Docker-Compose – Container-Orchestrierung mit YAML und Docker

Inhaltsverzeichnis

- 1 Funktionen und Vorteile von Docker-Compose
- 2 Grundlegende Struktur einer docker-compose.yml-Datei
- 3 Wichtige Docker-Compose-Befehle
- 4 Vergleich: Docker vs. Docker-Compose
- 5 Wann sollte Docker-Compose verwendet werden?
- 6 Alternativen zu Docker-Compose
- 7 Fazit

Docker-Compose ist ein Werkzeug zur Verwaltung von Multi-Container-Anwendungen mit Docker. Es ermöglicht die Definition aller Dienste in einer einzigen docker-compose.yml-Datei und vereinfacht das Starten, Stoppen und Skalieren von Containern. Besonders nützlich für Entwickler und Testumgebungen.

Docker-Compose ist ein Werkzeug zur Definition und Verwaltung von **Multi-Container-Anwendungen** mit **Docker**. Es ermöglicht Entwicklern, mehrere Container in einer **einzigen YAML-Datei** (`docker-compose.yml`) zu definieren und sie mit nur einem Befehl zu starten.

1 Funktionen und Vorteile von Docker-Compose

- **Einfache Konfiguration**: Verwaltung mehrerer Container mit einer `docker-compose.yml`-Datei.
- **Automatisierung**: Starten, Stoppen und Neustarten von Containern mit wenigen Befehlen.
- **Netzwerkverwaltung**: Automatische Erstellung und Verbindung von Containern in eigenen Netzwerken.
- **Volumen-Management**: Gemeinsame Speicherung und Nutzung von Daten über mehrere Container hinweg.
- **Umgebungsvariablen**: Verwaltung von Konfigurationen ohne Änderungen am Code.
- **Skalierung**: Erhöhen oder Verringern der Anzahl laufender Container nach Bedarf.

2 Grundlegende Struktur einer docker-compose.yml-Datei

Eine typische `docker-compose.yml`-Datei sieht wie folgt aus:

Code

```
version:                                                                '3'
services:
  web:

    ports:

    volumes:

  database:

    environment:

      MYSQL_DATABASE: app_db
```

Alles anzeigen

Hierbei werden zwei **Dienste (Services)** definiert:

1. **Ein Webserver (nginx)**, der den Inhalt aus einem lokalen Verzeichnis (`./html`) bereitstellt.
2. **Eine MySQL-Datenbank**, die über Umgebungsvariablen konfiguriert wird.

3 Wichtige Docker-Compose-Befehle

- **Container starten:** `docker-compose up -d`
- **Container stoppen:** `docker-compose down`
- **Status anzeigen:** `docker-compose ps`
- **Logs anzeigen:** `docker-compose logs -f`
- **Dienste skalieren:** `docker-compose up --scale web=3`

4 Vergleich: Docker vs. Docker-Compose

Funktion	Docker (CLI)	Docker-Compose
Container-Management	Einzelne Container	Mehrere Container gleichzeitig
Konfigurationsformat	Manuelle CLI-Befehle	YAML-Datei zur Definition
Skalierung	Manuell mit mehreren `docker run` Befehlen	Automatisch mit `docker-compose up --scale`
Netzwerke & Volumes	Müssen einzeln definiert werden	Automatische Verwaltung

5 Wann sollte Docker-Compose verwendet werden?

- **Bei der lokalen Entwicklung von Multi-Container-Anwendungen.**
- **Für Testumgebungen**, in denen mehrere Dienste zusammenarbeiten müssen.
- **Zur einfachen Verwaltung von Microservices und verteilten Anwendungen.**
- **Wenn eine reproduzierbare Umgebung benötigt wird**, die unabhängig vom Host-System ist.

6 Alternativen zu Docker-Compose

- **Kubernetes** – Leistungsfähige Orchestrierungsplattform für produktive Cloud-Deployments.
- **Docker Swarm** – Integrierte Lösung für verteiltes Container-Management.
- **Podman-Compose** – Alternative für Podman-Nutzer ohne Docker-Daemon.

7 Fazit

Docker-Compose ist ein **leistungsstarkes Werkzeug**, um Multi-Container-Anwendungen einfach zu definieren und zu verwalten. Es eignet sich besonders für **lokale Entwicklungsumgebungen** und **Testsysteme**, da es eine schnelle und wiederholbare Bereitstellung ermöglicht.