### 3.1. Konténer indítása:

- docker-compose build: A Dockerfile alapján felépíti az image-et.
- docker-compose up -d: Elindítja a konténert háttérben.

### 3.2. Slim Skeleton telepítése:

- docker exec -it slim-app bash: Belépsz a konténerbe.
- composer create-project slim/slim-skeleton .: A Composer letölti és inicializálja a Slim Skeleton projektet az aktuális könyvtárban.

## 3.3. Apache mod_rewrite bekapcsolása

A .htaccess fájlok szabályainak érvényesítéséhez szükséges:

- **a2enmod rewrite**: Engedélyezi az Apache mod_rewrite modulját.
- **service apache2 restart:** Újraindítja az Apache-ot, hogy az engedélyezett modul aktiválódjon. A service apache2 restart után: „Restarting Apache httpd web server: apache2Terminated” üzenettel kilép a konténerből, ezért elindítom a docker-compose start

Ez helyettesíthető azzal, hogy az a2enmod rewrite parancsot a Dockerfile-ba írod, így automatikusan aktiválódik a konténer építésekor:

```
# Apache konfiguráció: rewrite modul engedélyezése
RUN a2enmod rewrite
```

## 4. Kész. Hello world!

Ezután dolgozhatsz a Slim-mel a kontérben, a http://localhost:8080 útvonalon a böngésző megjeleníti a „Hello world!” szöveget.

Ha végeztél a programozással, a konténer leállítása a docker-compose stop paranccsal történik.

# 5. A docker-compose parancsokról

## 5.1. docker-compose down/up = Újratelepítés

Ez valójában olyan, mint amikor egy natív környezetet újratelepítesz.

- **down**:
  - Minden ideiglenes erőforrást (konténerek, hálózatok) eltávolít.
  - A volumeneket (adatokat) csak akkor törli, ha kifejezetten kéred (--volumes kapcsolóval).
  - Az újraindításkor (up) teljesen új konténerek jönnek létre a docker-compose.yml alapján.
  - Ha például hibás környezetet akarsz „újratelepíteni”, vagy teljes tisztítást szeretnél, ez a megfelelő módszer.
- **up**:
  - Új konténereket hoz létre, mintha most kezdenéd a projektet.
  - Telepíti a szükséges függőségeket (pl. Composer vagy NPM).
  - Frissen és tisztán indul el minden.

## 5.2. docker-compose stop/start = Hibernálás

Ez úgy működik, mint a számítógép hibernálása vagy alvó módja.

- **stop**:
  - Leállítja a konténereket, de megőrzi azok aktuális állapotát (például memóriában lévő adatokat vagy futó folyamatokat már nem, de a lemezen lévő adatokat igen).
  - A hálózat és a konténerek meglévő beállításai is megmaradnak.
  - Ha újraindítod (start), ott folytatódik, ahol abbahagytad, anélkül hogy újraépítené a környezetet.
- **start**:
  - Egyszerűen újraindítja a leállított konténereket az aktuális állapotukkal.
  - Gyorsabb, mert nem kell újraépítenie az image-eket és a környezetet.

## 5.3. Mikor melyiket használd?

| **Cél** | **Használandó parancs** |
|---|---|
| Gyorsan folytatni a fejlesztést, ahol abbahagytad | docker-compose stop → docker-compose start |
| Megszüntetni egy hibás vagy elavult környezetet | docker-compose down → docker-compose up |
| Tisztán újraépíteni a környezetet (pl. új függőségek miatt) | docker-compose down → docker-compose up --build |
| Csak ideiglenesen leállítani a konténereket | docker-compose stop |

### 5.4. Fontos tudni

1. **Volumenek (adattárolás)**:
    - Ha a konténer használ volume-okat (pl. adatbázisok), azok megmaradnak még down után is, kivéve ha explicit törlöd őket a –volumes kapcsolóval.
    - Ha újratelepítést akarsz **minden adat nélkül**, használd:
    - docker-compose down --volumes
2. **Újraépítés (--build)**:
    - Ha változtatsz valamin a Dockerfile-ban, a környezet újraépítéséhez mindig használd az –build kapcsolót az up parancsnál:
    - docker-compose up --build
3. **Adatok és állapotok**:
    - stop/start megőrzi a konténerek belső állapotát és konfigurációját.
    - down/up mindent alaphelyzetbe állít (kivéve a volume-ok adatait, ha nem törlöd azokat).

### 5.5. Röviden

- **stop/start = hibernálás**: gyors, mert semmi nem épül újra.
- **down/up = újratelepítés**: tiszta környezet, de minden konténer és hálózat újra létrejön.

## 6. Összefoglaló

Ezzel a dokumentációval egy teljes fejlesztési környezetet állíthatsz fel, és hatékonyan használhatod a Docker és Slim Framework nyújtotta előnyöket! Happy Coding!