Докер для розробки Django

1. Встановлення та перші кроки з Docker

Перш ніж почати, вам потрібно встановити Docker на вашу операційну систему. Для більшості платформ рекомендовано використовувати **Docker Desktop**, який включає Docker Engine, CLI, Docker Compose та інші інструменти в одному пакеті.

Встановлення на Windows

1. Перейдіть на офіційний сайт Docker Desktop і завантажте інсталятор для Windows.
2. Запустіть його та слідуйте інструкціям на екрані. Переконайтесь, що опція **"Use WSL 2 instead of Hyper-V"** активована, оскільки це забезпечить найкращу продуктивність.
3. Після завершення інсталяції, перезавантажте комп'ютер.

Використання WSL (Windows Subsystem for Linux)

Docker Desktop для Windows використовує **Windows Subsystem for Linux 2 (WSL 2)** як основу для запуску Docker Engine. Це забезпечує повноцінну сумісність з Linux та значно вищу продуктивність порівняно з Hyper-V.

Якщо у вас ще не встановлено WSL 2, виконайте наступні кроки у PowerShell від імені адміністратора:

1. Встановіть WSL та дистрибутив Linux за замовчуванням (наприклад, Ubuntu):
   wsl --install

2. Переконайтесь, що WSL 2 встановлено як версія за замовчуванням:
   wsl --set-default-version 2

3. Перевірте статус встановлених дистрибутивів:
   wsl --list --verbose

Після успішного встановлення WSL 2, Docker Desktop автоматично налаштує інтеграцію і дозволить вам працювати з контейнерами.

Встановлення на macOS

1. Перейдіть на офіційний сайт Docker Desktop і завантажте інсталятор для macOS.
2. Відкрийте завантажений файл .dmg і перетягніть іконку Docker до папки "Applications".
3. Запустіть Docker Desktop.

Встановлення на Linux

На Linux ви можете встановити Docker через термінал, оскільки він не має графічного інсталятора.

1. Оновіть системні пакети: sudo apt-get update.
2. Встановіть Docker Engine: sudo apt-get install docker-ce docker-ce-cli containerd.io.
3. Щоб запускати команди Docker без sudo, додайте свого користувача до групи docker: sudo usermod -aG docker $USER.
4. Вийдіть і знову увійдіть в систему, щоб зміни вступили в силу.

Перевірка встановлення

Після інсталяції відкрийте термінал і виконайте наступні команди, щоб переконатися, що все працює правильно:

- Перевірка версії Docker:
  docker --version

- Перевірка версії Docker Compose:
  docker-compose --version

- Запуск першого тестового контейнера hello-world:
  docker run hello-world

  Це завантажить невеликий образ і запустить його, підтверджуючи успішне встановлення.

2. Основні поняття Docker

Docker — це платформа для контейнеризації додатків. Вона дозволяє "упакувати" додаток з усіма його залежностями в єдиний, універсальний пакет, що називається **образом**. Коли ви запускаєте образ, він перетворюється на **контейнер**, який працює повністю ізольовано.

- **Контейнер (Container):** Ізольоване середовище, що створюється з образу. Це живий, запущений процес, що містить ваш додаток і все необхідне для його роботи. Контейнери ізольовані один від одного, що запобігає конфліктам між залежностями.
- **Образ (Image):** Статичний, незмінний шаблон, з якого створюються контейнери. Його можна порівняти з інструкцією для збірки LEGO. Образ містить код, системні бібліотеки, середовище виконання та інші інструменти.
- **Dockerfile:** Текстовий файл, що містить інструкції для створення образу. Це ваш "рецепт", де ви покроково описуєте, що має бути в образі.

Ось приклад простого Dockerfile:

```
# Використання офіційного образу Python 3.9 як базового
FROM python:3.9-slim

# Встановлення робочого каталогу всередині контейнера
WORKDIR /app

# Копіювання файлу залежностей до робочого каталогу
COPY requirements.txt .

# Встановлення залежностей, вказаних у requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Копіювання всього коду вашого додатку до контейнера
COPY . .

# Визначення команди для запуску додатку
CMD ["python", "app.py"]
```

3. Docker Compose: Оркестрація багатоконтейнерних додатків

Зазвичай додатки не працюють поодинці. Проект Django потребує бази даних (Postgres) і, можливо, кешування (Redis). **Docker Compose** — це інструмент, який дозволяє вам керувати всіма цими контейнерами як єдиним цілим.

Він використовує один файл docker-compose.yml, де ви визначаєте всі сервіси, їхні залежності та конфігурації.

Основні функції Docker Compose:

- **Конфігурація в одному YAML файлі:** Усі сервіси, мережі та томи описані в одному файлі, що спрощує їх налаштування.
- **Запуск і зупинка всього стеку:** Команди docker-compose up та docker-compose down дозволяють запускати та зупиняти всі сервіси одночасно.
- **Оркестрація:** Compose гарантує, що контейнери запускаються в правильному порядку. Наприклад, база даних запуститься раніше, ніж ваш веб-сервіс, який від неї залежить.

Приклад файлу docker-compose.yml:

```yaml
version: '3.8'
services:
  web:
    build: .  # Зібрати образ з поточного Dockerfile
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code # Монтування поточного каталогу в контейнер
    ports:
      - "8000:8000"
    depends_on:
      - db
  db:
    image: postgres:13
    volumes:
      - db_data:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=django_db
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=password
volumes:
  db_data: # Визначення тому для зберігання даних
```

- services: Два сервіси: web (Django) і db (Postgres).
- volumes: Використовуються для збереження даних. db_data зберігає дані бази даних, а . :/code дозволяє "гаряче" перезавантажувати код під час розробки.
- depends_on: Вказує, що сервіс web залежить від db, тому db запуститься першим.

4. Docker Machine: Управління Docker-хостами

Docker Machine — це інструмент для створення та керування Docker-хостами, тобто віртуальними машинами, на яких працює Docker Engine. Він спрощує розгортання вашого Docker-середовища на віддалених серверах або у віртуальних машинах.

Основні функції Docker Machine:

- **Створення хостів:** Автоматично створює віртуальні машини з Docker на різних платформах (наприклад, VirtualBox, DigitalOcean, AWS).
- **Керування:** Дозволяє керувати конфігурацією та станом віддалених хостів.
- **Підключення:** Команда docker-machine env допомагає легко підключитися до віддаленого хоста.

Приклад використання:

1. Створення віртуальної машини:

```
$ docker-machine create -d virtualbox my-docker-machine
```

Ця команда створює віртуальну машину my-docker-machine з Docker Engine на ній, використовуючи VirtualBox.

2. Підключення до хоста:

```
$ eval $(docker-machine env my-docker-machine)
```

Ця команда налаштовує ваше поточне середовище, щоб всі наступні команди docker і docker-compose виконувалися на віддаленій машині.

5. Висновок: Чому ці інструменти працюють разом?

Docker, Docker Compose та Docker Machine створюють повну екосистему для розробки.

- **Docker** забезпечує базову функціональність контейнеризації.
- **Docker Compose** спрощує управління складними, багатосервісними додатками.
- **Docker Machine** надає середовище, де все це може працювати, дозволяючи легко розгортати додатки з локальної машини в хмару.

Ця трійка дозволяє розробникам зосередитися на коді, не переймаючись налаштуванням середовища.